REVISTA TIA

Tecnología, Investigación y Academia -Red Avanzada – RITA
Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología

Modelos de procesos de desarrollo aplicado a un proyecto de arquitectura de software

Development process models applied to a software architecture project

Andrés Otavo Piraban[1], Juan Camilo Pardo Moreno[2]

Citar este documento:

Otavo Piraban Andrés, Pardo Moreno Juan Camilo. Modelos de procesos de desarrollo aplicado a un proyecto de arquitectura de software. . Revista Tecnol.Investig.Academia TIA, ISSN: 2344-8288, 2024. Vol 12 N 1, pp.150- 161- Bogotá-Colombia.

[1] Ingeniero Electrónico, Universidad Central, aotavop@udistrital.edu.co, https://orcid.org/0009-0000-7257-452X
[2] Ingeniero de Sistemas, Universidad de Cundinamarca, jucpardom@udistrital.edu.co, https://orcid.org/0009-0000-0135-8029

Vol. 12 No. 1
Artículo de revisión
Fecha de aceptación: Julio 2024
ISSN: 2344 8288
Fecha de recepción: abril 2024.

Resumen:

Este artículo se centra en la evaluación de diferentes modelos de proceso de desarrollo de software para determinar cuál es el más adecuado para un proyecto específico de definición de arquitectura de una aplicación de pagos electrónicos. Se destacan las características, ventajas y casos de aplicación de cada modelo, con el objetivo de proporcionar un análisis comparativo que sirva como guía para la selección del modelo más apropiado. Se analizarán algunos modelos de desarrollo de software conocidos, considerando aspectos como su estructura, las ventajas y ejemplos de aplicación en la industria, aplicando un enfoque en el proyecto específico para definir arquitectura de una aplicación de pagos electrónicos, resaltando los requisitos y desafíos de estos proyectos. Finalmente, se analizará los modelos de proceso de desarrollo de software, donde se evalúan los diferentes modelos según los criterios requeridos para el proyecto, utiliza como herramienta un cuadro comparativo para tomar una decisión informada y fundamentada sobre qué modelo de desarrollo de software es el más apropiado para el proyecto.

Palabras clave: arquitectura, pagos electrónicos, procesos de desarrollo

Abstract:

This article focuses on the evaluation of different software development process models to determine which is the most suitable for a specific project to define the architecture of an electronic payments application. The characteristics, advantages and application cases of each model are highlighted, with the aim of providing a comparative analysis that serves as a guide for selecting the most appropriate model.Some known software development models will be analyzed, considering aspects such as their structure, advantages and application examples in the industry, applying a focus on the specific project to define architecture of an electronic payments application, highlighting the requirements and challenges of these. Projects. Finally, the software development process models will be analyzed, where the different models are evaluated according to the criteria required for the project, using a comparative table as a tool to make an informed and informed decision about which software development model is the most suitable. appropriate for the project.

Keywords: *architecture, electronic payments, development processes*

1. INTRODUCCION

Los modelos de desarrollo de software son metodologías que definen un paso a paso estructurado para la planeación, desarrollo, implementación y mantenimiento de software orientadas a la creación de software. El objetivo de estos modelos es dar una guía a los equipos de trabajo para que de manera organizada logren construir las funcionalidades del programa según los requerimientos, de la manera más eficiente posible. Permiten controlar y hacer seguimiento de cada una de las actividades durante toda la ejecución del proyecto, evitando malas interpretaciones o desviación de los objetivos cuando los proyectos resultan ser particularmente extensos en términos de tiempo y/o recursos.

La necesidad de estos modelos nace a partir de la conocida "crisis del software" en donde este tipo de proyectos resultaban siendo fuertemente cuestionados incluso antes de su inicio, por la gran cantidad de proyectos fracasados, en su mayoría, por la falta de un esquema de trabajo definido y organizado.

Los proyectos de software en general suelen quedarse cortos. El 14% se cancela sin resultados, el 31% no cumple sus objetivos, el 43% supera su presupuesto y el 49% rebasa el plazo acordado. Sólo el 15% de los proyectos se entregan según lo previsto. (Project Management Institute, 2017). CHAOS Report of the Standish Group ha recogido nuevas cifras del sector de las TI cada dos años. En 2015, solo el 15% de los casos examinados concluyeron con éxito el proyecto. (The Standish Group International, 1995).

Dentro los motivos de fracaso de un proyecto de software están el no cumplimiento de los requisitos, costos excesivos, retrasos en los plazos de entrega, insatisfacción del cliente, la poca o mala documentación, la dificultad de mantenimiento e incluso el impacto negativo que puede causar una vez finalizado, ya sea por alguno de los motivos previamente mencionados o la suma de estos.

No solo basta con la utilización de un modelo de desarrollo de software, sino que sea cual sea el modelo escogido para el proyecto, este se adecue a las características, objetivos y recursos propios del proyecto de manera que su uso sea eficiente y permita un desarrollo optimo en cada una de sus etapas.

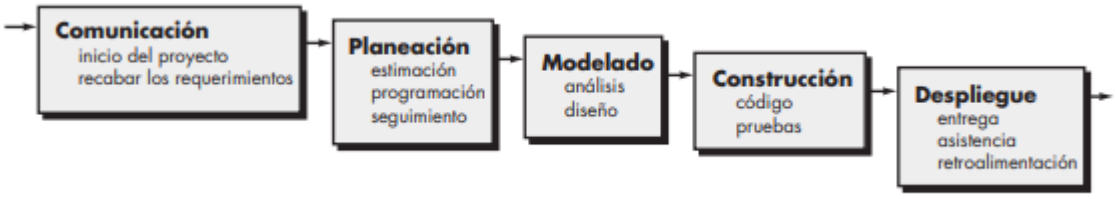
2. Metodología

Se compararán los modelos de proceso de desarrollo de software más conocidos, identificando sus ventajas y desventajas para seleccionar el más adecuado para un proyecto de arquitectura de una aplicación de pagos electrónicos de parqueaderos.

3. Marco teórico

Modelo cascada

Figura 1. Modelo Cascada



Nota. Pressman R. (2010). Ingeniería del Software un Enfoque Practico. [Diagrama]

El modelo de cascada se divide en distintas fases secuenciales, y solo se puede pasar a la siguiente cuando se complete la anterior. En este modelo no hay posibilidad de cambios o errores, por lo que la planificación inicial del proyecto es primordial. Teniendo en cuenta esto, los requisitos deben ser lo más completos posible ya que el equipo trabaja con la investigación y el diseño realizado en las etapas iniciales.

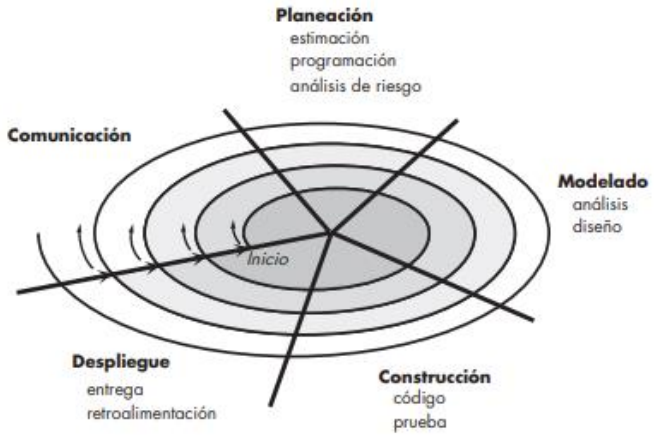
Fases del Modelo Cascada:

- Análisis: Planificación, análisis y especificación de los requisitos.
- Diseño: Diseño y especificación del sistema.
- Implementación: Programación y pruebas unitarias.
- Verificación: Integración de sistemas, pruebas de sistema e integración.
- Implementación: Despliegue de Sistemas
- Mantenimiento: Entrega, mantenimiento y mejora.

Artículo de revisión ISSN: 2344-8288 Vol.12 No. 01. 2024 Bogotá-Colombia

Modelo espiral

Figura 2. Modelo Espiral



Nota. Pressman R. (2010). Ingeniería del Software un Enfoque Practico. [Diagrama]

En este modelo se determina el ciclo de vida del programa a través de distintas espirales repetitivas, las cuales siguen funcionando hasta que se termina el producto.

Con este modelo se pretende mitigar los inconvenientes que surgían de la aplicación del modelo en cascada, basándose en la detección y resolución de riesgos, buscando controlar todos los factores que puedan comprometer la integridad y el funcionamiento del proyecto.

Se caracteriza por seguir ciclos repetitivos que se representan gráficamente en forma de espiral. Este enfoque entrelaza las actividades de especificación, desarrollo y validación, surge de un sistema inicial que se desarrolla rápidamente a partir de especificaciones abstractas, es decir, los requerimientos no están del todo claros y pueden darse cambios en cualquier momento.

Las fases del modelo espiral son:

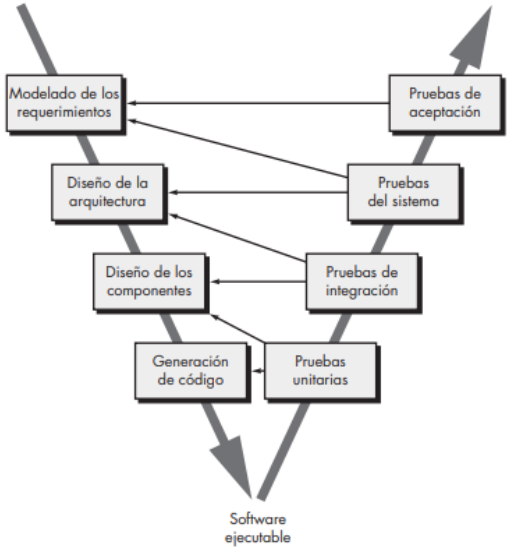
- Planificación: El paso inicial es identificar y establecer objetivos y metas a alcanzar. Luego, como alternativas, presentan las mejores formas potenciales de satisfacer los objetivos. Todo esto requiere una comunicación continua entre el cliente y el equipo de gestión del proyecto.
- Análisis de riesgos: Al planificar y finalizar la estrategia de reducción de riesgos, se identifican los posibles peligros. Cada peligro destacado se somete a un examen exhaustivo. Se pueden

crear prototipos para eliminar la posibilidad de requisitos ambiguos. Los riesgos se minimizan tomando precauciones.

- Ingeniería: Implica la codificación, prueba e implementación del software. Tras una evaluación de riesgos, se adopta el modelo de desarrollo. El modelo a utilizar está determinado por el nivel de riesgo que se ha reconocido para esa fase.
- Evaluación: Valoración del cliente sobre el programa. Se decide si repetir o no el ciclo. Aquí se está planificando la siguiente fase del proyecto.

Modelo en V

Figura 3. Modelo Espiral



Nota. Pressman R. (2010). Ingeniería del Software un Enfoque Practico. [Diagrama]

Este modelo divide el proceso en tres partes: diseño, implementación y pruebas de integración y cualificación. La letra V es una representación simbólica del flujo de desarrollo, aunque también es una referencia a sus dos fases principales, verificación y validación, las cuales a su vez se subdividen en los siguientes pasos:

Fase de Verificación:

- Análisis de requisitos: El objetivo es comprender las expectativas de los clientes sobre el producto, se debe asegurar una buena comunicación con los clientes.

Artículo de revisión ISSN: 2344-8288 Vol.12 No. 01. 2024 Bogotá-Colombia

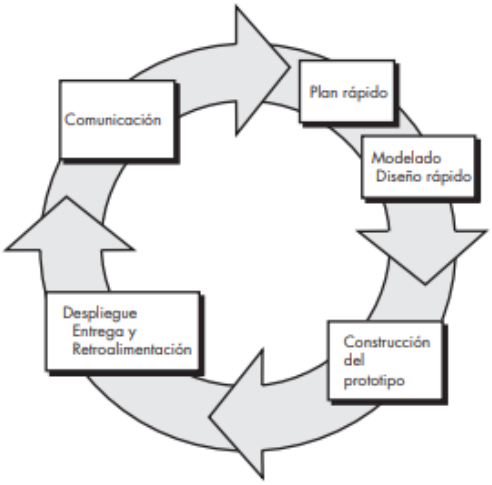
- Diseño de sistemas: Después de la identificación de los requisitos de los clientes y sus expectativas, se debe desarrollar el sistema de diseño detallado para el desarrollo del producto.
- Diseño arquitectónico: El diseño del sistema se segrega en diferentes módulos según sus funcionalidades. Se reconoce la transferencia de datos entre los módulos internos y otros sistemas.
- Diseño del módulo: Los diseños se segregan aún más en módulos más pequeños y detallados.

Fase de Validación:

- Examen de la unidad: Las pruebas unitarias eliminan errores a nivel de código o de unidad.
- Pruebas de integración: Las pruebas de integración validan la comunicación interna entre módulos dentro del sistema.
- Pruebas del sistema: Las pruebas del sistema examinan los requisitos funcionales y no funcionales de la aplicación desarrollada.
- Pruebas de aceptación del usuario (UAT): UAT valida la usabilidad del sistema desarrollado en el mundo real.

Modelo de prototipos

Figura 4. Modelo Prototipo



Nota. Pressman R. (2010). Ingeniería del Software un Enfoque Practico. [Diagrama]

Este modelo inicia con la comunicación y recolección de requerimientos, para luego definir los objetivos para el software, se lleva a cabo un modelado en forma de diseño rápido, este diseño se lleva a la construcción de un prototipo. Este se entrega para recibir la retroalimentación para mejorar los requerimientos. La iteración ocurre repetitivamente hasta que el prototipo satisface las necesidades del cliente. (Pressman R., 2010, p. 37)

Fases:

- Comunicación: En esta etapa inicial se tiene una interacción con el cliente, ya que este debe interaccionar con el prototipo para determinar el refinamiento del proyecto.
- Plan rápido: en esta etapa se realiza la toma y el análisis de requerimientos.
- Modelo diseño rápido: el diseño de un prototipo se enfoca normalmente hacia la arquitectura a nivel superior y a los aspectos de diseño de datos.
- Construcción del prototipo: El prototipo se crea, prueba y corrigen los posibles errores, para entregar un prototipo funcional.
- Desarrollo, entrega y retroalimentación: El cliente prueba el prototipo y realiza la retroalimentación sugiriendo cambios y mejoras, que harán que el prototipo cumpla mejor las necesidades reales.

4. CONCLUSIONES

Tabla 1. Cuadro comparativo de los modelos de proceso de software

Aspecto	Modelo Cascada	Modelo Espiral	Modelo en V	Desarrollo de Prototipos
Enfoque	Secuencial	Iterativo	Secuencial	Iterativo
Manejo de cambios	Dificil	Bueno	Moderado	Bueno
Flexibilidad	Baja	Alta	Media	Alta
Evaluación de riesgos	Baja	Alta	Alta	Media
Adaptabilidad a cambios	Baja	Alta	Media	Alta
Documentación	Completa	Variable	Completa	Variable
Adecuado para proyectos	Estables y bien entendidos	Riesgosos y complejos	Con requisitos claros y estables	Requisitos poco claros o cambiantes
Visibilidad del proceso	Alta	Media	Alta	Media
Requerimientos del cliente	Claros y estables	Variables	Claros y estables	Variables
Tiempo de desarrollo	Largo	Variable	Medio	Corto
Calidad del producto final	Buena	Buena	Buena	Variable
Costos	Moderados	Variables	Moderados	Variables

Nota. Elaboración propia

Los modelos de desarrollo de software pueden ser útiles incluso si el objetivo es definir la arquitectura de una aplicación y no desarrollar el software como tal. A continuación, veremos cómo aplicar cada modelo a un proyecto donde se busca definir la arquitectura de una aplicación para el pago electrónico de parqueaderos.

En el modelo Cascada, la fase de diseño es crucial para establecer la arquitectura del sistema. Se puede dedicar tiempo a realizar un análisis detallado de requisitos y diseñar una arquitectura sólida que cumpla con las necesidades de la aplicación de pago de parqueaderos. Entre las ventajas, se encuentra que proporciona una estructura secuencial que te permite abordar cada aspecto de la arquitectura de manera ordenada y completa. Es adecuado si se tienen requisitos estables y bien entendidos desde el principio.

El modelo espiral es útil cuando los requisitos no están completamente claros o definidos desde el principio del proyecto. En el caso del proyecto particular, se pueden usar las iteraciones para explorar diferentes opciones de arquitectura, evaluar riesgos y tomar decisiones informadas sobre la mejor

opción que aplique a la finalidad de esta. Entre los beneficios, proporciona flexibilidad para adaptarse a cambios y mitigar riesgos a lo largo del proceso de definición de la arquitectura.

El modelo en V se centra en la relación que existe entre las fases de desarrollo y las fases de prueba correspondientes, por lo que, para el proyecto en mención, se puede definir la arquitectura del sistema de manera que esté alineada con las pruebas necesarias para verificar y validar cada componente de la arquitectura. Este modelo proporciona una estructura clara para el desarrollo de la arquitectura junto con las pruebas necesarias para garantizar su calidad y funcionamiento.

El desarrollo de prototipos permitirá construir versiones rápidas y simplificadas de la arquitectura para obtener retroalimentación temprana y validar conceptos. Se pueden utilizar estos prototipos para probar diferentes enfoques arquitectónicos y refinarlos antes de tomar decisiones finales. Este modelo proporcionará una forma rápida y efectiva de validar y ajustar la arquitectura antes de su implementación definitiva.

En resumen, cada modelo de desarrollo de software puede ser adaptado y utilizado para definir la arquitectura de una aplicación de pago electrónico de parqueaderos, dependiendo de las necesidades, el nivel de incertidumbre en los requisitos y la flexibilidad requerida para explorar diferentes opciones arquitectónicas, sin embargo, teniendo en cuenta las ventajas y características de cada modelo particular, se optará por el modelo de prototipos ya que ofrece ventajas clave, como la retroalimentación temprana, su flexibilidad para hacer cambios, la comprensión de las necesidades y expectativas desde las primeras etapas del proyecto y su adaptabilidad. El enfoque de este modelo iterativo puede ayudar a garantizar el éxito del proyecto al satisfacer las necesidades cambiantes de usuarios y sistemas de pago electrónicos, donde la tecnología y las normas regulatorias pueden evolucionar de forma rápida y continua, pero partiendo de una base o prototipo inicial que contemple aspectos como la seguridad y alta disponibilidad, inherentes al tipo de proyecto.

Referencias

Pressman, Roger. (2010). Ingeniería de Software. Un enfoque práctico. sd: Editorial Mc Graw Hill.

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

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

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

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

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

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

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

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

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

Sharma, P., & Sharma, P. (2022, 3 noviembre). Top 9 Software Development Models to Choose From: Phases and Applications. Cynoteck. https://cynoteck.com/es/blog-post/top-software-development-models-to-choose-from/

De la Caridad Delgado Olivera, L., & Alonso, L. M. D. (2021, 31 marzo). Modelos de desarrollo de software. https://www.redalyc.org/journal/3783/378366538003/html/#B1

Felipe, & Felipe. (2021, 15 noviembre). Metodología de espiral: fases y desarrollo - Hosting Plus. Hosting Plus -. https://www.hostingplus.com.co/blog/metodologia-de-espiral-fases-y-desarrollo/

Project Management Institute. (2017). Success Rates Rise Transforming the high cost of low performance. https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf?sc_lang_temp=en

The Standish Group International. (1995). THE CHAOS REPORT. https://www.csus.edu/indiv/v/velianitis/161/chaosreport.pdf