Conceptualización del proceso de implementación de software: perspectivas ágil y disciplinada

Conceptualization of the software implementation process: agile and disciplined approaches

Castillo, Arístides*; Barrios, Judith; Montilva, Jonás y Rivero, Dulce
Postgrado en Computación. Universidad de Los Andes
Mérida, Venezuela
*aristides@ula.ve

Recibido: 17-06-2009 Revisado: 12-07-2010

Resumen

Este trabajo presenta en un modelo híbrido, el conjunto de conceptos que deben considerarse para especificar un proceso de implementación de software disciplinado con rasgos de agilidad. Se considera que los procesos de diseño detallado, codificación, verificación e integración de software, forman parte de la etapa de implementación del producto de software. El trabajo parte del análisis de las fases de implementación de software prescritas en los estándares metodológicos, marcos de trabajo y modelos conceptuales más conocidos y utilizados. Se analizan el modelo CMMI, los estándares IEEE 1074 e ISO/IEC 12207, el cuerpo de conocimiento de la ingeniería de software SWEBOK y el método WATCH en su versión empresarial. Para la perspectiva ágil se consideran la Programación eXtrema (XP) y el AgileUP. Los resultados de este trabajo sirven de base a una propuesta metodológica detallada y adaptada al contexto de las PyMEs venezolanas (Proyecto METHODIUS). Estos resultados son válidos y extensibles a otras organizaciones productoras de software en el contexto latinoamericano.

Palabras clave: Implementación de Software, enfoque ágil, métodos pesados, modelos conceptuales.

Abstract

Using a hybrid model, this paper presents a set of concepts that should be considered in the specification of a disciplined software implementation process with agile flavor. The processes of detailed design, coding, verification and software integration are considered as part of the software implementation phase. This work is based in the analysis of the software implementation phase description as found in some of the best known and used methodological standards, frameworks and conceptual models from both disciplined and agile software engineering. We analyzed the CMMI® model, IEEE 1074 and ISO/IEC 12207 standards, the Guide to the Software Engineering Body of Knowledge (SWEBOK) and the enterprise version of the WATCH method. As agile software engineering representatives we selected eXtreme Programming (XP and the AgileUP methods. Outcomes of this research will be used to create a detailed methodological proposal adapted to the context of the Venezuelan software development SMEs (METHODIUS Project). These results are valid and extensible to software development organizations in Latin American context.

Key words: Software implementation, agile approach, heavyweight methods, conceptual models.

1 Introducción

Recientemente, se ha promovido el empleo de la agilidad y la disciplina para llevar a cabo el desarrollo de productos de software dentro de restricciones de tiempo, costos y disponibilidad de recursos; esto, sin descuidar los aspectos de calidad tanto del producto como del proceso empleado para producirlo (Ambler, 2002), (Dutton y McCabe, 2005); (Pikkarainen y Mäntyniemi, 2006); (Anderson, 2005). En la literatura especializada, se encuentran autores e instituciones (Abran et al., 2001); (CMMI, 2006); (ISO, 2008); (IEEE, 1998); (Montilva et al., 2000); (Montilva y Barrios, 2003); (Beck, 2000) que

buscando abordar la problemática de la producción de software de calidad, han propuesto métodos, modelos de referencia y estándares que prescriben un conjunto de procesos, actividades y prácticas adaptables a gran diversidad de aplicaciones. En la mayoría de los casos se consigue establecer algunas similitudes entre ellos, por ejemplo, el hecho de centrarse en las actividades básicas del ciclo de desarrollo de software, como son el análisis, el diseño, la construcción y las pruebas de software (IEEE, 1998) e (ISO, 2008); en otros casos se consideran aspectos más detallados y específicos relacionados con la descripción de procesos, actividades y prácticas. (Ambler, 2002), (Abran et al., 2001), (CMMI, 2006), (Montilva et al., 2000); (Montilva y Barrios, 2003) y (Beck, 2000).

Un modelo de procesos ágil-disciplinado, debe guiar la implementación de productos de software que evolucionen, que se desarrollen de manera cíclica (por refinamiento, incrementos o mejora de producto), y que preconicen la participación activa del cliente/usuario durante dicho proceso. Por lo tanto, desde un punto de vista práctico, un proceso de implementación ágil/disciplinado debe incluir actividades que conlleven a la visualización, uso y prueba temprana de la funcionalidad del producto, permitiendo una puesta en operación de modo parcial y progresivo. Así, los desarrolladores de software, podrían observar la evolución del producto mediante la transformación progresiva o refinamiento de modelos conceptuales, implementables y operativos.

El presente trabajo presenta un modelo que representa el conjunto de conceptos, abstraídos de estándares, modelos y métodos, de implementación de software tanto ágil como disciplinado. Este modelo conceptual, enmarcado en el proyecto de investigación *METHODIUS* (FONACIT 2005000165), sirve de base a una propuesta metodológica de proceso de desarrollo de software, adaptada al contexto de las PyMEs venezolanas. Los resultados del trabajo, son extensibles a otras organizaciones de software dentro del contexto latinoamericano.

La investigación se inicia con el análisis y, consecuente, representación de los conceptos incluidos en cinco de los modelos, estándares y métodos, más conocidos y utilizados, ellos son: el Modelo Integrado de Capacidad y Madurez (CMMI®.) (CMMI, 2006), el Cuerpo de Conocimientos de la Ingeniería de Software SWEBOK (Abran et al., 2001), el Estándar ISO/IEC 12207 (ISO, 2008), el estándar IEEE 1074 (IEEE, 1998), y el método *Gray WATCH* (Montilva, et al., 2008). Luego, se analizan los representantes de la perspectiva ágil: Programación extrema XP (Beck, 2000) y *AgileUP* (Ambler, 2002). La selección de los representantes se basa en el sondeo realizado entre PyMEs y cooperativas en Venezuela, en el proyecto METHODIUS (Rivero et al., 2007). Finalmente, se comparan los modelos conceptuales elaborados buscando la integración de conceptos comunes y no comunes de ambas perspectivas. En este trabajo, el diseño detallado, la codificación, la verificación y la integración se incluyen en la fase de implementación de software.

El resto del documento se estructura de la siguiente manera: En las secciones 2 y 3 se presentan, las estructuras conceptuales asociadas a modelos, estándares y métodos de desarrollo, disciplinados y ágiles, respectivamente. En la sección 4, se contrastan las estructuras conceptuales de las secciones 2 y 3 y se extrae un modelo que integra ambas perspectivas. Por último, se presentan las conclusiones y el trabajo futuro.

2 Desarrollo disciplinado

La definición, adopción y mejora de un proceso de desarrollo, acorde con el contexto organizacional y el tipo de aplicación a desarrollar, ha promovido la investigación tanto a nivel de prácticas y procesos estándares como a nivel de la formación profesional en el desarrollo de las competencias requeridas para realizar las actividades propias de desarrollo de software. Esta sección presenta en primera instancia el modelo CMMI, luego el SWEBOK; posteriormente y, considerando que el ejercicio de la Ingeniería de Software es guiado por el establecimiento de estándares de desarrollo que definen un lenguaje común de expresión e interpretación de modelos, se presentan los estándares ISO/IEC 12207 (ISO, 2008) e IEEE 1074 (IEEE, 1998). Al final de la sección, se estudia el método *Gray WATCH* como representante de los métodos tradicionales de desarrollo de software.

2.1 Modelos integrados de capacidad y madurez (CMMI®)

Según (CMMI *Product Team*, 2006), los Modelos Integrados de Capacidad y Madurez (CMMI® por sus siglas en inglés) desarrollados por el *Software Engineering Institute* (SEI), para medir la capacidad y la madurez de los procesos de software. El CMMI® recomienda las mejores prácticas para la realización de actividades de desarrollo y de mantenimiento de software. El modelo establece 5 niveles para clasificar la madurez de los procesos organizacionales, en función de áreas de procesos que alcanzan sus objetivos y que son gestionadas con principios de ingeniería.

La implementación de software está incluida en el área de procesos denominada Solución Técnica, se corresponde con las metas *SG2: Desarrollar el diseño* y *SG3: Implementar el diseño del producto*. Esta área de procesos tiene como propósitos diseñar, desarrollar e implementar soluciones a los requisitos del sistema y, se enfoca en evaluar y seleccionar soluciones que potencialmente satisfagan un conjunto de requisitos establecidos, y en desarrollar diseños detallados para tales soluciones. Como sub-prácticas, se plantea el uso de métodos efectivos, tales como programación estructurada, programación orientada a objetos, generación automática de código reutilización de código, uso de patrones de diseño, la adhesión a criterios y estándares aplicables, la realización de revisiones de pares y de pruebas unitarias de los componentes implementados. La práctica de documentación contempla documentos que son necesarios para la instalación, operación y mantenimiento del producto: manuales de usuario, de entrenamiento, de operación, de mantenimiento y la ayuda en línea.

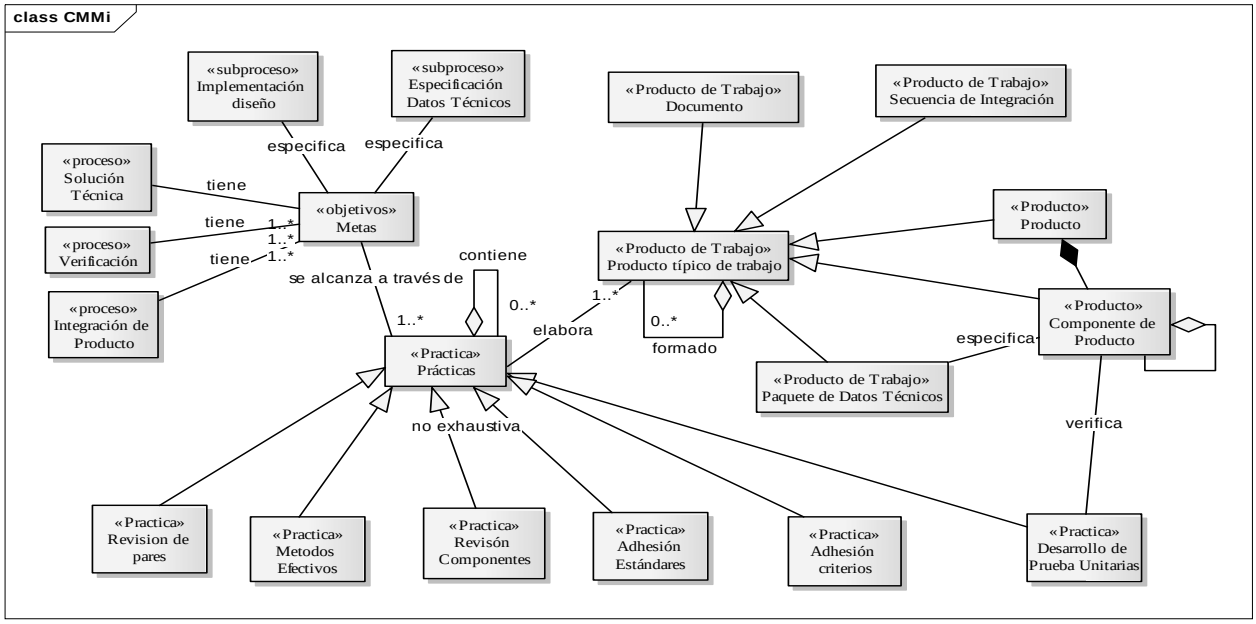


Fig. 1. Modelo conceptual del proceso de implementación de software según el modelo CMMI

El área de procesos de Verificación, contempla las metas y las prácticas necesarias para asegurar que los productos de trabajo y los componentes de producto cumplen con los requisitos especificados. Se utiliza la verificación formal de los requisitos, como un proceso que requiere ser planeado, ejecutado y analizado, generando sus correspondientes incidencias y métricas y, la realización de revisiones de pares en las actividades propias de la construcción del producto de software.

El área de procesos denominada Integración de Productos se incluye en el proceso de implementación. Las metas de esta área son: (1) Preparar la integración de producto (SG1). (2) Asegurar la compatibilidad de las interfaces (SG2). (3) Ensamblar los componentes de producto y entregar el producto (SG3). La Fig. 1 presenta un modelo conceptual derivado del análisis del proceso de Implementación de Software desde la perspectiva del modelo CMMI.

2.2 Cuerpo de conocimientos de la Ingeniería de Software (SWBOK)

El cuerpo de conocimientos de la Ingeniería de Software detalla el subconjunto del conocimiento aceptado como requerido para ejercer la profesión de la Ingeniería de Software (IEEE, 2008); (Abran et al., 2001).

La Construcción de Software (Implementación de Software) es una de las diez áreas de conocimiento del SWEBOK, definiendo el término Construcción de Software como "la creación detallada de software funcional y significativo por medio de la combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración del código" (Abran et al., 2001). Esta área tiene una estrecha relación con las áreas de Diseño y Pruebas de Software. Esto se debe a que el proceso de construcción de software involucra actividades significativas del diseño y de las pruebas. El área Construcción de Software, contiene fundamentos que marcan directrices a seguir al ejecutar las actividades incluidas; Entre ellos están (1) minimización de la complejidad: creación de código simple y legible antes que astuto; (2) anticipación del cambio: orientado hacia la preparación y facilitación de cambios en el código fuente debido a cambios en el entorno del software; y, (3) construcción para la verificación: inclusión de prácticas que permitan cotejar fácilmente, manual o automático, si el producto creado cumple con la especificación provista.

El área de Gestión de la Construcción contempla actividades de preparación y soporte, y de captura de métricas asociadas a las actividades de construcción; esto permite una revisión del desempeño del proceso de construcción, y su consecuente, mejora a lo largo del tiempo.

Entre las principales prácticas del área de Construcción de Software están: (1) la construcción involucra algún tipo de diseño detallado; (2) la actividad de codificación, debe procurar la legibilidad del código, el uso de estructuras de control y datos, la consistencia en la organización del código fuente, su documentación y su entonación; la aplicación de pruebas unitarias y de integración; (3) formalización de la reutilización al integrar procesos en el ciclo de vida del software; y, (4) la aplicación de técnicas de aseguramiento de la calidad como el desarrollo guiado por pruebas y las revisiones técnicas, entre otros.

La Fig. 2 muestra los conceptos involucrados en la construcción de software según la Guía SWEBOK.

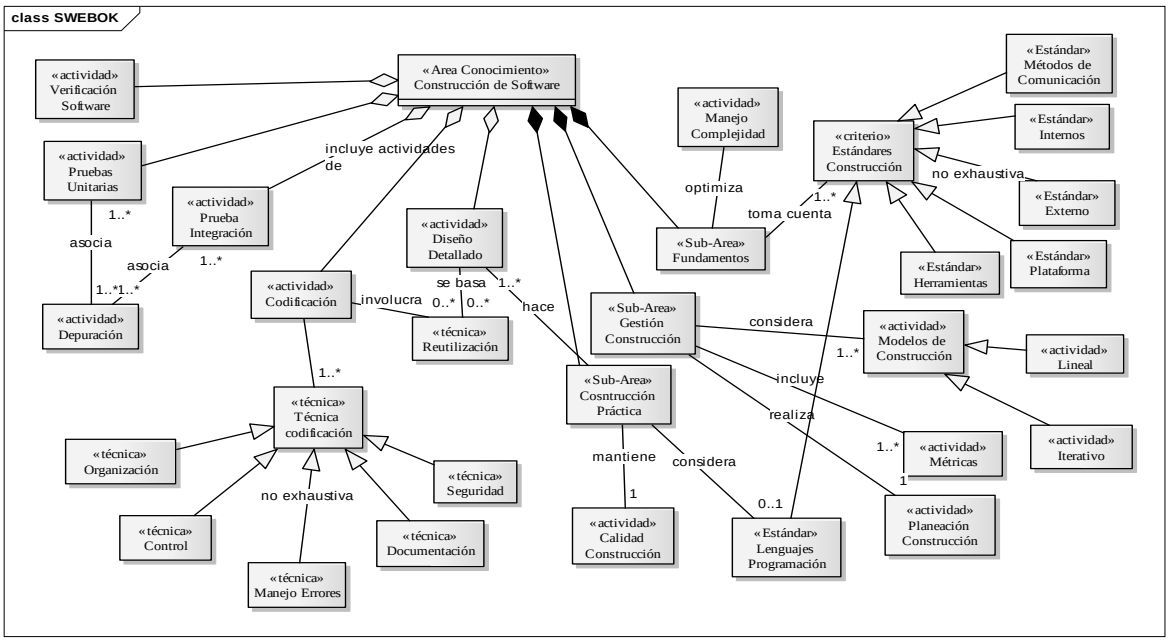


Fig. 2. Modelo conceptual del proceso de implementación de software según la guía del SWEBOK

2.3 Estándar ISO/IEC 12207

El estándar ISO/IEC 12207 desarrollado por la IEEE en 1995 (ISO, 2008), establece un marco común de trabajo para los procesos del ciclo de vida del software que pueden usarse como referencia en la industria del desarrollo de software. En dicho estándar se consideran los procesos, actividades y tareas que deben ser realizarse durante la adquisición de un producto o servicio de software y durante el suministro, desarrollo, operación, mantenimiento y disposición de un producto de software. ISO/IEC 12207, contempla tres grupos de procesos: principales, generales y soporte. En la Fig. 3 se resumen los conceptos manejados por este estándar.

Dentro del grupo de procesos principales se encuentra el proceso de desarrollo, que contiene las actividades de diseño, de codificación, pruebas e integración del producto. Las actividades relacionadas con los procesos de construcción, verificación e integración de software son las siguientes:

- Diseño detallado del Software: refinación del diseño de los componentes del software a niveles más bajos para la codificación, compilación y pruebas. Las interfaces externas, diseño interno de cada componente, la base de datos y los requisitos de las pruebas de integración, deben ser diseñados explícitamente con suficiente nivel de detalle.
- Codificación y pruebas del software: creación simultánea del código, de la base de datos, los datos y procedimientos de pruebas unitarias, de verificación y de validación de componentes, así como su documentación.
- Integración del Software: creación de un plan de integración de unidades y componentes de software y generación de procedimientos y elementos para la realización de las pruebas de integración.

2.4 Estándar IEEE 1074

El estándar IEEE 1074 (IEEE, 1998) describe el conjunto de actividades y procesos obligatorios para el desarrollo y mantenimiento del software, estableciendo un marco común para el desarrollo de modelos de ciclo de vida.

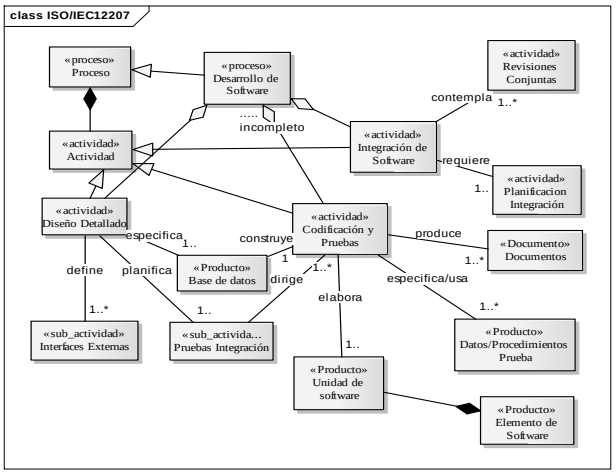


Fig. 3. Modelo conceptual del proceso de implementación de Software según el estándar ISO/IEC 12207

Los procesos del ciclo de vida del estándar IEEE 1074, enmarcan los procesos de construcción, las pruebas e integración de software en actividades y sub-actividades:

- **Diseño**:

- Realizar diseño detallado: selección de alternativas de diseño. Como salida de la actividad están: la estructura de da-

tos, el algoritmo y la especificación de la información de control de cada componente.

• **Construcción**:

- Crear código ejecutable: código fuente que se puede compilar y los comentarios incluidos en el código.
- Crear documentación operativa: documentación necesaria para la instalación, operación y soporte del software.
- Realizar la integración del software: componer e integrar, por separado, los componentes de software para que conformen un solo producto entregable.

• **Evaluación**:

- Conducir revisiones: de diseños, implementaciones, documentación e integración del producto. Estas revisiones producen indicadores de gestión sobre el proceso ejecutado, y sobre los productos de trabajo.

En la Fig. 4, se muestra el conjunto de conceptos derivados del análisis del estándar IEEE1074.

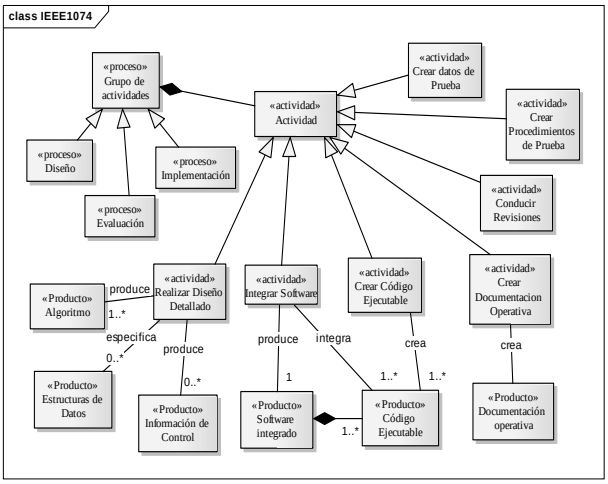


Fig. 4. Modelo conceptual del proceso de implementación según el estándar IEEE 1074

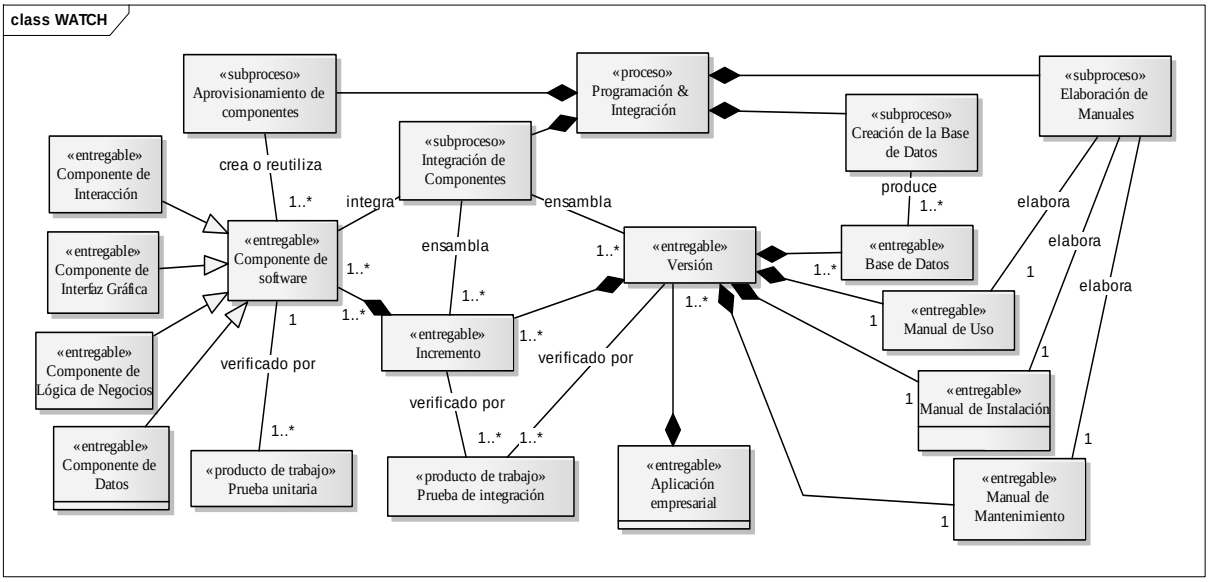


Fig. 5. Modelo conceptual del proceso de implementación en el contexto del Método WATCH

2.5 Método WATCH

El método WATCH es una guía metodológica dirigida al desarrollo de software basado en componentes. WATCH proporciona una visión clara de los procesos de desarrollo de componentes (p.ej., COTS y *Web Services*) y de aplicaciones distribuidas (p.ej. Aplicaciones Web) (Hamar, 2004). Este método ha evolucionado en diversas versiones (Montilva et al., 2000); (Montilva y et al, 2003); (Montilva et al., 2008).

El método contempla, dentro del grupo de Procesos de Implementación, los procesos de Programación & Integración y Pruebas. El proceso de Programación & Integración se encarga de producir, probar e integrar los componentes arquitectónicos de la aplicación, en cada una de sus versiones y consiste en: elaborar, codificar y/o adaptar cada uno de los componentes que integran las diferentes versiones de la aplicación; probar cada componente como una unidad; integrar estos componentes de acuerdo a la arquitectura diseñada; y probar la integración de estos componentes (Montilva et al., 2008).

El proceso de Pruebas de la Aplicación, la verifica y valida para asegurarse que cumple con los requisitos especificados y satisface las necesidades que tienen sus usuarios. El proceso consiste en verificar cada versión de la aplicación como un todo y depurar los errores encontrados. Las pruebas se realizan a tres niveles: 1) Unidad: cada componente es probado separadamente; 2) Integración: se prueba la integración de los componentes y sus interacciones; y, 3) Sistema: la versión de la aplicación se prueba como un todo. Las pruebas de unidad y de integración tienen lugar durante

el proceso de Programación & Integración; las pruebas de sistema se realizan en el proceso de Pruebas de Aplicación.

Los Procesos de Implementación producen productos técnicos intermedios y/o finales: Especificaciones de Pruebas, Mecanismos de Pruebas, Componentes de Software, Incrementos, Bases de Datos, Manual de Instalación, Manual de Uso, Manual de Mantenimiento, Versión de la aplicación y Aplicación Empresarial completa. Entre las prácticas propuestas se encuentran las siguientes: Uso de documentos de diseño detallado actualizados por los programadores, así como los respectivos modelos de diseño que puedan estar disponibles; Programación guiada por pruebas (*Test-Driven Development*), donde las pruebas de unidad son diseñadas y preparadas previamente a la codificación del componente; Reutilización de software como principio guía para el aprovisionamiento de los componentes que se utilizarán; Integración Incremental de Componentes e incrementos, previo a la generación de versiones; y, Inclusión de Revisiones Técnicas, Análisis de la Trazabilidad y Pruebas del Software como procesos técnicos de verificación de los productos finales e intermedios.

En la Fig. 5, se representan los conceptos manejados por el método *WATCH*.

3 Desarrollo ágil

El Manifiesto Ágil (Fowler y Highsmith, 2001), propuesto por un grupo de representantes de Programación eXtrema, *SCRUM*, *DSDM*, *Adaptative Software Development*, *Crystal*, *FDD*, *Pragmatic Programming*, expresa que hay que dar mayor valor a los individuos y sus interacciones antes que a los procesos y las herramientas, al software que funcione antes que a una documentación detallada, a la participación del cliente antes que a la negociación del contrato, y a responder al cambio antes que seguir un plan estricto. El manifiesto establece doce principios para estas *mejores maneras* de desarrollar software: (1) mayor prioridad a la satisfacción del cliente: entrega temprana y continua de software valioso. (2) bienvenida a los requisitos cambiantes, incluso tarde en el desarrollo del software. (3) entrega frecuente de software que funciona, desde un par de semanas hasta un par de meses a escalas de tiempo cortas. (4) trabajo conjunto entre personas del negocio y desarrolladores. (5) individuos motivados, proporcionando el entorno y el apoyo que necesitan y confiando en que ellos realizarán la tarea. (6) conversación cara a cara como método eficiente y efectivo para transmitir la información a y dentro de un equipo de desarrollo. (7) software funcionando es la medida primaria de progreso. (8) desarrollo sostenible, patrocinantes, desarrolladores y usuarios capaces de mantener un ritmo constante de manera indefinida. (9) atención continua a la excelencia técnica y al buen diseño. (10) simplicidad esencial como el arte de maximizar la cantidad de trabajo que no se hace. (11) Equipos auto-organizados. (12) Reflexión en equipo acerca de cómo ser más efectivos, entonar y ajustar su comportamiento en consecuencia.

3.1 Programación eXtrema (XP)

La Programación Extrema (XP, por *eXtreme Programming*) creada por Kent Beck (Wells, 2006), está basada en los valores de simplicidad, comunicación, retroalimentación y valor (coraje), que propone a los equipos de trabajo, la implantación de prácticas simples que les permite recibir retroalimentación de la situación actual del proyecto y ajustar las prácticas a dicha situación específica (Jeffries, 2001). XP tiene como meta reducir el costo del cambio mientras que las metodologías tradicionales buscan que los requisitos sean determinados y establecidos al comienzo del proyecto de desarrollo procurando que así permanezcan de ahí en adelante (Beck, 2000).

En el Proceso de Implementación, englobado como un episodio de desarrollo protagonizado por equipos de dos programadores, se observa la auto-organización, la asignación de las tareas a realizar, la comunicación descentralizada con el cliente, la especificación funcional en forma de casos de prueba, el diseño evolutivo en forma de refactorización, el diseño detallado en forma de pruebas unitarias y el proceso de integración de los componentes como actividad del día a día. XP propone prácticas para la construcción del software, entre ellas: Programación guiada por pruebas (TDD); Mejoras evolutivas del diseño (Refactorización); Integración Continua; Pertenencia colectiva del código[1]; Diseño simple y Adhesión a estándares de codificación

El proceso de Verificación relacionado con la revisión constante del producto, mediante la Programación en Pares y la ejecución continúa de pruebas unitarias. El proceso de Integración manifestado en la práctica de Integración Continua, que promueve la disponibilidad de una última versión del código a la cual se le integran componentes ya probados de manera unitaria. La Fig. 6 muestra los conceptos manejados por la disciplina XP.

3.2 AgileUP

El Proceso Ágil Unificado (*Agile UP*, por sus siglas en inglés) es un enfoque de desarrollo de software basado en el *Unified Process* (UP) (Ambler, 2002). El ciclo de desarrollo del *Agile UP* es serializado en sus procesos generales, iterativo a nivel detallado y entrega versiones incrementales del producto a lo largo del tiempo (Ambler, 2002).

AgileUP maneja los conceptos de entregables, productos de trabajo empresariales y otros productos de trabajo. Los entregables son aquellos productos de trabajo que deben ser producidos. Los otros productos de trabajo son aquellos que no se requiere mantenerlos en el tiempo. Los productos de trabajo empresariales son aquellos que son mantenidos dentro de la organización TI y son compartidos entre proyectos. Entre las prácticas recomendadas están: Mantener los productos de trabajos simples y concisos; Menos documentación de la que se puede pensar; Trabaja de manera cercana al cliente/usuario y se

[1] Representada como “Código colectivo” en la Fig. 6.

produce sólo aquello que realmente se necesita; Varias personas discutiendo alrededor de un pizarrón es la mejor forma de comunicar información; Considerar el uso de plantillas *open-source* para crear las propias.

En la fase de Construcción se desarrolla el software hasta que se encuentra listo para las pruebas de pre-producción. En las fases previas, la mayor parte de los requisitos han sido identificados y la arquitectura del sistema se encuentra en la línea base. Así, el énfasis se concentra en priorizar, y en entender, los requisitos, en hacer tormentas de ideas sobre las soluciones propuestas, y en codificar y probar el producto. En caso de ser necesario, las primeras versiones del producto son desplegadas, interna o externamente, con el objeto de obtener retroalimentación por parte del usuario. La fase de Construcción culmina cuando el equipo de desarrollo alcanza a desarrollar un producto operacional.

El Proceso de Implementación de software se encuentra dentro de la Disciplina de Implementación, cuya meta es transformar los modelos en código ejecutable y realizar un nivel básico de pruebas, refiriéndose a las pruebas unitarias.

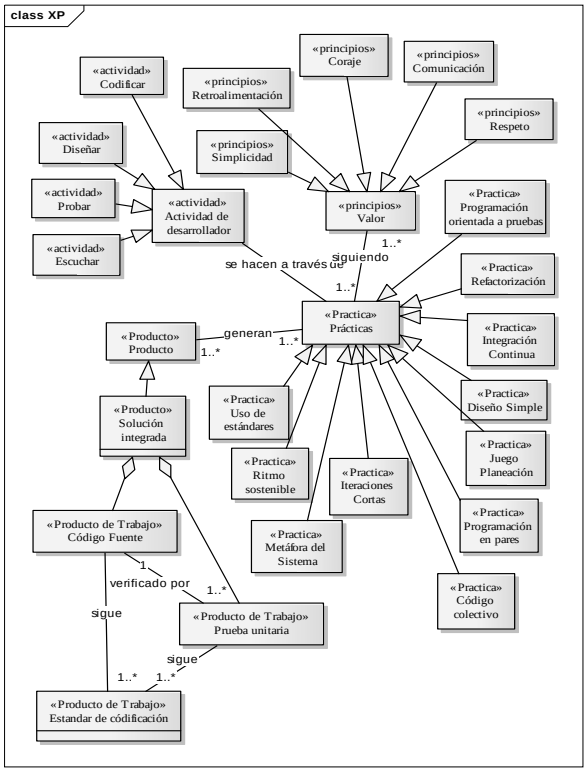


Fig. 6. Modelo Conceptual del Proceso de Implementación de Software según Programación Extrema (XP)

El flujo de trabajo de esta disciplina se describe a través de subprocesos y tareas:

- Desarrollo de la aplicación: codificación y prueba del producto final. Entre las tareas que se llevan a cabo están: Escribir las pruebas unitarias; Escribir el código de producción; Ejecutar las pruebas unitarias; Perfilar el desempeño del sistema y Gestionar las dependencias en el código.
- Construir el sistema: compilación y empaquetado del sistema para su despliegue. Las tareas son: Compilación (Integración) continúa del sistema; Promover al entorno de integración del proyecto.
- Desarrollar la Base de Datos con las tareas Evolucionar el esquema de base de datos y Gestionar los derechos de acceso.

La disciplina de Implementación en la fase de Construcción, presenta como actividades primarias el desarrollo evolutivo de la lógica del dominio, de la interfaz de usuario y del esquema de datos; el desarrollo de interfaces a sistemas legados y la escritura de *scripts* de conversión de datos. *AgileUP* recomienda incluir prácticas que tienen como fin primordial la agilización de las actividades de implementación: Programación por pares, Desarrollo guiado por pruebas (TDD), Integración Continua, Modelar antes de codificar, Adoptar un estándar de codificación, Refactorízar el código y los esquemas de bases de datos, y tener ambientes separados para el desarrollo, las pruebas y la producción.
El Proceso de Integración se traduce en la actividad continua de compilación del sistema. Esta compilación se realiza cada vez que el código fuente cambia. Así, el repositorio de código fuente se encuentra, en todo momento, en un estado estable respecto a las nuevas funcionalidades que han sido agregadas al producto. El conjunto de conceptos de *AgileUP* se muestra en la Fig. 7.

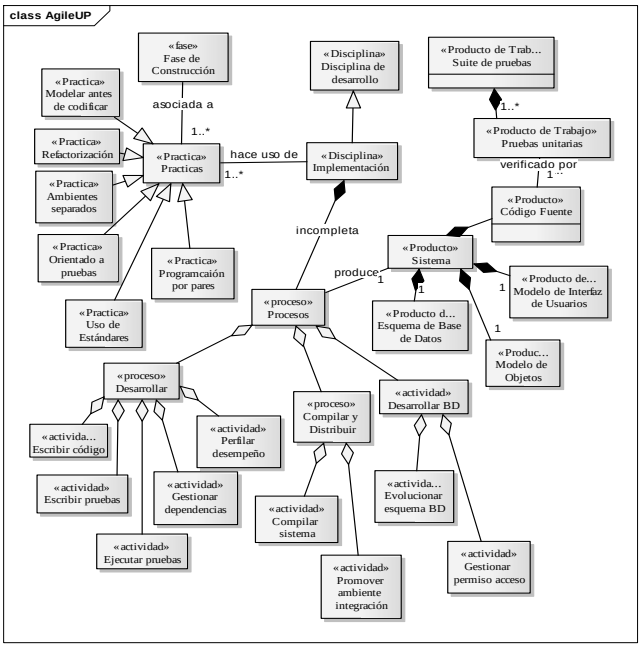


Fig. 7. Modelo Conceptual del Proceso de Implementación de Software según AgileUP

4 Proceso de implementación disciplinado vs. proceso de implementación ágil

Se analizan y se comparan los distintos modelos conceptuales construidos para establecer de manera clara y precisa la relación y el grado de similitud y/o distanciamiento

entre los métodos y modelos de desarrollo ágiles estudiados contra los modelos tradicionales. Se busca integrar, conceptos, prácticas y actividades comunes o similares a ambas perspectivas, con aquellos otros elementos que son propios de cada una de ellas y que las distinguen.

4.1 Similitudes

- Acuerdo en establecer como requisito previo, la existencia de algún tipo de diseño detallado (formal o informal) antes de iniciar la actividad de codificación del producto de software.
- Se considera elemento crucial, la inclusión de pruebas unitarias sobre el producto y la definición del modelo de procesos de desarrollo de software basado en pruebas, ya sea de manera explícita (ágiles) o de manera implícita (diseño de los casos de prueba antes de codificar en los tradicionales).
- Se promueve la existencia de estándares de codificación y es vital la adhesión a dichos estándares por parte de los desarrolladores.

El Código fuente es objeto de revisiones regulares para asegurar su calidad, que es correcto y el grado de adhesión a los estándares de codificación predefinidos.

4.2 Diferencias

- En representantes disciplinados, el documento de diseño detallado es necesario e incluye documentos y modelos. En la perspectiva ágil, el diseño detallado se logra con la creación de la prueba unitaria y con el código fuente, legible y documentado, sin que se requiera algún documento o artefacto adicional.
- En representantes disciplinados, el proceso de integración se realiza a posteriori a la codificación de los componentes de la aplicación, en tanto que en los ágiles este proceso es sustituido por la práctica de integración continua. Como consecuencia de lo anterior, en los representantes tradicionales existen, de manera explícita, las pruebas de integración, en tanto que en los ágiles, éstas son pruebas no tan unitarias, que solo se distinguen de las reales por probar más de un componente o por su integración con algún componente externo.
- La Revisión Técnica del código es una práctica propuesta en el modelo CMMI para realizarla entre pares. En el método WATCH la realiza un grupo de expertos organizados por el grupo de V&V. En el caso de los representantes ágiles, la revisión de pares, se plantea de manera continua y regular para asegurar la calidad del código, sin que por ello se requiera una revisión adicional externa.
- Los representantes disciplinados presentan una diversidad de productos de trabajo, entre ellos, documentos y modelos de diseño resguardados y que deben ser actualizados ante cualquier cambio. En los ágiles, existe la temporalidad de los productos de trabajo, en algunos casos son descartables, y en otros no se prescriben como necesarios, reduciendo así la carga adicional de resguardo y actualización ante cambios.
- Todos los modelos disciplinados mencionan el proceso de documentación como parte de la implementación. Esta documentación es operativa e incluye manuales de sistema, de usuario, ayuda en línea y de diseño. En el caso de AgileUP se contempla solo la documentación que debe ser considerada entregable, más no los productos de trabajo internos; el método XP no propone documentación alguna.
- Los métodos ágiles incluyen la refactorización del código, es decir, el cambio a discreción del desarrollador para hacer que un código complejo realice la misma funcionalidad, de manera más sencilla. Esta actividad discrecional del desarrollador no aparece mencionada o propuesta por ninguno de los representantes disciplinados.

Esta última diferencia abre un poco más la brecha entre ambas perspectivas, resaltando a la faceta humana de quienes desarrollan el producto como elemento diferencial clave. Si bien la refactorización a discreción no tiene prohibición expresa en ninguno de los representantes disciplinados, su aplicación se puede ver obstaculizada por el cambio cultural que representa para una organización que aplica habitualmente el enfoque disciplinado, el hecho de otorgar tal poder de acción a un desarrollador.

4.3 Modelo conceptual integrado de un proceso de implementación de software ágil y disciplinado

Como se mencionó previamente, el objetivo de este trabajo es establecer el conjunto de conceptos asociados a la definición de un proceso de implementación de software equilibrado que integre características de las perspectivas ágil y disciplinada. La Fig. 8 muestra que el proceso propuesto tiene como subprocesos Construcción, Integración y Verificación. Es importante aclarar que no hay ningún modelo de ejecución ni implícito ni explícito para estos subprocesos, sólo es un enunciado de la estructura del proceso de implementación equilibrado.

El subproceso de Construcción tiene como actividades el diseño detallado de los componentes a implementar, la codificación y la documentación - ya sea en el código fuente y, la generación de documentación de operación y alguna otra documentación que haya sido establecida como entregable del proyecto. El subproceso de Integración abarca las actividades de planificación de la integración de la aplicación, el ensamblado de los componentes a ser integrados y la ejecución de las pruebas de integración. Estas actividades son realizadas en el contexto de la práctica de Integración Continua. Las pruebas de Integración se realizan como parte de las pruebas unitarias. Estas pruebas unitarias están asociadas a la práctica de desarrollo de software basado en las pruebas del producto.

El subproceso de Verificación contiene las actividades de realización de revisión de pares (*Peer Review*) y la verificación detallada, realizada por miembros externos al equipo de desarrollo.

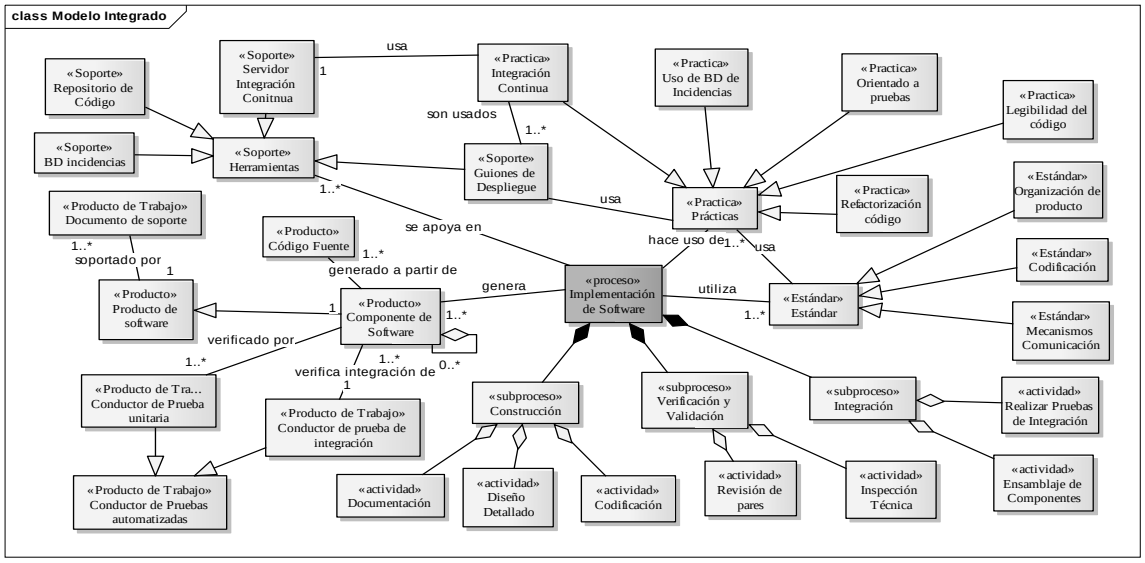
El proceso global de implementación se apoya en estándares, los cuales se pueden conseguir a través de medios de públicos (internet) o privados (experiencia interna de la organización), el estilo de codificación y la organización de los componentes. En relación a las herramientas de apoyo a los subprocesos, se recomienda usar: el repositorio de código o sistema de control de versiones, la base de datos de incidencias, los guiones de generación del despliegue y etiquetado y el servidor de integración continua. Además, se tiene el Desarrollo guiado por pruebas y la refactorización del código, que consiste en el cambio de implementación, sin afectar la interfaz ni el funcionamiento del componente, para obtener un código más limpio, legible y de mayor calidad.

4.4 Lineamientos de la propuesta conceptual

Considerando que, además, de las diferencias conceptuales y pragmáticas, observadas entre los modelos y los términos utilizados, se debe tener en cuenta la influencia que la cultura organizacional tiene sobre el proceso de implementación; es decir, la experiencia y el modo de trabajo de los miembros que constituyen el equipo de desarrollo (Fowler y Highsmith, 2001) influye en la factibilidad de realizar o no una práctica de software. Así, esta propuesta se define basándose en la inclusión de los principios y prácticas propuestas por los métodos ágiles dentro de los procesos, actividades y prácticas propuestas en la perspectiva disciplinada.

La documentación, se minimiza a nivel interno manteniendo los documentos entregables y los de relación con el cliente. Estos lineamientos se resumen a continuación:

- Establecer un mínimo de documentación básica, dejando el resto como documentación opcional. La decisión de generar algún documento opcional será de acuerdo al cliente, o por el valor que se considere éste agrega al software en desarrollo.
- Debe tenerse presente que todo producto de trabajo que se mantenga, agrega esfuerzo y carga de trabajo durante el desarrollo, lo que pudiera ir en detrimento de la agilidad del proceso de implementación. Se sugiere hacer uso de herramientas automatizadas de apoyo al desarrollo que se encarguen de la especificación del producto contribuyendo a la disponibilidad y facilidad de actualización de documentos. Así, se sugiere minimizar la generación de especificaciones y de *modelos del diseño detallado*, especialmente, si deben ser actualizados por separado al código fuente y a las pruebas unitarias. Se sugiere apoyar el proceso a través de una herramienta de generación de código que tome como entrada el modelo o la especificación de diseño detallado.
- Asumir que el proceso de Integración de Componentes se hace como una práctica de *Integración Continua*, realizada con cada nuevo incremento. Las pruebas de integración se convierten en pruebas unitarias realizadas por el desarrollador. Esto requiere tener la capacidad de generar versiones y ejecutar pruebas unitarias de manera automatizada, con algún *script* de generación de versiones y despliegue.
- Promover la *Refactorización* del código. Es un paso importante para la evolución de la calidad del código fuente, y suscita que el desarrollador, no sólo esté pensando en la solución de la funcionalidad a completar o error actual a corregir, sino en la constante inspección de su trabajo.
- Enfatizar la importancia de la *calidad y la legibilidad del código fuente*. Un código con estas características puede ser usado como su propia documentación y modelo, además de facilitar el mantenimiento y la inclusión de nuevos desarrolladores a mitad del proyecto.



8. Modelo conceptual integrado de la implementación de software ágil y disciplinado

5 Conclusiones

Se presentó una propuesta conceptual de especificación de un proceso de implementación de software ágil y disciplinado. Esta propuesta integra, de modo complementario, las prescripciones típicas de modelos tradicionales o disciplinados con prácticas ágiles, buscando un equilibrio entre formalidad y agilidad que mejore el desempeño y la productividad en el proceso de desarrollo, sin atentar contra la calidad del producto que se elabora.

El trabajo realizado permitió establecer las diferencias entre los modelos disciplinados y ágiles, resaltando, no solo las diferencias en prácticas, actividades o productos de trabajo, sino también, en el espíritu que las fundamenta. Se destaca que el desarrollo ágil tiene como objetivo primordial satisfacer las necesidades del cliente, sin que se afecte el logro de un producto técnico de alta calidad. Se reconoce la influencia de la cultura organizacional – tradicional, sobre los procesos de desarrollo, la cual intenta mantener el control permanente de las actividades que se realizan, limitando el grado de discrecionalidad o de toma de decisiones por parte de las personas que participan en el proyecto. Este último punto representa una de las diferencias más importantes entre ambas perspectivas.

La propuesta considera que un producto de software bien documentado tiene mayor probabilidad de tener una vida útil larga, soportada por la disponibilidad de documentación técnica actualizada. Así, se plantea complementar la definición del proceso de implementación con la inclusión de prácticas ágiles y con el uso de herramientas automatizadas eliminando la sobrecarga de trabajo de mantener modelos y documentos intermedios. La propuesta preconiza la obtención de productos de software de calidad y bien especificados y documentados, reduciendo el tiempo de entrega del producto funcional.

El trabajo futuro se orienta a la definición formal y validación del modelo de procesos que contenga los conceptos de la propuesta. Los procesos, actividades y prácticas serán representados a través de fragmentos de métodos reutilizables. Las relaciones entre fragmentos son representadas por mapas de rutas, los cuales permiten describir el producto de software y el proceso que transforma el producto, las situaciones y las decisiones implicadas en cada transformación. Los fragmentos de método deberán poder seleccionarse e integrarse según necesidades del desarrollador, como parte de las versiones del método *WATCH (Gray, Blue, White)* que se concretan en el Proyecto *METHODIUS* (FONACIT 2005000165).

Agradecimientos

Este trabajo ha sido financiado por el Proyecto FONACIT 2005000165, y contribuye a alcanzar los objetivos de los sub-proyectos 4 y 5. (www.methodius.info.ve).

Referencias

Abran A, Bourque P, Dupuis R, y Moore JW, Eds, 2001, Guide to the Software Engineering Body of Knowledge SWEBOK. IEEE Press.

Ambler S, 2002, Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process, John Wiley & Sons, New York.

Anderson DJ, 2005, "Stretching Agile to fit CMMI Level 3 the story of creating MSF for CMMI® Process Improvement at Microsoft Corporation" Conference, IEEE Computer Society, Denver.

Beck K, 2000, Extreme Programming Explained: Embrace Change, Addison-Wesley.

CMMI Product Team, 2006, CMMI® for Development, Version 1.2. CMU/SEI-2006-TR-008. Carnegie Mellon/ Software Engineering Institute, Pittsburgh.

Dutton J, McCabe, R, 2005, Agile/ Lean Development and CMMI®. presentado en SEPG 2006, SEI, Tennessee.

Fowler M; Highsmith J, 2001, The Agile Manifesto. Dr. Dobbs, ed. Agosto, 2001, en línea http://www.ddj.com/architect/184414755.

Hamar V, 2004, Aspectos metodológicos del desarrollo y reutilización de componentes de software. Universidad de Los IEEE Standards Board, 1998, IEEE Std 1074-1997. IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society, New York.Andes, Postgrado en Computación, Mérida.

IEEE Computer Society, 2008, Guide to the Software Engineering Body of Knowledge website. http://www2.computer.org/portal/web/swebok.

ISO 2008. ISO/IEC 12207:2008. Systems and software engineering Software life cycle processes. http://www.iso.org/iso/catalogue_detail?csnumber=43447

Jeffries R, 2001, Extreme Programming. http://www.xprogramming.com.

Montilva J, Barrios J, 2003, A Component-Based Method for developing Web Applications. Proceedings 5th International ICEIS'2003, Angers, France.

Montilva J, Barrios J Y Rivero D, 2008, Gray WATCH: Proyecto Methodius. ULA, GIDyC. Mérida. http://www.methodius.info.ve

Montilva J, Hamzan K y Gharawi M, 2000, The Watch Model for Developing Business Software in Small and Midsize Organizations. publicado en Proceedings of the IV World Multiconference on Systemics, Cybernetics and Informatics – SCI'2000, Orlando.

Pikkarainen M y Mäntyniemi A, 2006, An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies, SPICE 2006 Conference, ISO, Luxemburg.

Rivero D, Montilva J, Granados G, Barrios J, Besembel I y Sandia B, 2007, "La Industria de Software en Venezuela Una caracterización de su recurso humano". X Workshop IDEAS'07, EVETIS'07, pp. 435-443.

Wells D, 2006, Extreme Programming: A gentle introduction. http://www.extremeprogramming.org.