Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

Tipo de artículo : Artículo original
Temática : Ingeniería y gestión de software
Recibido : 01/10/2020 | Aceptado : 02/12/2020

Modelos de Desarrollo de Software

Software Development Models

Lisdania de la Caridad Delgado Olivera [1*] https://orcid.org/0000-0001-9593-2772
Lexys Manuel Díaz Alonso [2] https://orcid.org/0000-0001-7110-4745

[1] Facultad 1. Departamento de Inteligencia Computacional. Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, km 2 ½, Torrens, Boyeros, CP. 19370 La Habana, Cuba.
[2] Centro de Software Libre. Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, km 2 ½, Torrens, Boyeros, CP. 19370 La Habana, Cuba.

*Autor para la correspondencia. (lcdelgado@uci.cu)

RESUMEN

El desarrollo de software ha sido históricamente cuestionado debido a problemas asociados a sus resultados finales. Estos inconvenientes dieron origen al concepto de "crisis del software" en 1968, que prácticamente surgió conjuntamente con la creación del software. Una parte importante de la ingeniería de software es el desarrollo de metodologías y modelos para lograr sistemas más eficientes y de mayor calidad, con la documentación necesaria en perfecto orden y en el tiempo requerido. Existen diferentes modelos y metodologías que han sido utilizados en los últimos años como herramientas de apoyo para el desarrollo del software. La interrogante principal está en conocer cuál modelo utilizar en el prdeoceso de desarrollo de

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

software de un proyecto. Para darle solución a este problema se define como objetivo de la investigación proponer alternativas viables a la hora de seleccionar un modelo para el desarrollo de software teniendo en cuenta las características del proyecto.

Palabras clave: modelo; proceso de desarrollo de software; proyecto.

ABSTRACT

Software development has been historically questioned due to problems associated with its final results. These drawbacks gave rise to the concept of "software crisis" in 1968, which practically arose in conjunction with the creation of software. An important part of software engineering is the development of methodologies and models to achieve more efficient and higher quality systems, with the necessary documentation in perfect order and in the time required. There are different models and methodologies that have been used in recent years as support tools for software development. The main question is to know which model to use in the software development process of a project. To give solution to this problem is defined as research objective, propose viable alternatives when selecting a model for software development taking into account the characteristics of the project

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

Introducción

El desarrollo de software ha sido históricamente cuestionado debido a problemas asociados a sus resultados finales, entre los que se encuentran que: los sistemas no responden a las expectativas de los usuarios, los programas fallan con cierta frecuencia, los costos del software son difíciles de prever y normalmente superan las estimaciones, la modificación del software es una tarea difícil y costosa, entre muchas otras

como que: el software se suele presentar fuera del plazo establecido y con menos prestaciones de las consideradas inicialmente. Solo una pequeña parte de los proyectos de software desarrollados terminan dentro de plazos y costos, y cumplen los requerimientos acordados (Pressman, 1993).

Estos inconvenientes dieron origen al concepto de "crisis del software" que prácticamente surgió conjuntamente con la creación del software. Es un término informático acuñado en 1968, en la primera conferencia organizada por la OTAN (Organización del Tratado del Atlántico Norte) sobre desarrollo de software, y residía en la complejidad de construir sistemas de software, y también en los cambios constantes a los que debía someterse el software para adaptarse a las necesidades cambiantes de los usuarios y a las innovaciones tecnológicas (Naul-Rander, y otros, 1968).

En esta conferencia se pretendía acordar las bases para una ingeniería de construcción de software. Según Fritz Bauer se necesitaba *"establecer y usar principios de ingeniería orientados a obtener software de manera económica, que sea fiable y funcione eficientemente sobre máquinas reales"* (Peter, y otros, 1968). Sin embargo, dicho planteamiento además debía incluir otros aspectos, tales como: la mejora de la calidad del software, satisfacción del cliente, mediciones y métricas. A partir de aquí empieza a usarse el término Ingeniería del Software, para expresar el área de conocimiento que se estaba desarrollando en torno a las problemáticas que ofrecía el software (Pons-Giandini, y otros, 2010).

Una parte importante de la ingeniería de software es el desarrollo de metodologías y modelos. En la actualidad ha habido muchos esfuerzos que se han encaminado al estudio de los métodos y técnicas para lograr una mejor aplicación de las metodologías y lograr sistemas más eficientes y de mayor calidad con la documentación necesaria en perfecto orden y en el tiempo requerido (Rumbaugh-Jacobson, y otros, 2000). La experiencia ha demostrado que los proyectos exitosos son aquellos que son administrados siguiendo una serie de procesos que permiten organizar y luego controlar el proyecto, considerando válido destacar que aquellos procesos que no sigan estos lineamientos corren un alto riesgo de fracasar (Pons-Giandini, y otros, 2010).

Existen diferentes modelos y metodologías que han sido utilizados en los últimos años como herramientas de apoyo para el desarrollo del software. Teniendo en cuanta los elementos anteriores surge como interrogante ¿cuál modelo utilizar en el proceso de desarrollo de software de un proyecto? Seguidamente se hace referencia a varios autores que analizan el tema, todos con el objetivo de proponer alternativas viables

a la hora de seleccionar un modelo para el desarrollo de software teniendo en cuenta las características del proyecto.

Métodos o Metodología Computacional

Sommerville define modelo de proceso de software como "*Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto, un modelo de procesos del software es una abstracción de un proceso real* (Sommerville, 2005).

Los modelos genéricos no son descripciones definitivas de procesos de software; son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software (Pons-Giandini, y otros, 2010). Algunos de los modelos más conocidos son:

1. Prototipo.
2. Desarrollo basado en componentes (reutilización).
3. Desarrollo en espiral.
4. Modelo RAD (Rapid Application Development).
5. Modelo en cascada.

Prototipo

Su objetivo es entender los requisitos del usuario y trabajar para mejorar la calidad de los mismos. Este modelo inicia con la recolección de requerimientos del cliente, con base en estos se define el conjunto de objetivos para el software, se identifican los requisitos conocidos y con base en estos se desarrolla rápidamente un prototipo o maqueta que posteriormente evalúa el cliente utilizándolo y ayudando a refinar de nuevo los requisitos del software a desarrollar; este proceso se seguirá repitiendo hasta que el cliente quede satisfecho con el desarrollo del software (Salazar-Aguirre, y otros, 2011). La Fig.1 muestra cómo se realiza el modelo de construcción de prototipos, iniciando en el momento en que los analistas del sistema.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

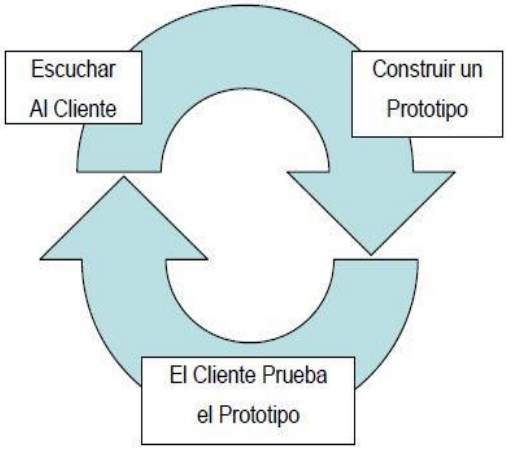

Fig. 1– Modelo Prototipo.

Ventajas

1. Genera una buena comunicación con los clientes.
2. Recomendado para proyectos de pequeño y mediano alcance.

Desventajas

1. Las planificaciones pudieran sufrir cambios.
2. Dificultades para mantener el sistema en proyectos grandes y en la respuesta a cambios en los requisitos del usuario.
3. Períodos de desarrollo largos.

Desarrollo basado en componentes (reutilización)

1. Los compromisos en los requisitos son inevitables, por lo cual puede que el software no cumpla las expectativas del cliente.
2. Las actualizaciones de los componentes adquiridos no están en manos de los desarrolladores del sistema.
3. Alto costo, si la adaptación de nuevos paquetes es costosa.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

4. Dificultades para asegurar el versionamiento si los suministradores de componentes no mejoran continuamente sus productos.

Desarrollo en espiral

El modelo de desarrollo en espiral es actualmente uno de los más conocidos y fue propuesto por Barry Boehm en 1986. El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Es decir, surge de un sistema inicial que se desarrolla rápidamente a partir de especificaciones abstractas. En la Fig.2 se muestran las fases que propone el modelo Desarrollo en espiral.

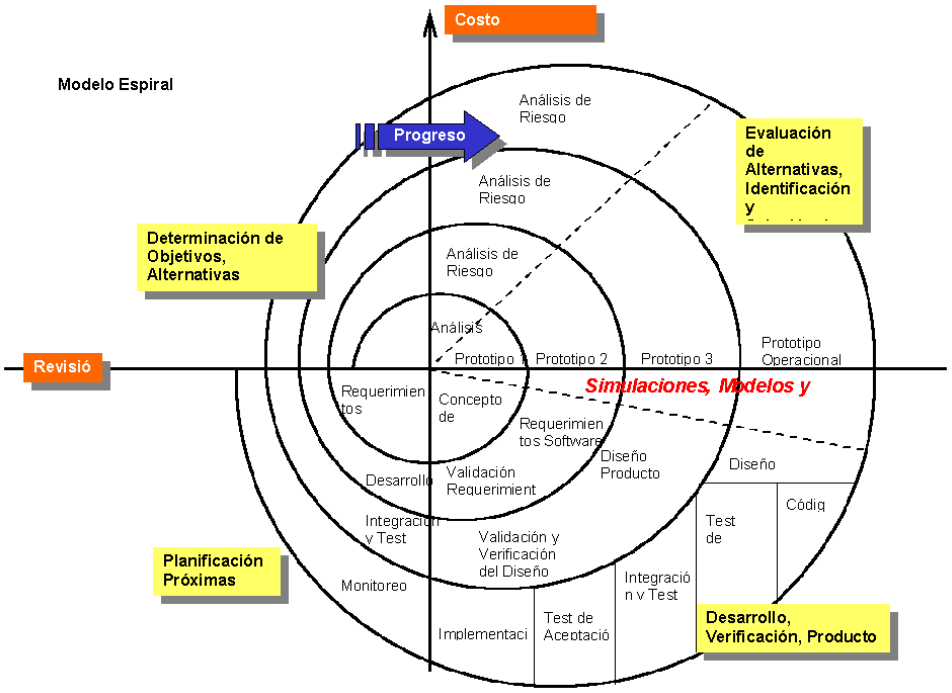

Fig. 2 – Modelo de desarrollo en espiral.

Cada ciclo de desarrollo se divide en cuatro fases:

1. **Definición de objetivos**: Se definen los objetivos. Se definen las restricciones del proceso y del

producto. Se realiza un diseño detallado del plan administrativo. Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.

2. **Evaluación y reducción de riesgos**: Se realiza un análisis detallado de cada riesgo identificado. Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos. Se llevan a cabo los pasos para reducir los riesgos.
3. **Desarrollo y validación**: Se escoge el modelo de desarrollo después de la evaluación del riesgo. El modelo que se utilizará depende del riesgo identificado para esa fase.
4. **Planificación**: Se determina si continuar con otro ciclo. Se planea la siguiente fase del proyecto (Boehm, 1988).

Este modelo, a diferencia de los otros, toma en consideración explícitamente el riesgo, esta es una actividad importante en la administración del proyecto.

Variaciones del desarrollo en espiral

1. **Desarrollo incremental**: Se va creando el sistema añadiendo pequeñas funcionalidades. Cada uno de los pequeños incrementos es parecido a lo que ocurre dentro de la fase de mantenimiento. Bajo este modelo se entrega software por partes funcionales más pequeñas, pero reutilizables, llamadas incrementos. Cada incremento se construye sobre aquel que ya fue entregado.
2. **Desarrollo iterativo**: El sistema se desarrolla como una secuencia de pasos e iteraciones una vez establecida la arquitectura global. Los usuarios pueden experimentar con los productos resultantes de cada iteración, y usualmente el equipo de desarrollo puede continuar con el trabajo mientras que los usuarios experimentan con el sistema. Cada iteración refina lo realizado en la iteración anterior (Boehm, 1988).

Ventajas

1. Buena comunicación con los clientes a partir de que combina las ventajas de prototipo y cascada.
2. Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo

desde el primer incremento.

3. Recomendado para proyectos de gran alcance.
4. Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada incremento.
5. Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

Desventajas

1. Cada incremento debe ser pequeño para limitar el riesgo y debe aumentar la funcionalidad.
2. Es difícil establecer las correspondencias de los requisitos contra los incrementos y detectar las unidades o servicios genéricos para todo el sistema.
3. Dificultades para mantener las planificaciones, y concebir y graficar el sistema globalmente.

Modelo RAD

Desarrollo Rápido de Aplicaciones (por sus siglas en inglés) es un modelo de proceso de desarrollo de software relativamente corto (dura entre 60 y 90 días). Se utiliza la construcción de software basada en componentes, utilizando herramientas de software que permitan de forma ágil y efectiva realizar una aplicación con altos estándares de calidad. La Fig.3 muestra las etapas del modelo RAD. El Modelo RAD comprende las siguientes etapas:

1. **Modelado de gestión**: Se genera la información que conduce el proceso de gestión, se identifica a dónde va la información y quién la procesa.
2. **Modelado de datos**: Se definen los almacenes de datos y cómo se relacionan los almacenes entre sí.
3. **Modelado del proceso**: Se utiliza para añadir, modificar, suprimir o recuperar un objeto de datos.
4. **Generación de aplicaciones**: Se utiliza una herramienta de cuarta generación que permite crear el software y facilitar la construcción del programa.
5. **Pruebas y entrega**: El proceso de desarrollo finaliza realizando pruebas de calidad del software diseñado con la herramienta RAD, posteriormente se realiza la implementación de la aplicación (Salazar-Aguirre, 2011).

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

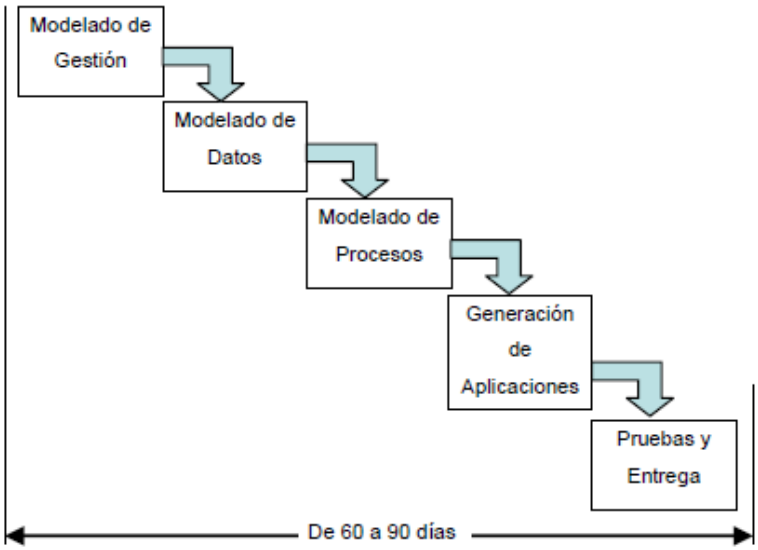

Fig. 3 – Modelo de Desarrollo Rápido de Aplicaciones.

Ventajas

1. Los usuarios pueden revisar el sistema sistemáticamente.
2. Los entregables pueden ser fácilmente trasladados a otra plataforma.
3. El desarrollo se realiza a un nivel de abstracción mayor.
4. Visibilidad temprana y mayor flexibilidad.
5. Menor codificación manual y mayor involucramiento de los usuarios.
6. Posiblemente menos fallas y menor costo
7. Ciclos de desarrollo más pequeños.

Desventajas

1. Para proyectos grandes, aunque por escalas, requiere recursos humanos suficientes como para crear el número correcto de equipos.
2. Requiere clientes y desarrolladores comprometidos en las rápidas actividades. Si no hay compromiso los proyectos fracasaran.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

3. Costo de herramientas integradas y equipo necesario.
4. Progreso más difícil de medir.
5. Menos eficiente y menor precisión científica.
6. Dependencia en componentes de terceros: funcionalidad de más o de menos, problemas legales.

Modelo en cascada

Este modelo toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso. La interacción entre fases puede observarse en la Fig.4.

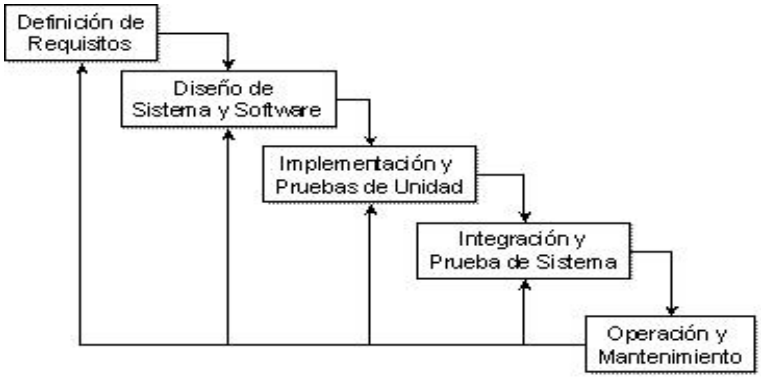

Fig. 4 – Modelo de desarrollo en cascada.

El modelo en cascada consta de las siguientes fases:

1. **Definición de los requisitos**: Los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.
2. **Diseño de software**: Se particiona en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
3. **Implementación y pruebas unitarias:** Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.
4. **Integración y pruebas del sistema**: Se integran todas las unidades. Se prueban en conjunto. Se

entrega el conjunto probado al cliente.

5. **Operación y mantenimiento**: Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

Cada fase tiene como resultado documentos que deben ser aprobados por el usuario. Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas (Royce, 1988).

Ventajas

1. Permite estimar calendarios y presupuestos con mayor precisión.
2. Facilita un nivel de satisfacción del cliente más elevado que otros enfoques.
3. Es fácil de manejar los planes de proyectos.
4. Alto nivel de seguridad y confiabilidad.

Desventajas

1. Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
2. Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
3. Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
4. Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
5. Es inflexible a la hora de evolucionar para incorporar nuevos requisitos. Es difícil responder a cambios en los requisitos.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

Resultados y discusión

¿Qué modelo de proceso de desarrollo de software utilizar?

Cada proyecto de software requiere una forma 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. En la Tabla 1 se expone un cuadro comparativo de acuerdo con algunos criterios básicos para la selección de un modelo de proceso, la medida utilizada indica el nivel de efectividad del modelo de proceso de acuerdo al criterio (Laboratorio Ing. Soft, 2002).

Tabla 1 - Comparación entre los modelos de desarrollo teniendo en cuenta sus características.

Modelo de proceso	Prototipo	Desarrollo basado en componentes	Desarrollo en espiral	Modelo RAD	Modelo en cascada
Funciona con requisitos y arquitectura no predefinidos	Alto	Medio	Alto	Alto	Bajo
Produce software altamente fiable	Medio	Bajo a Alto	Alto	Medio	Alto
Gestión de riesgos	Medio	Bajo a Alto	Alto	Bajo	Bajo
Permite correcciones sobre la marcha	Alto	Alto	Medio	Alto	Bajo
Visión del progreso por el Cliente y el Jefe del proyecto	Alto	Alto	Medio	Alto	Bajo

Aunque estos criterios son válidos para seleccionar un modelo de desarrollo de software, es muy importante tener en cuenta las características del software y del equipo de proyecto. En la Tabla 2 se muestra una comparación teniendo en cuenta criterios más representativos que servirán de referencia en la toma de decisión para adoptar un modelo de desarrollo (Saurith, y otros, 2010).

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 1, Enero-Marzo, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 37-51

Tabla 2 - Comparación entre modelos de desarrollo teniendo en cuenta características del software y del equipo de proyecto.

Modelo de proceso	Prototipo	Desarrollo basado en componentes	Desarrollo en espiral	Modelo RAD	Modelo en cascada
Disponibilidad de recursos	Algunos	Todos	Algunos	Todos	Todos
Complejidad del software	Media	Alto	Alta	Baja	Baja
Entendimiento de los requisitos	Bajo	Medio	Bajo	Específico	Específico
Conocimiento del dominio del problema	Pobre	Alto	Pobre	Alto	Alto
Tiempos de desarrollo	Alto	Bajo	Bajo	Bajo	Medio
Costos de los proyectos	Medio	Alto	Alto	Medio	Medio
Calidad del software	Medio	Medio	Alto	Medio	Medio
Documentación	Bajo	Bajo	Alto	Bajo	Bajo

Conclusiones

El estudio de los modelos para el proceso de desarrollo del software contribuyó a determinar que los modelos son actividades que están relacionadas con la especificación del software (el análisis y diseño), el desarrollo (codificación), la elaboración de pruebas que evidencien la calidad del software y la implementación del producto en su entorno real.

La comparación entre los modelos de desarrollo de software permitió definir un nivel de efectividad teniendo en cuenta algunos criterios de selección. Esta comparación será muy útil cuando se desee seleccionar un modelo, considerando además las características del software y del equipo de proyecto.

Referencias

Pressman, Roger. Ingeniería de Software. Un enfoque práctico. sd: Editorial Mc Graw Hill. 1993.

Peter, Naur; Randell, Brian (ed.). Software Engineering: Report on a conference sponsored by the Nato Science Committee, Garmisch, Germany, 7th to 11th October 1968.

Pons, Claudia; Giandini, Roxana; Pérez, Gabriela. Desarrollo de Software Dirigido por Modelos. Teorías, Metodologías y Herramientas. McGraw-Hill Education, 2010, p. 978-950.

Rumbaugh, James; Jacobson, Ivar; Booch, Grady. El proceso unificado de desarrollo de software. Addison-Wesley. Madrid, España, 2000.

Sommerville, Ian. Ingeniería del software. Pearson Educación, 2005.

Salazar, O., Aguirre, F., Osorio, J., Herramientas para el desarrollo rápido de aplicaciones web. Scientia et technica, 2011, p. 254–258

Boehm, Barry W., A spiral model of software development and enhancement. Computer, 1988, vol. 21, no 5, p. 61-72.

Royce, Winston W. Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering. IEEE Computer Society Press, 1987. p. 328-338.

Laboratorio Ing. Soft., Ingeniería de software 2, Departamento de Informática, 2002

Saurith, A., Estay-Niculcar, C., Análisis y Diseño Integral de Sistemas y Requerimientos. Fundación Universitaria Iberoamericana. Barcelona, España, 2010. 167 pp.

Conflicto de interés

Los autores autorizan la distribución y uso de su artículo.

Contribuciones de los autores

1. Conceptualización: Lisdania de la Caridad Delgado Olivera

2. Curación de datos: Lexys Manuel Díaz Alonso
3. Análisis formal: Lisdania de la Caridad Delgado Olivera
4. Adquisición de fondos: -
5. Investigación: Lisdania de la Caridad Delgado Olivera, Lexys Manuel Díaz Alonso
6. Metodología: Lisdania de la Caridad Delgado Olivera
7. Administración del proyecto: Lisdania de la Caridad Delgado Olivera
8. Recursos: Lexys Manuel Díaz Alonso
9. Software: -
10. Supervisión: Lisdania de la Caridad Delgado Olivera
11. Validación: Lexys Manuel Díaz Alonso
12. Visualización: Lisdania de la Caridad Delgado Olivera
13. Redacción – borrador original: Lisdania de la Caridad Delgado Olivera
14. Redacción – revisión y edición: Lexys Manuel Díaz Alonso

Financiación

El trabajo no requirió financiación. Este forma parte de una de las líneas de investigación de la Maestría de Calidad de Software que se desarrolla en la Universidad de las Ciencias Informáticas.