**SERVICIO NACIONAL DE APRENDIZAJE – SENA**

Logotipo

Descripción generada automáticamente

**CENTRO DE COMERCIO REGIONAL ANTIOQUIA**

**TECNOLOGÍA EN ANÁLISIS Y DESARROLLO DE SOFTWARE - 2675805**

**Evidencia conocimiento: GA1-220501093-AA1-EV01 taller sobre metodologías de desarrollo de software**

**DANIEL FELIPE ARIAS CORREDOR**

**2023**

1. Introducción

En el presente documento se hará un recuento de las metodologías de desarrollo de software mas usadas en la actualidad. Se explicará la diferencia entre metodologías tradicionales y ágiles, se darán ejemplos de estas y sus principales características.

1. Definición de metodología de desarrollo de software

Una metodología de desarrollo es el conjunto de técnicas y métodos ideados para “realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito” (Maida y Pacienzia, 2015, p. 12). En otras palabras, una metodología ofrece un marco de trabajo mediante el cual un equipo gestiona el desarrollo de un proyecto. Por tanto, una metodología optimiza el proceso de desarrollo y el producto final, definiendo que se debe hacer, de que manera y en qué momento del proyecto.

Las metodologías de desarrollo se dividen en dos grandes grupos: marcos de trabajo (o metodologías) tradicionales y marcos de trabajo ágiles.

1. Metodologías tradicionales

Enfatizan en la previa planeación del proceso, por lo que se esfuerzan en documentar exhaustivamente todo el proyecto. Por tanto, su fase inicial se enfoca en la planificación, levantamiento de todos los requerimientos y definición de roles, artefactos y herramientas (Maida y Pacienzia, 2015).

Son una herramienta poderosa cuando en un inicio todos los requisitos están plenamente identificados y definidos, además de no proveerse ningún cambio en el desarrollo. De tal manera, su aplicación se hace ideal en proyectos de gran dimensión y en un entorno controlado y predictivo. Permite un flujo de trabajo lineal, lo que hace que la dirección y etapas del proyecto sean bastante claras para todas las partes involucradas en el desarrollo. La arquitectura de software es esencial, por lo que se debe contar con un modelo establecido desde su inicio.

Entre sus desventajas se encuentra su poca flexibilidad y maniobrabilidad cuando surge un cambio. Adicionalmente requiere una documentación extensa y precisa, lo que implica mayor tiempo de desarrollo.

* 1. Cascada

Su nombre se debe a la posición de las actividades de desarrollo, donde “el inicio de cada etapa debe esperar a la finalización de la etapa anterior” (Maida y Pacienzia, 2015). Es una metodología bastante lineal, lo que permite un claro entendimiento del proceso y una buena documentación, a costa de su limitada flexibilidad e identificación de errores cuando se está finalizando el proceso, por lo que todos los requerimientos deben estar bien definidos. La figura 1 (Intriago y Romero, 2018) ilustra el flujo de trabajo de esta metodología.

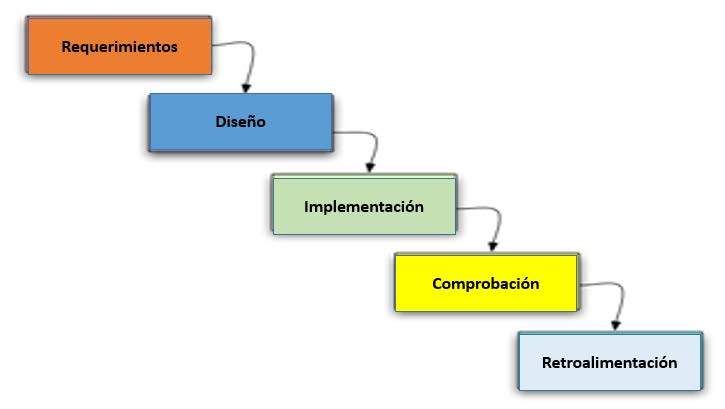


Figura . Actividades del desarrollo de software en el modelo de cascada.

* 1. Espiral

Combina características del modelo en cascada con prototipos iterativos. Los pasos típicos y lineales de la cascada pasan a ser iteraciones, cada uno con cuatro fases:

1. Planificación: Se recolectan los requisitos iniciales y los nuevos
2. Análisis de riesgo: Estudia la viabilidad de los requerimientos
3. Ingeniería: Se desarrolla un prototipo a partir de los requisitos
4. Evaluación del cliente: El cliente comenta y aprueba o rechaza el prototipo

Añade una gestión de riegos y una retroalimentación del cliente, ideal para proyectos grandes. Como desventaja hace mas complejo el desarrollo y añade pasos y costos.

El flujo de trabajo se ejemplifica en la figura 2.

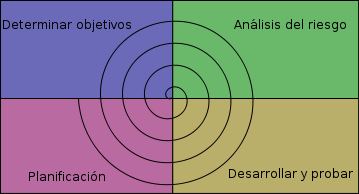


Figura . Modelo espiral. Fuente: Chabacano, [Public domain] via Wikimedia Commons. <https://commons.wikimedia.org/wiki/File:ModeloEspiral.svg>

* 1. Incremental

Se basa en etapas incrementales donde cada etapa o iteración entrega funcionalidad al cliente, por lo que se van entegando partes del proyecto desde las más esenciales a las secundarias. Cada iteración sigue los pasos del modelo cascada, tal como en la figura 3 (Maida y Pacienzia, 2015).

Cuanta con la ventaja de entregar resultados que el cliente puede evaluar o testear haciendo mas énfasis en los desarrollos de cada iteración. Entre sus desventajas hereda del modelo cascada la identificación de errores solo al final del proceso, en este caso al final de cada iteración.

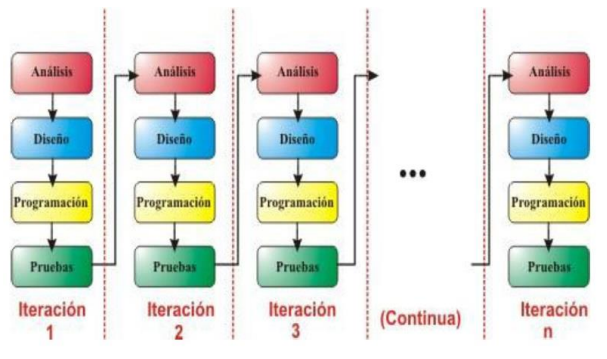


Figura . Modelo de ciclo de vida incremental

1. Metodologías ágiles.

Surgen a partir de la necesidad de reducir la burocracia y rigidez de las metodologías tradicionales, específicamente en proyectos de pequeña y mediana escala. Se fundamenta en la adaptabilidad y capacidad de maniobra. (Navarro et al. 2013). Enfatiza en los individuos e interacciones frente a los procesos y lineamientos, por lo que se hace imprescindible el esfuerzo colaborativo, que viene a ser una comunicación constante entre desarrolladores, clientes y usuarios. Son procesos adaptativos y no predictivos, a diferencia de las metodologías tradicionales, por lo que hace menos hincapié en la arquitectura y documentación, en cambio, requieren más ciclos de entrega.

Entre sus desventajas se encuentra su imposibilidad para se usadas en proyectos grandes con equipos de trabajo extensos. Además, requiere que el cliente supervise constantemente los ciclos de desarrollo.

* 1. Scrum

Se centra en la colaboración, flexibilidad y la entrega incremental de software. Sus iteraciones pasan a denominarse Sprint, donde cada uno es casi un desarrollo independiente, donde se crea una versión utilizable del producto. (SCRUMstudy, 2017). Cada sprint dura un máximo de un mes. Al inicio de cada sprint hay una reunión de planeación, donde se estudian los requerimientos de alta prioridad y al final una reunión de finalización del sprint. Adicionalmente cada día de trabajo cuanta con una reunión inicial de planeación, donde se discute los avances del día anterior y lo que se espera avanzar antes de la siguiente reunión, siendo cada día de trabajo un tipo de iteración individual. Se hace énfasis en la transparencia o visibilidad del proyecto, inspección o adaptación.

La figura 4 (SCRUMstudy, 2017) ejemplifica el flujo de trabajo para un sprint.

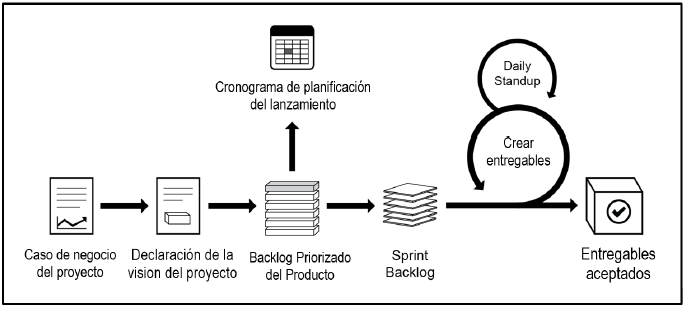


Figura . Flujo de Scrum para un sprint

* 1. Extreme Programming [XP]Ideado para equipos de desarrollo pequeños o medianos en ambientes de desarrollo inciertos o cambiantes. “XP tiene como base cinco valores: Simplicidad, Comunicación, Retroalimentación, Respeto y Coraje” (Navarro et al. 2013). Se enfatiza en la programación en pareja, las pruebas unitarias frecuentes y la propiedad común de todo el código escrito. El diseño debe ser tan simple como se pueda y las mejoras pueden cambiar la estructura del sistema, pero no su funcionamiento. El cambio de requisitos sobre la marcha es un proceso natural. Lo ideal es que haya un representante del cliente a tiempo completo en el sitio de trabajo.
  2. Kanban

Se basa en la visualización del flujo de trabajo mediante el uso de tableros Kanban. Estos tableros se dividen en listas de tareas por hacer, en proceso y realizadas. Tal como se muestra en la figura 5 (Maida y Pacienzia, 2015), los requerimientos o tareas se representan por medio de tarjetas, donde se pasa de una tarea a trabajar en otra únicamente cuando está se terminó o paso a otra lista del tablero. Su visibilidad permite detectar cuellos de botella en el flujo de trabajo. Estos tableros con frecuencia son usados como herramienta en otros modelos de desarrollo.

Diagrama, Tabla

Descripción generada automáticamente

Figura . Tablero Kanban

Conclusión

Este documento sirve como abrebocas para entrar a estudiar las metodologías de desarrollo de software, ya que ejemplifica los modelos de desarrollo mas renombrados. En adición, el desarrollo de esta actividad permite evaluar cual es la metodología idónea para aplicar en el proyecto personal de software.

Referencias

Maida, E., Pacienzia, J. (2015). *Metodologías de desarrollo de software [en línea]. Tesis de Licenciatura en Sistemas y Computación*. [Tesis de Licenciatura, Universidad Católica Argentina]. <http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf>

Navarro, A., Fernández, J. D., Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software. *PROSPECTIVA*, *11*(2), 30-39. <https://www.redalyc.org/pdf/4962/496250736004.pdf>

Intriago, I., Romero, E. (2018). Evaluación del entrenamiento físico y técnico en futbol de las categorías formativas del Club Liga Deportiva Universitaria de Quito. 10.13140/RG.2.2.30187.23840.

SCRUMstudy. (2017). *Una guía para el Cuerpo de Conocimiento de Scrum (Guía SBOK™).* (3ra Ed). ISBN: 978-0-9899252-0-4