Aprendiz: Rafael Dario Escalante Sandoval

Instructor: Andrés Rubiano Cucarían

Equipo de Proyecto:

Saray Agua Acosta

Cristian David Mosquera Rodríguez

Danny Alexander Minota Soto

ANALISIS Y DESARROLLO DE SOFTWARE. (2977466)

Taller sobre metodologías de desarrollo de software.

GA1-220501093-AA1-EV01.

Introducción

Las metodologías de desarrollo son indispensables en los grupos de trabajo y organizaciones relacionadas a la industria del software, partiendo de la información abordada en este componente desarrollar el taller sobre metodologías de desarrollo de software propuesto La planificación y el diseño son fundamentales para el desarrollo del software tanto como en su control, ahorro de tiempo y costes.

Marcos de trabajo tradicionales

Para el desarrollo de un buen producto de *software* se debe iniciar por un excelente proceso de planificación y gestión de este durante todas las etapas y actividades que involucran transformar una idea o requerimiento en un producto o servicio que será usado por un cliente particular. Lo anterior supone que este tipo de enfoques son óptimos en proyectos en los cuales los requisitos están plenamente identificados y delimitados, donde no se producirá ningún cambio en lo establecido mientras el proyecto es finalizado. A continuación, se describen algunas metodologías que se enmarcan en los marcos tradicionales de desarrollo de *software*.

Sus diferentes versiones de metodologías son:

Cascada:

Este es uno de los modelos genéricos más ampliamente conocido en la ingeniería de *software* y se deriva de procesos de ingeniería de sistemas más generales (Royce, 1970). Este modelo plantea un proceso lineal donde las actividades de desarrollo de un producto o servicios de *software* se agrupan en un conjunto de fases sucesivas donde estas son desarrolladas una única vez y los resultados de cada fase son la entrada requerida para cada fase subsiguiente, ninguna fase puede iniciar si la fase anterior no ha sido finalizada generalmente mediante un formalismo que puede ser un documento.

Según Sommerville (2005) el modelo en cascada se compone de cinco (5) etapas principales que se asocian con actividades fundamentales en el proceso de desarrollo de *software*, las cuales son:

¿Cuándo se aplica el método de cascada?

Los gerentes de proyecto suelen recurrir al método de cascada:

- Cuando hay una visión clara de lo que debería ser el producto final.
- Cuando los clientes no tienen posibilidad de cambiar el alcance del proyecto una vez que ha comenzado.
- Cuando el concepto y la definición son las claves del éxito (pero no la velocidad).
- Cuando no hay requisitos ambiguos.

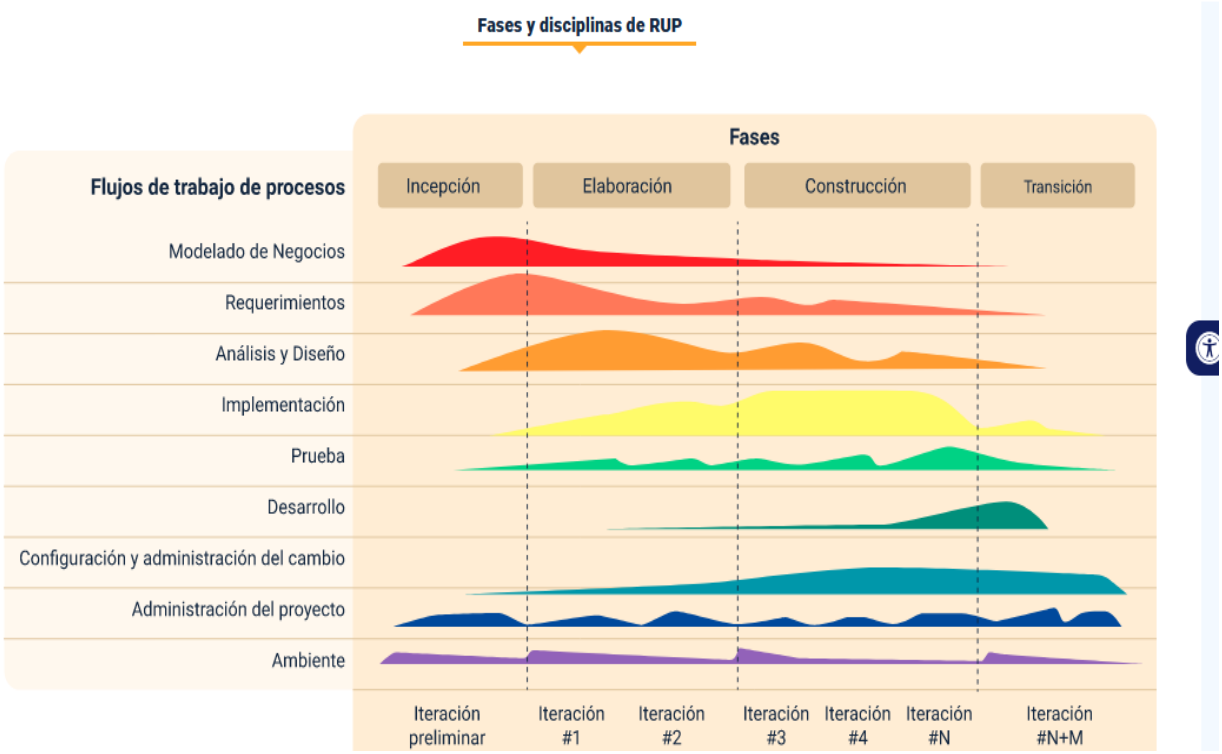
Proceso Racional Unificado - RUP

RUP es una sigla en inglés equivalentes a Proceso Racional Unificado, el cual es un proceso de desarrollo de software tradicional basado en la modelo cascada y que fue desarrollado por la empresa Rational Software que es propiedad de IBM, esta metodología se centra en la arquitectura y es guía por casos de uso (requerimientos) (Kruchten, 2003).

RUP divide el proceso de desarrollo en cuatro grandes fases, dentro de las que se realizan algunas iteraciones donde se desglosan, en mayor o menor intensidad, un

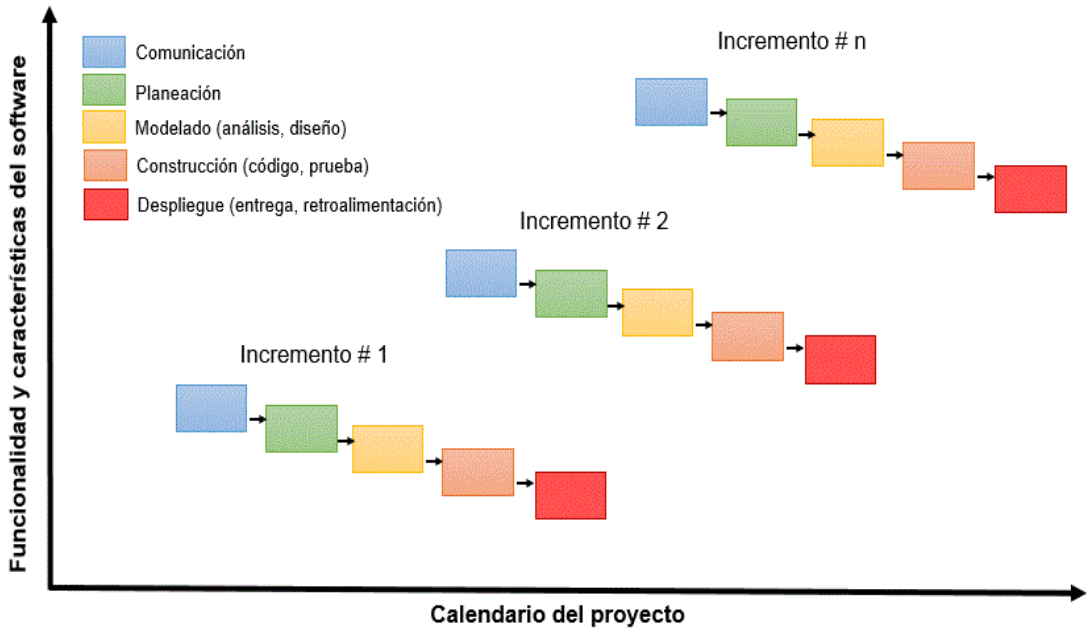
conjunto de disciplinas según la fase que se está abordando. A continuación, se observan las fases y disciplinas propuestas por RUP.

sarrollo de *software*



Incremental

En esta metodología de desarrollo de software se va elaborando el producto final de manera progresiva. En cada etapa se añade una nueva funcionalidad, con la finalidad de ver resultados de una forma más rápida en comparación con el modelo en cascada. Una de las características de este modelo de desarrollo es que **el software se puede empezar a utilizar incluso antes de que se complete totalmente** y, en general, es mucho más flexible que las demás metodologías. En otras palabras el modelo de desarrollo incremental es la duración de vida de desarrollo software, el mismo descompone un proyecto en una serie de incrementos, los cuales suministran una porción de la funcionalidad respecto de la totalidad de los requisitos del proyecto. Los requisitos tienen asignada una prioridad y son entregados según el orden de prioridad en el incremento correspondiente.

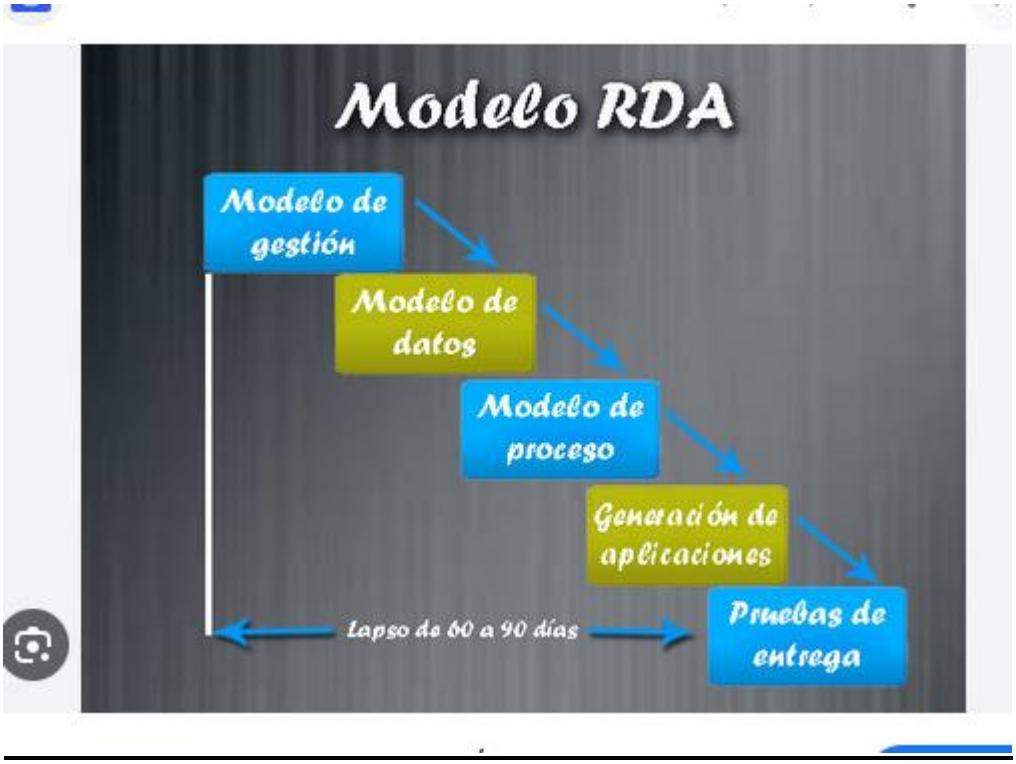


Desarrollo Rápido de aplicaciones

En español Modelo de Desarrollo Rápido de Aplicaciones (DRAI), y por sus siglas en inglés RAD, (Rapid Application Development) y es un procedimiento ágil de desarrollo de software, el cual **da prioridad a las entregas y repeticiones rápidas de prototipos**. Cabe destacar que las repeticiones rápidas reducen el periodo de desarrollo y agilizan la entrega.

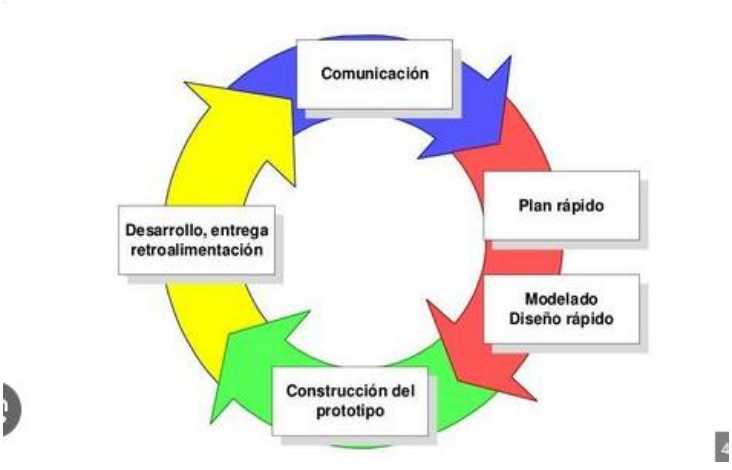
Este modelo se caracteriza por sus equipos compuestos por alrededor de seis personas, incluyendo desarrolladores y usuarios de tiempo completo del sistema, así como aquellas personas involucradas con los requisitos.

Cuando se toma la decisión de adoptar la metodología DRA, se debe tener en cuenta las ventajas y desventajas de su uso y saber por qué utilizar RAD. A continuación, se mencionan algunos de los beneficios principales del uso de la metodología RAD: Avances medibles, Productivos más pronto, Separación de los componentes del sistema, Comentarios constantes de los usuarios, Integración temprana de sistemas, Adaptabilidad, entre otros.



Prototipo

La metodología de desarrollo de software prototipo, se fundamenta en la **elaboración de un prototipo que se construye rápidamente**, para que el cliente o usuario lo pruebe y proporcione su feedback, lo que permite detectar y arreglar lo que está mal, e introducir requerimientos que puedan presentarse, este modelo se fundamenta en el método de **ensayo y error** para entender las especificidades del producto.



Marcos de trabajo ágiles

Las metodologías ágiles proveen un conjunto de pautas y principios que buscan facilitar y priorizar la entrega de producto sobre procesos de documentación exhaustiva, haciéndolos más simples, donde interactúa el cliente final desde las primeras etapas del proyecto. El inicio de las metodologías ágiles nació en el año 2001 a partir del manifiesto ágil de *software* donde se establecen cuatro valores fundamentales (Manifiesto Ágil, 2001):

Programación Extrema - XP

XP es la abreviación comúnmente utilizada para referirse a Extreme Programming, que es un marco de desarrollo de software ágil que busca producir software de alta calidad en contextos con requisitos altamente cambiantes, riesgos que involucran tiempos fijo con tecnologías nuevas y equipos de trabajo pequeños ubicados en un mismo sitio.

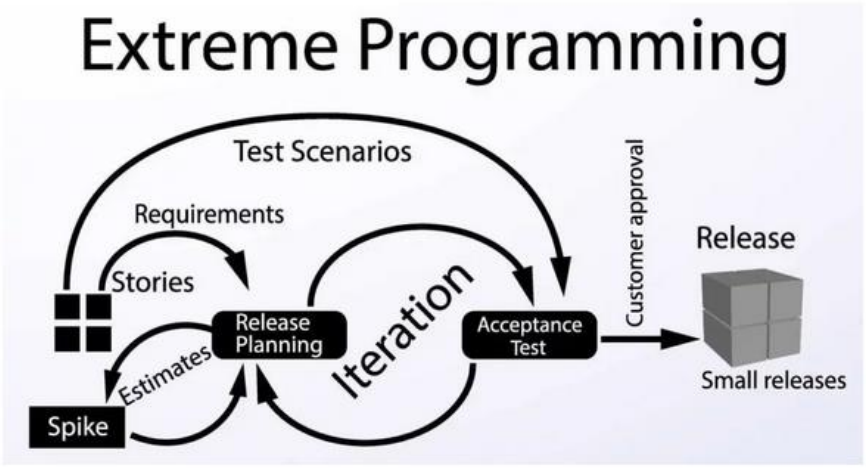
A pesar de que XP promueve principalmente un enfoque de desarrollo de software basado en valores, principios y prácticas de ingeniería, cuando lo analizamos detenidamente (excluyendo las prácticas de ingeniería) podemos ver de es aplicable a cualquier equipo de trabajo, sea o no de software, porque su objetivo principal es promover buenas prácticas y herramientas para que un equipo pueda convertirse en un equipo de excelencia y alto rendimiento.

Roles de XP

Actualmente, los roles de XP no son tan populares como las prácticas, sin embargo, vale la pena conocerlos y entenderlos para comprender la metodología de programación extrema (XP) en su contexto general.

Un equipo de XP incluye seis roles:

- **El cliente** (*Customer*) es la persona responsable de escribir historias de usuarios, establecer prioridades y formular la cartera de productos.
- **El programador** (*Developer*) es un desarrollador normal, que escribe el código y realiza la totalidad de las tareas del proyecto.
- **El entrenador** (*Coach*) es la persona que vigila el trabajo del equipo, lo controla y enseña a sus miembros a implementar las prácticas más efectivas.
- **El rastreador** (*Tracker*) es la persona cuya tarea principal es monitorear el progreso del desarrollo del software y detectar todos los problemas en él.
- **El probador** (*Tester*) es el miembro del equipo responsable de la prueba del producto. La calidad del producto final depende en gran medida de su trabajo.
- **El pronosticador** (*Doomsayer*) es la persona que rastrea los riesgos del proyecto y advierte al equipo sobre ellos.



Scrum

Scrum es un marco de trabajo ágil de muy amplio uso en la industria del software que se fundamenta en los valores y principios ágiles definidos en (Manifiesto Ágil, 2001) y donde se definen tres pilares fundamentales según (SCRUMstudy, 2013) los cuales se describen a continuación:

Transparencia

Hace referencia a que cualquier proceso de Scrum puede ser conocido por cualquiera. Esto es posible por medio de eventos como:

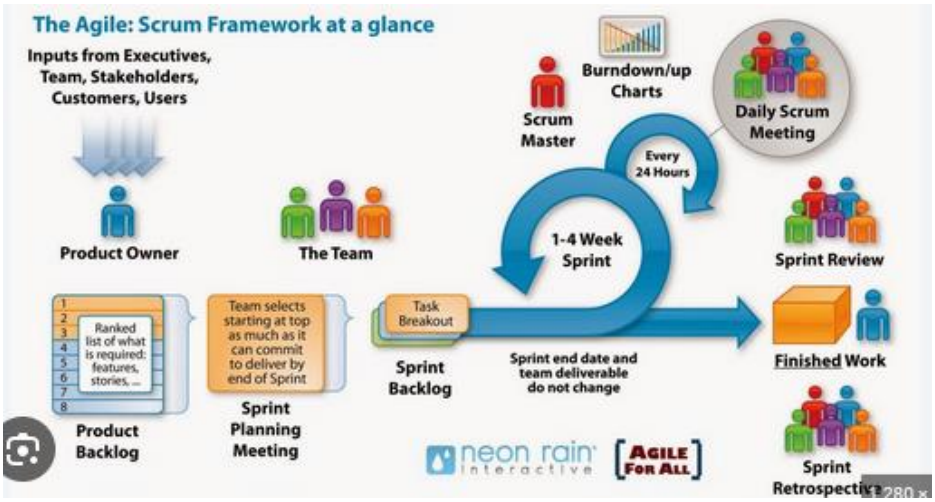
- Las reuniones de revisión y reuniones diarias.
- Artefactos como la pila de producto.
- Cronogramas de lanzamiento.
- Documentos de visión del proyecto.
- Instrumentos de seguimiento, como: el *burndown chart* o el tablero de *Scrum* (*Scrum board*).

Inspección

Permite que cualquiera pueda estar enterado de las actividades realizadas por otros y en general conocer el estado actual de los procesos.

Adaptación

Por medio de la transparencia y la inspección es posible fijar actividades de mejoras que permitan modificar todo tipo de proceso en pro de lograr más altos estándares de calidad.



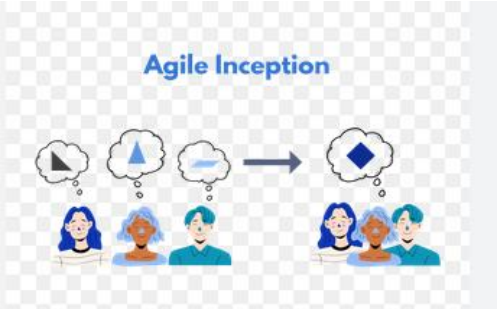
Desarrollo Ligero (Lean)

El Desarrollo Ligero, también conocido como **Lean**, es una metodología especialmente diseñada para equipos de trabajo pequeños. Es necesario que los miembros del mismo sean personas preparadas y motivadas, ya que **el desarrollo del proyecto no se encaminará siguiendo los procesos de un sistema de producción típico, sino que dependerá de la eficacia y unión del equipo de trabajo**. Al igual que Kanban y otras metodologías ágiles, **Lean** está basado en el método de producción Toyota y es una de las más utilizadas para el desarrollo de proyectos a medio plazo.

Agile Inception.

Para finalizar nuestro listado de metodologías ágiles de desarrollo de software, dejamos este caso. Una metodología se enfoca en la entrega temprana de una **versión mínima viable del producto**.

Este proceso se utiliza **al comienzo de un proyecto para establecer una visión clara del producto final** y para definir las características y requisitos que se deben implementar en las primeras etapas de desarrollo. Se basa en el trabajo en equipo y la colaboración con los usuarios para asegurar que el producto satisfaga sus necesidades y expectativas.



Conclusión

Describa con sus propias palabras cuáles son las características fundamentales de un marco de trabajo ágil y un marco de trabajo tradicional:

Marco de trabajo tradicional o de cascada es un proceso mas riguroso, con más enfoque, preciso y ordenado para la elaboración y planificación del desarrollo del software, se caracteriza por colocar primero la estabilidad que la adaptabilidad, es más jerarquizada por un gerente con su función de supervisar.

Marco de trabajo ágil es un enfoque descentralizado con principios adaptables o de un carácter de priorización por parte de los desarrolladores, testeadores y ejecutores comuna participación y comunicación activa entre ellos y el cliente.