Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 3, Mes Marzo, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 34-55
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Gestión de Proyectos Informáticos
Recibido: 02/01/18 | Aceptado: 10/03/18 | Publicado: 30/03/18

Modelos de desarrollo de software. ¿Cuál elegir ?

Software development models. Which to choose ?

Angel Miguel Hernández Oliva [1*], Bárbara Bron Fonseca[2], Liliannes Caridad Matamoros Benitez[3]

[1]Empresa de Telecomunicaciones del CITMA, CITMATEL. La Habana, Cuba. ahernandez@ceniai.inf.cu
[2]Universidad de las Ciencias Informáticas, La Habana, Cuba. bbron@uci.cu
[3]Universidad de las Ciencias Informáticas, La Habana, Cuba. lbmatamoros@uci.cu

[*] Autor para correspondencia: ahernandez@ceniai.inf.cu

Resumen
La elección del modelo de desarrollo de software siempre es una disyuntiva a seguir en tanto se desea comenzar un nuevo proyecto o tarea; o simplemente querer aumentar la producción de software, mejorando a la vez la calidad. En este artículo se hace una revisión de los principales modelos de desarrollo de software existentes hasta la actualidad. El objetivo principal trazado es poder definir qué modelo de desarrollo de software emplear en dependencia al ecosistema donde se desarrolla su producción; así como futura implementación. Para analizar los diferentes modelos de desarrollo de software, se empleó como metodología la revisión bibliográfica y el análisis de un caso de estudio; lo cual permitió llegar a conclusiones referentes a qué modelo emplear en dependencia de la situación concreta del mismo. Los resultados alcanzados en este artículo vierten información relevante sobre la impetuosa necesidad de siempre adaptar la elección del modelo de desarrollo de software a seguir; dado que la misma variará en dependencia del ambiente y supuestos que se sigan tanto por la empresa productora como dentro del ecosistema en el cual se explotará el software. Por otro lado gana importancia en cuanto a la actualización de la información brindada; así como a la definición de métricas para efectuar la comparación entre los diferentes modelos de desarrollo de software.

Palabras clave: modelo, desarrollo, software, proyecto, proceso

Abstract
The choice of software development model is always a dilemma to follow as long as you want to start a new project or task; or simply want to increase software production, while improving quality. In this article, a review is made of the main models of software development existing up to now. The main objective is to define which software development model to use depending on the ecosystem where its production is developed; as well as future implementation. To analyze the different models of software development, the bibliographic review and the analysis of a case study were used as a methodology; which allowed to reach conclusions regarding which model to use depending on the concrete situation of the same. The results achieved in this article provide relevant information on the impetuous need to always adapt the choice of software development model to follow; since it will vary depending on the environment and assumptions that are followed both by the production company and within the ecosystem in which the software

will be exploited. On the other hand, it gains importance in terms of updating the information provided; as well as the definition of metrics to make the comparison between the different models of software development.

Keywords: *model, development, software, project, process.*

Introducción

El desarrollo de software es el centro de las actividades concernientes a los sistemas informáticos. Esta conclusión es basada en que el desarrollo de software está presente tanto en el desarrollo de aplicaciones dentro del área como fuera del área como pudiese ser un programa que controla la asignación de salas de embarque en un aeropuerto. Por otro lado, vale destacar que el desarrollo de software no es una tarea solamente técnica, sino que también involucra a terceros interesados, como por ejemplo: el usuario o cliente al cual está dirigido. De ahí que el éxito de un software es que satisfaga las necesidades del usuario; el cual no es siempre el mismo que el programador, que sea confiable, que sea realizado con los recursos estimados y que sea seguro. De ahí que gane primordial importancia gestionar de forma eficaz y eficiente los esfuerzos del desarrollo de software a través de los modelos de desarrollo de software (Ridel y Buemo 2007).

Los principales modelos de desarrollo de software son(Munassar y Govardhan 2010):

- Modelo cascada.
- Modelo prototipo.
- Modelo de desarrollo rápido de aplicaciones.
- Desarrollo evolutivo.
- Modelo en espiral.
- Modelo incremental (iterativo).
- Software basado en componentes (fábrica de software, líneas de producto software, ecosistema de software).

Este artículo tiene como objetivo establecer un criterio a seguir en el momento de definir qué modelo de desarrollo de software escoger para llevar a cabo soluciones de desarrollo de software; lográndose el mismo a través del análisis de un caso de estudio.

En este artículo se describen y analizan diferentes modelos de desarrollo de software los cuales serán: modelo en cascada, modelo de desarrollo incremental, modelo en espiral, factoría de software, líneas de producto de software (LPS) y ecosistema de software. Por otro lado, se hace una breve introducción de los conceptos básicos, se describen las fases que los constituyen y se presentan las ventajas y desventajas de los mismos; se definen métricas para efectuar la comparación entre los diferentes modelos; pudiéndose efectuar de esta forma una comparación entre los diferentes modelos de desarrollo de software descritos en este artículo. Por último, a través de un caso de estudio, se emite un criterio de elección de dichos modelos en correspondencia a las características del proyecto en el cual se desea aplicar; para luego arribar a conclusiones.

Modelo de cascada.

El modelo de cascada original se desarrolló sobre la década de los 70 donde se definió como una secuencia de actividades, siendo su eje fundamental un progreso del desarrollo de software a través de pautas de revisión bien definidos mediante entregas programadas con fechas precisas. La Figura 1 muestra un diagrama del modelo de cascada que describe el orden de las actividades del desarrollo de software. Vale apreciar, la ausencia explícita de una etapa de documentación; esto es motivado a que la misma se efectúa a lo largo de todo el desarrollo. Originalmente el modelo no comprendía el retorno a actividades anteriores; pero una posterior revisión amplió el concepto, permitiendo de esta forma retornar a actividades ya efectuadas(Scacchi 2001).

En el modelo de cascada la documentación técnica es comprensible tanto para usuarios como para administradores no técnicos. Todos los requisitos del desarrollo tienen que ser enunciados al inicio del proyecto. Las pruebas del software y su evaluación ocurren al final del proceso. Lo anterior destaca la rigidez del modelo en cuanto a la posibilidad de insertar nuevos requerimientos, modificar los existentes o eliminarlos (lo cual es altamente importante tomando como base la dificultad de poder definir todos los requisitos al inicio y que los mismos se mantengan estables durante todo el desarrollo). Además, en el modelo se denota la falta de comunicaciones entre usuario y desarrollador durante el proceso de concepción del software(Bass, Clements y Kazman 2002). Como los desarrollos de software no siguen un flujo secuecial; dado que en la práctica se realizan iteraciones en más de una etapa, se puede decir que el modelo de cascada no describe el proceso real de desarrollo de software. Como punto fuerte este modelo facilita la gestión de control del progreso del desarrollo del sistema, de las fechas de entrega y de los costos esperados(Fernandes y Machado 2016).

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 3, Mes Marzo, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 34-55
http://publicaciones.uci.cu

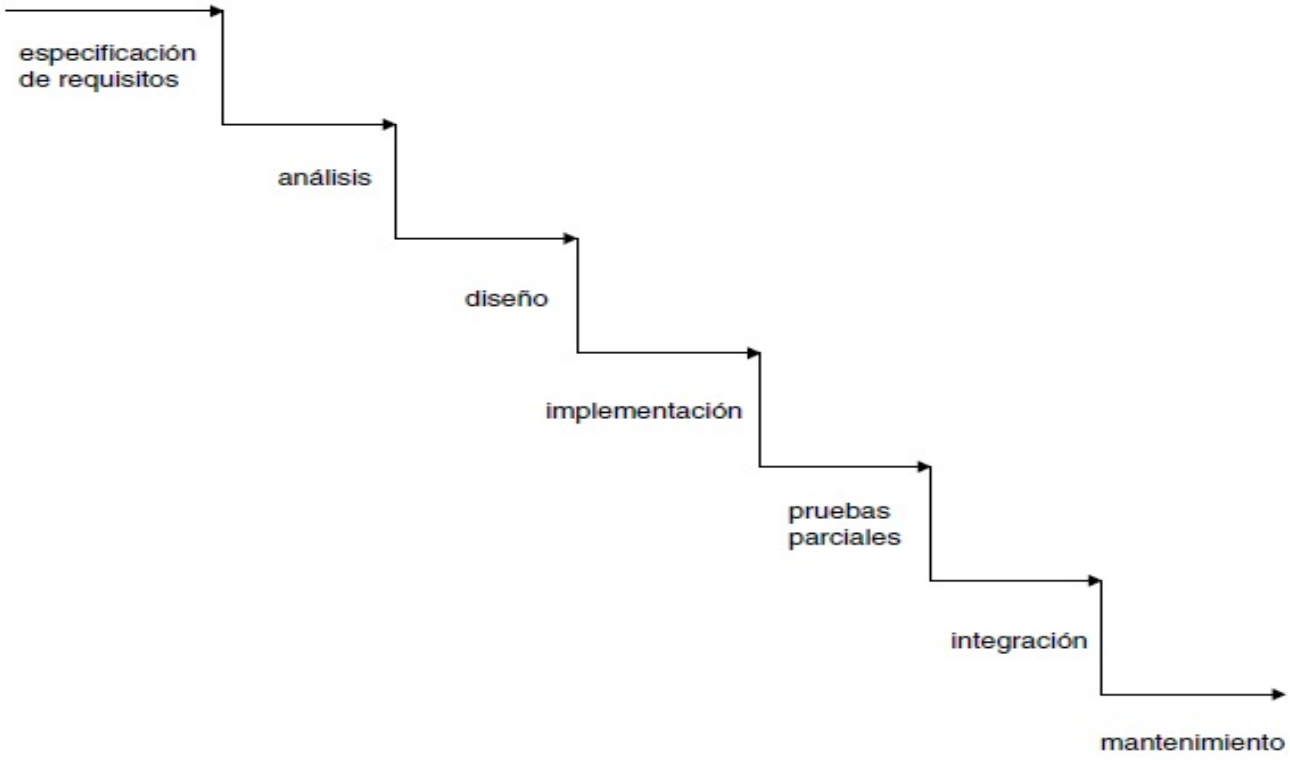

Figura 1. Secuencia de actividades para el modelo cascada(Ridel y Buemo 2007).

Este modelo presenta las siguientes ventajas y desventajas(Sitams 2012):

Ventajas

1- Planificación simple.
2- No necesita un personal altamente calificado.
3- Adecuado para proyectos en los que se cuenta con todos los requerimientos al comienzo, para desarrollar productos con funcionalidades conocidas, o para proyectos que se entiendan perfectamente desde el principio.
4- Fácil gestión del desarrollo del proyecto, de las fechas de entrega y de los costos esperados.
5- Genera una serie de documentos que pueden utilizarse para la validación y el mantenimiento del sistema.

Desventajas

1- No refleja realmente el proceso de desarrollo del software
2- Se tarda mucho tiempo en pasar por todo el ciclo

3- El mantenimiento se realiza en el código fuente
4- Las revisiones de proyectos de gran complejidad son muy difíciles
5- A menudo, durante el desarrollo, se pueden tomar decisiones que den lugar a diferentes alternativas. El modelo en cascada no reconoce esta situación.
6- Asume que los requisitos de un sistema pueden ser congelados antes de comenzar el diseño. Esto es como decir que el usuario rara vez evoluciona en cuanto a sus necesidades.
7- Cuando se entrega el sistema, éste obviamente no satisfará las expectativas actuales del cliente; en el mejor de los casos sólo cumplirá con las expectativas que tenía tiempo atrás, cuando se comenzó el desarrollo.

Modelo en espiral.

El modelo en espiral es un modelo de proceso de software evolutivo. Dentro de este modelo los requerimientos del usuario pueden cambiar en cualquier parte del desarrollo. El modelo presenta un enfoque evolutivo basado a la determinación del riesgo en el desarrollo del Software. Los ciclos dentro del modelo se organizan en forma de espiral, donde los ciclos interiores representan análisis temprano y construcción de prototipo y los ciclos exteriores representan el ciclo de vida clásico. Esta técnica se combina con análisis de riesgo durante cada ciclo(Kumar y Bhatia 2014).

La dimensión radial indica los costes de desarrollo acumulativos y la angular el progreso hecho en cumplimentar cada desarrollo en espiral. El análisis de riesgos, que busca identificar situaciones que pueden causar el fracaso o sobrepasar el presupuesto o plazo, aparecen durante cada ciclo de la espiral. En cada ciclo, el análisis del riesgo representa la misma cantidad de desplazamiento angular, mientras que el volumen desplazado barrido denota crecimiento de los niveles de esfuerzo requeridos para el análisis del riesgo como se ve en la Figura 2. El modelo se acerca paulatinamente a un sistema completo. Es adecuado para proyectos de gran escala.

El modelo espiral tiene cuatro actividades principales:

- Planificación (objetivos, alternativa, restricciones).
- Análisis de riesgo (análisis de alternativas y riesgos).
- Ingeniería (desarrollo de producto “próximo”).
- Evaluación del cliente

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 3, Mes Marzo, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 34-55
http://publicaciones.uci.cu

Algunos problemas son(Kaur 2017):

- Criticidad del análisis de riesgo.
- Difícil de "vender" como algo controlable.

Algunas ventajas son:

- Demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto y, si se aplica adecuadamente, debe reducir los riesgos antes que se conviertan en problemas.
- Permite la utilización de la creación de prototipos como un mecanismo de reducción del riesgo. Pero aún más importante, permite a quién lo desarrolla utilizar el enfoque de creación de prototipos en cualquier etapa de evolución del producto.

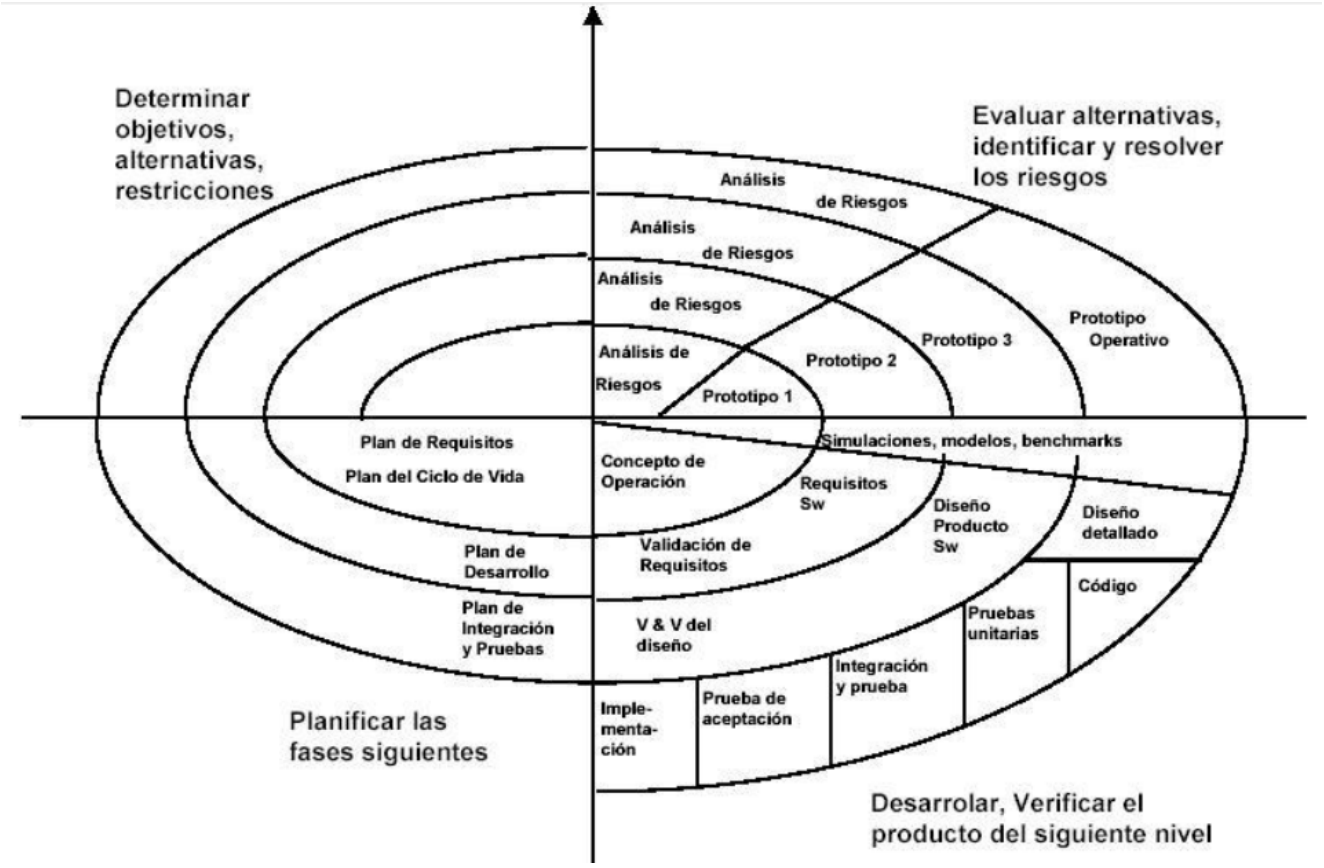

Figura 2. Modelo espiral(Kumar y Bhatia 2014).

Modelo de desarrollo incremental (iterativo)

Existen situaciones en las que los requerimientos iniciales del software están razonablemente bien definidos, pero el alcance general del esfuerzo de desarrollo imposibilita un proceso lineal. Además, tal vez haya una necesidad imperiosa de dar rápidamente cierta funcionalidad limitada de software a los usuarios y aumentarla en las entregas posteriores de software. En tales casos, se elige un modelo de proceso diseñado para producir el software en incrementos.

El modelo incremental combina elementos de los flujos de proceso lineal y paralelo. En relación con la Figura 3, el modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades. Cada secuencia lineal produce "incrementos" de software susceptibles de entregarse de manera parecida a los incrementos producidos en un flujo de proceso evolutivo. Resalta que en cualquier incremento este modelo se puede unir con el modelo de prototipos; al igual que guarda semejanza con el modelo iterativo. Se pudiera decir que el modelo incremental y prototipos no son más que subproductos del paradigma del modelo iterativo. Se puede arribar a esta conclusión partiendo de que cada incremento puede ser tomado como una iteración en sí. El modelo de proceso incremental se centra en que en cada incremento se entrega un producto que ya opera (Benediktsson y Dalcher 2003).

Dentro del modelo incremental existe una tendencia de crear en el primer incremento el producto base o fundamental; sobre el cual se realizará el desarrollo. Se puede decir que el primer incremento es el esqueleto del software. Esto permite que el usuario o cliente emplee ese producto base, lo evalué y en base a dicha evaluación guíe la futura iteración y en su defecto el desarrollo del software. Esto permite incluso modificar el producto de la primera iteración para orientarlo al cumplimiento de las necesidades específicas del cliente. El proceso se repite hasta culminar el producto final(Raval y Rathod 2013).

El empleo del desarrollo incremental resulta útil cuando no se dispone de los recursos humanos necesarios para hacer frente a un proyecto de software dentro del plazo establecido por el negocio. Se puede afirmar que si el producto entregado en el primer incremento es bien recibido por el cliente entonces se puede proceder a incorporar a más personal al proyecto. Esto posibilita un ahorro en costes por concepto de contratación inicial en el proyecto. (Pressman y Troya 1988).

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 3, Mes Marzo, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 34-55
http://publicaciones.uci.cu

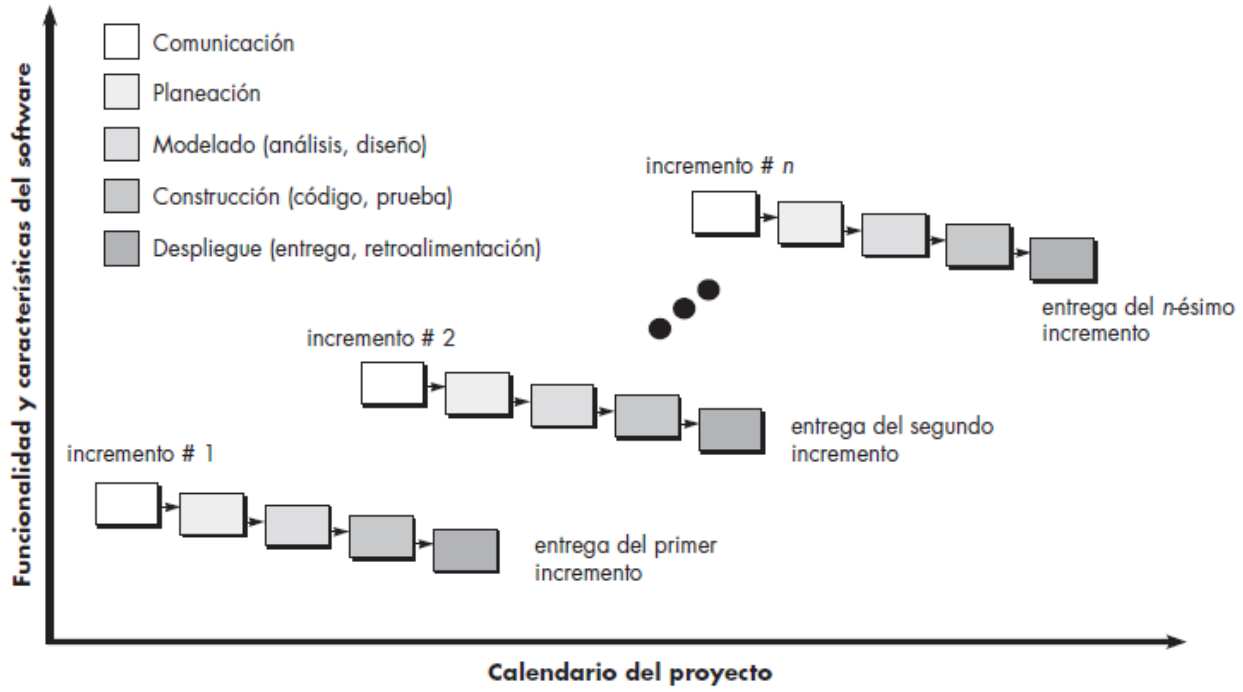

Figura 3. Representación del modelo de desarrollo incremental(Pressman y Troya 1988).

Las ventajas del modelo incremental son(Mujumdar, Masiwal y Chawan 2012):

- Divide el proyecto en partes más pequeñas.
- Crea un modelo de trabajo temprano y proporciona valiosa realimentación.
- La retroalimentación de una fase proporciona información de diseño para la siguiente fase. Muy útil cuando no hay mucho personal disponible.

Las desventajas del modelo incremental son(Mujumdar, Masiwal y Chawan 2012):

- La comunidad de usuarios debe participar activamente en el proyecto. Esto exige tiempo del personal y agrega el retraso del proyecto.
- Las habilidades de comunicación y coordinación son indispensables.
- Las solicitudes informales de mejora para cada fase pueden generar confusión.

Fábrica de software

Una fábrica de software es un área de desarrollo dedicada a producir componentes y procesos completos para ejecución de sistemas basados en especificaciones. Opera como una línea de ensamblado basada en los planos para armado(González 2013).

Una fábrica de software es una empresa de la industria del software cuya misión es el desarrollo de software para sus clientes de acuerdo a los requerimientos específicos que aquel le solicita; tiene como su principal fuente de ingreso la venta de proyectos de desarrollo de software. Generalmente, la propiedad intelectual de las aplicaciones informáticas desarrolladas le pertenece al cliente(Stojanovski y Dzekov 2012).

El modelo de fábrica de software se basa en que la empresa ofrezca servicios a la medida a sus clientes, es un concepto de subcontratación o tercerización, en el cual se delega el diseño de software a una empresa dedicada totalmente a ese fin, la cual es encargada de desarrollar plataformas para sistemas administrativos, nominas, control, procesos y mucho más(Robert Percy 2012).

De acuerdo con (Usaola 2013), una fábrica de software requiere una serie de buenas prácticas que organicen el trabajo de "una forma determinada, con una considerable especialización, así como una formalización y estandarización de los procesos". Entre estas buenas prácticas, los autores citan:

- La definición de la estrategia de fabricación, tanto en desarrollo tradicional, como en generación automática de código, como en reutilización.
- La implantación de un modelo de procesos y de un modelo de calidad de software.
- La certificación de las fábricas de software en algún nivel de madurez es, muchas veces, un requisito imprescindible para el desarrollo de proyectos con grandes empresas o con la Administración pública.
- La implantación de una metodología de fabricación.
- Integración continua y gestión de configuración.
- Control de calidad exhaustivo, periódico y automático.
- Diseño basado en el conocimiento, que evite la dependencia excesiva de determinadas personas.
- Construir el software tomando los casos de uso como pieza esencial del desarrollo.
- Utilizar ciclos de desarrollos evolutivos e iterativos, y no en cascada.

- Estimar los costes utilizando puntos función adaptados, para lo que hay varias propuestas en la literatura que dan buenos resultados.
- Guiar las acciones por su valor, evaluando el retorno de la inversión de cada una.

Las fábricas de software pueden ser implementadas siguiendo los siguientes modelos: Modelo basado en la norma ISO 9001 y CMM, modelo Eureka, modelo clasificatorio, modelo propuesto por Basili, modelo replicable, etc. Cada uno de los cuales presentan ventajas y desventajas y su elección dependerá de las condiciones específicas de la solución propuesta; así como se podrá tomar lo mejor de cada modelo para realizar un modelo propio.

Líneas de Producción de Software

La producción de software de calidad, en el tiempo adecuado y con unos costes razonables, continúa siendo un problema abierto de la ingeniería del software que ha sido abordado desde distintas aproximaciones. Una aproximación industrial para reducir los costes en la producción de software a gran escala consiste en aplicar el desarrollo de LPS. El desarrollo de software mediante LPS permite obtener los siguientes beneficios: mejora de la productividad, reducción del tiempo de desarrollo de nuevos productos, aumento de calidad de los productos desarrollados, disminución de los riesgos, incremento de la satisfacción del cliente, habilidad de adaptación a la personalización del producto, etc. Todos estos beneficios provienen del propio proceso de desarrollo de LPS. Se obtienen sustanciales mejoras de la productividad y se reducen los costes cuando se desarrollan los productos a partir de un conjunto de activos existentes de un modo predefinido, en vez de desarrollarlos separados desde cero o de un modo arbitrario(Blanes Domínguez 2016).

¿Qué es una Línea de Producción de Software (LPS)?. Según (Díaz y Trujillo 2010) una LPS se definen las líneas del producto de software como un conjunto de sistemas software, que comparten un conjunto común de características, las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base de una manera preestablecida.

Lo que diferencia la ingeniería de líneas de producto software de la ingeniería de software tradicional es la gestión de la variabilidad. El proceso de desarrollo de una línea de productos implica gestionar los puntos de variación entre los diferentes miembros de la línea. Con este fin, se identifica los aspectos comunes y los variables del dominio en

cuestión. Cabe destacar que las características variables y las comunes entre los productos son de igual importancia en el modelado de la línea de producto(Angélica Gabriela Salguero Espinosa y Fanny Paola Salguero Espinosa 2015).

Puntos fundamentales de una LPS(Eun-Jung Lee 2004) (Charles W. Krueger 2006):

- El objetivo de una LPS no es el desarrollo de un producto, sino el de un conjunto de productos.
- El conjunto de productos se define por medio de características.
- Se desarrolla orientándose a un segmento de mercado concreto.
- Los productos son desarrollados a partir de un conjunto común de activos reutilizables.
- Los productos se construyen de una forma preestablecida.

Ventajas:

- Incrementa significativamente la productividad de los ingenieros de software.
- Reduce el esfuerzo y el coste para desarrollar.
- Disminución de la tasa de defectos en los productos.
- La entrega de productos de software es más Rápida, Económica y de Mejor calidad.
- Produce mejoras en el tiempo de entrega del producto, menores costos de ingeniería.
- Crecimiento del portafolio de productos.

Desventaja:

- Reto constante en lograr un ajuste a las necesidades de cada cliente, sin que esta variabilidad aumente los costes.
- La implantación del paradigma LPS en una empresa es un proceso complejo.
- Requiere de un esfuerzo inicial importante.

Ecosistema de Software

Varios autores han expresado los elementos fundamentales que conforman los ecosistemas de software, como idea general varios coinciden en que los ecosistemas de software están orientados a crear alianzas entre diferentes organizaciones, enfoque a la externalización de las relaciones de las organizaciones, potenciando la productividad y compartir sectores de mercado(Sosa González et al. 2016) (Jansen, Brinkkemper y Cusumano 2013).

Tomando como referencia la definición efectuada por (Dr. Pedro Y y colectivo de autores 2013) donde define a un ecosistema de software como: "*un ecosistema formado a partir del sistema de relaciones entre entidades diferentes con el objetivo de compartir segmentos del mercado o aumentar la eficiencia y la eficacia en los procesos productivos que desarrollan. Está soportado por la definición de arquitecturas y plataformas comunes que facilitan la integración de soluciones y componentes, el intercambio de información, recursos, artefactos y activos en general. Estas relaciones deben estar soportadas por relaciones económico-financieras, convenios de trabajo o modelos de desarrollo que promuevan el intercambio libre de datos y códigos*". Se puede llegar a la conclusión que un ecosistema de software se define como: La interacción entre un grupo de actores que se encuentra sobre una plataforma tecnológica común que se torna en un número de soluciones de software o servicios. Cada actor es motivado por una serie de intereses o modelos de negocios conectados con el resto de los actores y con el ecosistema como un todo con relaciones simbióticas, mientras que la plataforma tecnológica es estructurada de una forma que permite la participación y la contribución de los diferentes actores.

Unas de las principales características de los ecosistemas de software identificados en algunos estudios afirman que los ecosistemas de software están vinculados a los ecosistemas naturales, las líneas de productos de software, y los ecosistemas de negocios. Estos campos son considerados por muchos autores como los orígenes de la investigación de los ecosistemas de software(Gabriel 2013).

Ventajas de un ecosistema de software (Sosa González et al. 2016):

- Fomento de la co-evolución.
- Aumento de la innovación dentro de la organización.
- Incremento del atractivo para nuevos actores.
- Disminución de los costos involucrados en el desarrollo y distribución de software.
- Mejora el análisis y comprensión de la arquitectura de software con el fin de decidir qué plataforma usar.
- Apoya la cooperación y el intercambio de conocimientos entre las múltiples e independientes entidades.
- Permite el análisis de los requisitos de comunicación entre las partes interesadas.
- Proporciona ayuda a las tareas de identificación de negocios, diseño de la arquitectura del producto, y la identificación de riesgos.

- Proporciona información para el administrador de la línea de productos en relación con las dependencias de software.
- Sostenibilidad tecnológica.

Desafíos de un ecosistema de software (Dr. Pedro Y y colectivo de autores 2013):

- El establecimiento de relaciones entre los actores del ecosistema y la propuesta de una representación adecuada de las personas y sus conocimientos en la modelización de ecosistemas.
- Desafíos arquitectónicos clave como: la estabilidad de la interfaz de la plataforma, la gestión de la evolución del software, la seguridad, la fiabilidad, la forma de apoyar la estrategia de negocio, arquitecturas adecuadas para apoyar el desarrollo del estilo de código abierto.
- La heterogeneidad de las licencias de software y la evolución de sistemas en un ecosistema. Las organizaciones deben manejar estos temas con el fin de disminuir los riesgos de dependencia.
- Obstáculos técnicos y socio-organizativos para la coordinación y la comunicación de los requisitos de los proyectos distribuidos geográficamente.
- Infraestructura y herramientas para fomentar la interacción social, la toma de decisiones y el desarrollo a través de las organizaciones que participan tanto de código abierto y de los ecosistemas propios.

Resultados y discusión

Los diferentes modelos de softwares analizados presentan distintos niveles de complejidad, tanto a lo que se refiere a su grado de implementación y control, como en lo relacionado a su comprensión por parte de directivos y trabajadores asociados a su puesta en práctica.

A primera vista resalta que los modelos tradicionales, entiéndase cascada, iterativo y espiral, son los más sencillos en cuanto a nivel de implementación en el momento de proponer su puesta en práctica; tanto por las demoras acarreadas para su comprensión como en los recursos necesarios (entiéndase recursos materiales como humanos), lo cual representa una ventaja contra los modelos actuales.

Por otro lado; dado el nivel alcanzado por las tecnologías actuales, la necesidad impetuosa de siempre aumentar los niveles de producción de software y un mercado cada vez más especializado, los modelos tradicionales se ven en desventaja contra las bondades que ofrecen los modelos actuales; desde las fábricas de software; hasta el actual

ecosistema de software. También resalta que las diferencias existentes entre los modelos de desarrollo de software actuales consisten en su mayoría en la forma de organización de trabajo, así como llama la atención que no es necesario tener implementado el último modelo de desarrollo de software para considerar a una empresa de la industria del software como apta para la producción y comercialización. Esto es dado; puesto que se sigue el principio que el desarrollo de software no es algo estático, sino dinámico, por lo tanto, los ambientes cambian; debiéndose de esta forma adoptar el modelo que más de acorde a las necesidades tanto de la empresa desarrolladora como del medio en el cual la misma tiene su radio de acción.

Observando desde otro punto de vista la problemática, se pudiera avizorar erróneamente que para lograr un último estado –en referencia al ecosistema de software- es necesario cursar por los diferentes estadios (entiéndase modelos anteriores al mismo). Este tipo de análisis estaría cargado de supuestos erróneos, puesto que en principio no es necesario haber implementado modelos anteriores para lograr llegar a modelos más avanzados dentro de una empresa o industria del software. Cada modelo puede ser implementado desde cero, sin tener en su concepción la semilla del otro.

Aunque es válido razonar que los modelos no son excluyentes y en su evolución los mismos van supliendo las carencias de sus antecesores; y convirtiendo al proceso de desarrollo de software en un proceso más refinado y complejo. Por ello es factible, aunque no necesario haber tenido experiencia en modelos anteriores para poder llevar a cabo la implementación de los nuevos modelos; si a experticia y experiencia se refiere. En la siguiente Tabla 1, se puede apreciar una comparación entre los diferentes modelos estudiados.

Para efectuar la comparación de los diferentes modelos de desarrollo de software se propone emplear las siguientes métricas:

- Especificación de los requerimientos (permite conocer en qué momento del desarrollo pueden ser especificados los requerimientos del producto)
- Entendimiento de los requerimientos (Brinda información relevante a la necesidad de comprender los requerimientos al inicio del proyecto. Está enfocada a ser una posible característica de los requerimientos iniciales del proyecto, no a ser una característica del paradigma en sí.)
- Flexibilidad (Indica la capacidad de adaptación)

- Implementación (Hace referencia al tiempo existente entre el inicio del proyecto y la implementación del producto final)
- Superposición de fases (Está ligada a la flexibilidad. Da una medida de la repetitividad del proceso)
- Costo (Costo necesario para la implementación del modelo, ya sea por su costo económico o esfuerzo)
- Complejidad (Indica el grado de complejidad o entendimiento para su puesta en práctica del modelo)
- Reutilización de componente (Indica la posibilidad de la reutilización de códigos, documentos, manuales, etc)
- Involucración del cliente (Indica la presencia del cliente a lo largo del desarrollo del proyecto)
- Análisis de Riesgo (Brinda información a la realización del análisis de los riesgos dentro del proyecto si/no o en qué momento)
- Garantía de logro efectivo (Da el grado de posibilidad de lograr un producto final terminado y de calidad)
- Puntos fuertes (indica las características más destacadas del modelo)
- Puntos Débiles (indica las características deficientes del modelo)
- Adecuado (Brinda información sobre hace que sector, dominio o tipo de solución está orientado el modelo)

Tabla 1. Comparación entre los modelos de desarrollo de software

	Cascada	Espiral	Iterativo	Fábrica de software	LPS	Ecosistema de software
Especificación de los Requerimientos	Al comienzo	Al comienzo	Al comienzo	Pueden ser al comienzo o pueden variar	Pueden ser al comienzo o pueden variar	Pueden ser al comienzo o pueden variar
Entendimiento de los requerimientos	Bien entendidos	Bien entendidos	No bien entendidos	Bien entendidos	Bien entendidos	Bien entendidos
Flexibilidad	Rígida	Flexible	Poco flexible	Flexible	Flexible dentro del dominio	Flexible
Implementación	larga	Depende del proyecto	No demora	Depende del proyecto	Depende del proyecto	Depende del proyecto
Superposición	No	Si	No	Si	Si	Si

de fases						
Costo	Bajo	Medio	Bajo	Alto	Alto	Alto
Complejidad	Simple	Compleja	Simple	Compleja	Compleja	Compleja
Reutilización de componentes	No	Si	Si	Si	Si	Si
Involucración del cliente	Al principio	Alto	Intermedio	Alto	Alto	Alto
Análisis de Riesgo	Solo al principio	Si	No	Si	Si	Si
Garantía de logro efectivo	Bajo	Alta	Alta	Alta	Alta	Alta
Puntos fuertes	Minimiza la planificación, fácil de usar y entender	Mejor productividad, más participación del usuario.	Mejor que la cascada, proporciona comentarios	Aumenta la productividad	Especialización en un dominio, aumenta la calidad del software	Co-ayuda. Aumento de la productividad
Puntos débiles	Requisitos rígidos, difícil de administrar los riesgos	No adecuado para proyectos pequeños, la gestión de riesgos es difícil.	Ninguna etapa está realmente terminada, es difícil de manejar.	Reutilización de código inferior a una LPS	No presenta variedad de productos en diferentes dominios	Difícil de implementar.
Adecuado	Personal técnicamente débil e inexperto	Para sistemas grandes y de misión crítica.	Cuando los requisitos del usuario no son claros.	Para sistemas grandes	Para sistemas grandes dentro de un dominio	Para sistemas grandes

Caso de estudio

Descripción de la empresa.

El caso de estudio es la empresa de desarrollo de software, la cual desarrolla, produce y comercializa aplicaciones informáticas, proyectos, equipamiento y asistencia técnica, producciones multimedia, audiovisuales y ediciones electrónicas, aplicando la ciencia y la innovación, e integrando soluciones de nuevas tecnologías de la información y las comunicaciones, con profesionalidad, competitividad y calidad avalada por la certificación de: Sistema integrado de gestión, con alcance para los sistemas de calidad, medio ambiente y seguridad y salud del trabajo.

Por otro lado, se observa que la empresa posee un alto grado de especialización en el dominio de los negocios empresariales; demostrables a través de sus productos especializados en el dominio de soluciones financieras y contables. En cuanto a los recursos humanos, la empresa dispone de un personal altamente capacitado, conocedor de las normas, procedimientos y el saber-hacer específico dentro del dominio de los productos de software empresariales; presenta una baja fluctuación de su personal; así como el mismo dispone de años de experiencia laborando dentro del dominio de los productos informáticos para el sector empresarial.

Productos y servicios.

En la actualidad la empresa cuenta con los siguientes productos y servicios:

- Crédito Hipotecario
- Microcrédito
- Plan Ahorro Cuenta Propia
- Depósito a Plazo Fijo
- Acreditación de Sueldos
- Anticipo de Sueldos
- Crédito Ordinario
- Apertura de Cuentas
- Tarjetas Tele banca
- Ayuda Mutua
- Transferencia de Efectivo
- Red de Pagos por Tele banca

Haciendo referencia a los activos; se puede afirmar que la empresa dispone de un repositorio propio de códigos, manuales, y toda una gama de activos que pueden ser empleados para la creación de una línea de producción. Por otro lado, se desea efectuar la elección de un modelo de desarrollo de software para la creación de software especializado en el dominio de la administración financiera que permita:

- Mejorar el desempeño, con un estímulo de eficiencia y mejor gestión de la empresa.
- Facilitar el trabajo de creación y documentación de los productos de software.
- Tener como referencia una metodología ágil para la creación de productos software.
- Mejorar la administración de los recursos humanos (Tiempo).
- Obtener mayor porcentaje de proyectos concluidos satisfactoriamente.

Para efectuar la elección del modelo de desarrollo de software a implementar, se seguirán los criterios de calidad de software expuestos en la norma (ISO 25010 2011):

- Funcionabilidad
- Fiabilidad
- Usabilidad
- Mantenibilidad
- Seguridad
- Portabilidad

Por todo lo expuesto anteriormente y tomando como base las características de los modelos estudiados, se procede a la elección de crear una Línea de Producto Software para solucionar la problemática. En este caso, una Línea de Producto Software permitirá aumentar la cartera de productos de la empresa; aprovechar la disponibilidad de recursos humanos estables y altamente capacitados existentes en la entidad (lo que permite establecer sistemas de trabajo complejos);reducción del coste y tiempos de producción; aumento de la calidad del software producido tomando como referencia la disminución de la tasa de errores detectados; aumentar la competitividad en el mercado de soluciones software financieras y contables; así como orientará a la empresa hacia soluciones de proyectos cada vez más complejos. El desarrollo del modelamiento del dominio de la Línea de Producto Software para el dominio de sistemas financiero quedaría como se muestra en la Figura 4.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 3, Mes Marzo, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 34-55
http://publicaciones.uci.cu

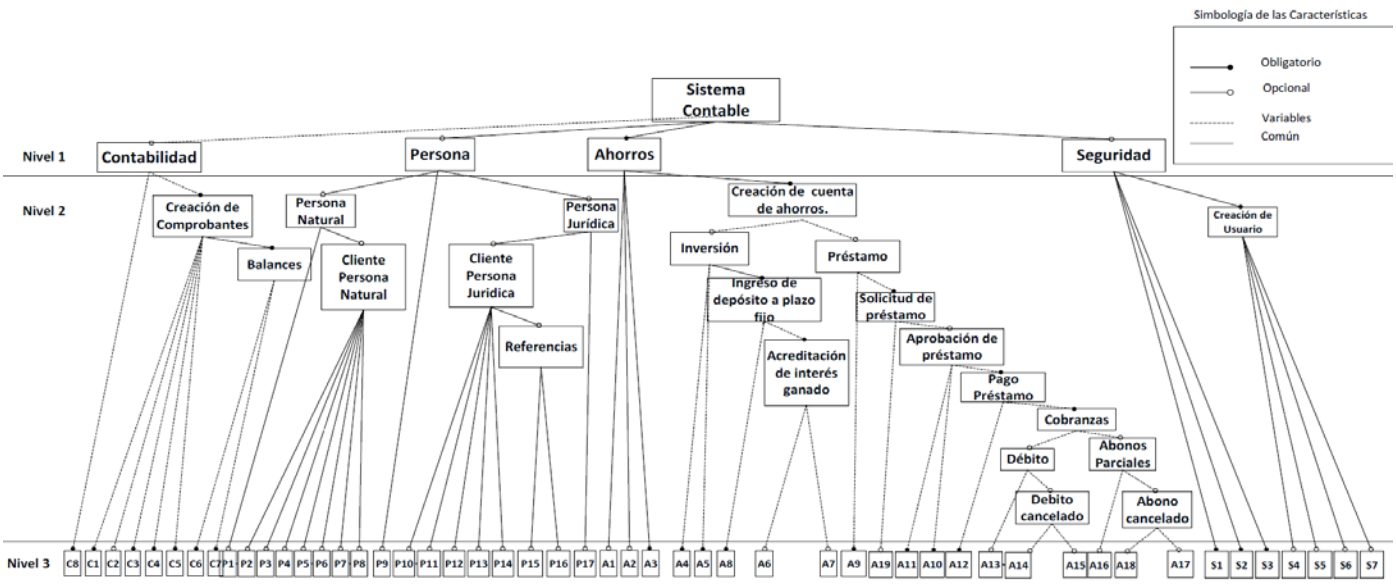

Figura 4. Modelamiento del dominio de la Linea Producto Software de un Sistema Financiero

Conclusiones

Analizando los elementos expuestos en este artículo; se aprecia que se cumplen los objetivos trazados. En el análisis efectuado a los diferentes modelos de desarrollo de software, resalta la siguiente conclusión: Ningún modelo de desarrollo de software es mejor a otro. La elección a efectuar con respecto a qué modelo emplear siempre dependerá del problema de resolver, así como su grado de complejidad. Cada proyecto de software requiere de una forma de particular de abordar el problema. Las propuestas comerciales y académicas actuales promueven procesos iterativos, donde en cada iteración puede utilizarse uno u otro modelo de proceso, considerando un conjunto de criterios (Por ejemplo: grado de definición de requisitos, tamaño del proyecto, riesgos identificados, entre otros).

En el caso particular de la empresa abordada es válido pensar en implementar una línea de producción de software; dado que sus productos de software son altamente especializados en un solo dominio (productos empresariales financieros); así como se dispone de un alto nivel de preparación de los recursos humanos relacionados con la investigación y producción de software.

Agradecimientos

Dc. Maibys Sierra Lorenso

Referencias

ANGÉLICA GABRIELA SALGUERO ESPINOSA y FANNY PAOLA SALGUERO ESPINOSA, 2015. Diseño de una LPS (Línea de Productos de Software) para optimizar los productos intermedios y finales del proceso de desarrollo de software en la empresa CLOUDSTUDIO Servicios de Tecnología Informática Cía. Ltda. de la ciudad de Quito. [en línea]. Latacunga, Ecuador: Universidad de las Fuerzas Armadas ESPE. [Consulta: 2 febrero 2018]. Disponible en: http://repositorio.espe.edu.ec/handle/21000/10118.

BASS, L., CLEMENTS, P. y KAZMAN, R., 2002. Software architecture in practice [en línea]. 9. printing. Boston: Addison-Wesley. SEI series in software engineering. ISBN 978-0-201-19930-7. Disponible en: http://www.idt.mdh.se/kurser/ISD_cdt417/files/articles/paper5.pdf.

BENEDIKTSSON, O. y DALCHER, D., 2003. Effort estimation in incremental software development. IEE Proceedings-Software, vol. 150, no. 6, pp. 351–357.

BLANES DOMÍNGUEZ, D., 2016. Una Aproximación de Ingeniería de Requisitos para Líneas de Productos Software Basada en una Estrategia de Desarrollo Dirigido por Modelos. Doctoral. Valencia: Universitat Politècnica de València. AP2009-4635

CHARLES W. KRUEGER, 2006. Methods & Tools. Practical knowledge for the software develo per, tester and project manager [en línea]. S.l.: s.n. [Consulta: 5 febrero 2018]. 3. ISBN 1661-402X. Disponible en: http://www.methodsandtools.com/PDF/mt200603.pdf.

DÍAZ, Ó. y TRUJILLO, S., 2010. Líneas de producto software. Publicado en "Fábricas de Software: experiencias, tecnologías y organización" 2º edición, M. GPiattiniJGarzás (editores) Editorial Ra-Ma2010,

DR. PEDRO Y, P.P. y COLECTIVO DE AUTORES, 2013. Gespro.Paquete para la gestión de proyectos. [en línea]. 2013. S.l.: Registro Centro Nacional de Registro de Derecho de Autor de Cuba. [Consulta: 30 enero 2018]. Disponible en: https://www.researchgate.net/publication/260418890.

EUN-JUNG LEE, 2004. A Feature-Based Technique for Product Line Software Component Implementation [en línea]. Máster. Pohang, Korea: Pohang University of Science and Technology. [Consulta: 5 febrero 2018]. Disponible en: http://postech.dcollection.net/common/orgView/000001907420.

FERNANDES, J.M. y MACHADO, R.J., 2016. Software Engineering. En: J.M. FERNANDES y R.J. MACHADO, Requirements in Engineering Projects [en línea]. Cham: Springer International Publishing, pp. 15-44. [Consulta: 6 febrero 2018]. ISBN 978-3-319-18596-5. Disponible en: http://link.springer.com/10.1007/978-3-319-18597-2_2.

GABRIEL, O.M., 2013. Análisis de grandes ecosistemas de software abierto. Máster. Madrid: Universidad Politécnica de Madrid.

GONZÁLEZ, D., 2013. Fábrica de Software [en línea]. mayo 2013. S.l.: s.n. [Consulta: 30 enero 2018]. Disponible en: www.northware.mx/wp-content/uploads/2013/06/Art%C3%ADculoMayo.pdf?x69168.

ISO 25010, 2011. ISO/IEC 25010:2011(E): Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models [en línea]. 2011. S.l.: ISO INTERNATIONAL STANDARD. Disponible en: https://webstore.iec.ch/preview/info_isoiec25010%7Bed1.0%7Den.pdf.

JANSEN, S., BRINKKEMPER, S. y CUSUMANO, M.A., 2013. Software ecosystems: Analyzing and managing business networks in the software industry [en línea]. Cheltenham: Edward Elgar Publishing. ISBN 978-1-78195-562-8. Disponible en: http://ebookcentral.proquest.com/lib/subhh/detail.action?docID=1190645.

KAUR, S., 2017. Comparative analysis of software development models. , pp. 7. ISSN 2277 - 9655. DOI 10.5281/zenodo.1036644.

KUMAR, G. y BHATIA, P.K., 2014. Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies. [en línea]. S.l.: IEEE, pp. 189-196. [Consulta: 6 febrero 2018]. ISBN 978-1-4799-4910-6. DOI 10.1109/ACCT.2014.73. Disponible en: http://ieeexplore.ieee.org/document/6783451/.

MUJUMDAR, A., MASIWAL, G. y CHAWAN, P.M., 2012. Analysis of various software process models. International Journal of Engineering Research and Applications, vol. 2, no. 3, pp. 2015–2021.

MUNASSAR, N.M.A. y GOVARDHAN, A., 2010. A comparison between five models of software engineering. IJCSI, vol. 5, pp. 95–101.

PRESSMAN, R.S. y TROYA, J.M., 1988. Ingeniería del software. ,

RAVAL, R.R. y RATHOD, H.M., 2013. Comparative Study of Various Process Model in Software Development. International Journal of Computer Applications, vol. 82, no. 18.

RIDEL, A.W. y BUEMO, S.G., 2007. Ingeniería de software: el proceso para el desarrollo de software. ,

ROBERT PERCY, O.A., 2012. Metodología de desarrollo y mantenimiento de software para una fábrica de software [en línea]. Tesis de Diploma. Perú: Universidad Nacional de Ingeniería. [Consulta: 2 febrero 2018]. Disponible en: http://cybertesis.uni.edu.pe/handle/uni/3323.

SCACCHI, W., 2001. Process models in software engineering. Encyclopedia of software engineering [en línea], [Consulta: 6 febrero 2018]. DOI 10.1002/0471028959.sof250. Disponible en: http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf.

SITAMS, C., 2012. Explore 10 Different Types of Software Development Process Models. , ISSN 0975-9646.

SOSA GONZÁLEZ, R., PÉREZ PUPO, I., GARCÍA, R., PEÑA HERRERA, E. y PIÑERO PÉREZ, P.Y., 2016. Ecosistema de Software GESPRO-16.05 para la Gestión de Proyectos. Revista Cubana de Ciencias Informáticas, vol. 10, pp. 239–251.

STOJANOVSKI, T. y DZEKOV, T., 2012. Rapid Application Development Using Software Factories. arXiv, pp. 13.

USAOLA, M.P., 2013. Desarrollo de software basado en reutilización [en línea]. Catalunya: Universitat Oberta de Catalunya. [Consulta: 30 enero 2018]. Disponible en: https://www.exabyteinformatica.com/uoc/Informatica/Tecnicas_avanzadas_de_ingenieria_de_software/Tecnicas_avanzadas_de_ingenieria_de_software_(Modulo_2).pdf.