DISEÑO DE SISTEMAS DE INFORMACIÓN

TALLER 1

HERIBERTO DAVID YEPES

CORPORACION DE ESTUDIOS

TECNOLÓGICOS DEL NORTE DEL VALLE

INGENIERIA DE SISTEMAS

CARTAGO

2017

TALLER 1

1. ¿Qué es un proceso de software?

Un proceso de desarrollo de software es un conjunto de personas, estructuras de organización, reglas, políticas, actividades y sus procedimientos, componentes de software, metodologías, y herramientas utilizadas o creadas específicamente para definir, desarrollar, ofrecer un servicio, innovar y extender un producto de software.

* Un proceso de software efectivo habilita a la organización a incrementar su productividad al desarrollar software:
* Permite estandarizar esfuerzos, promover reúso, repetición y consistencia entre proyectos.
* Provee la oportunidad de introducir mejores prácticas de la industria.
* Permite entender que las herramientas deben ser utilizadas para soportar un proceso.
* Establece la base para una mayor consistencia y mejoras futuras.

Un proceso de software mejora los esfuerzos de mantenimiento y soporte:

* Define cómo manejar los cambios y liberaciones a sistemas de software existentes.
* Define cómo lograr la transición del software a la operación, y cómo ejecutar los esfuerzos de operación y soporte.

Necesitamos un proceso de software cuya funcionalidad esté probada en la práctica, y personalizado para que cumpla con nuestras necesidades específicas. (Ruvalcaba, 2000)

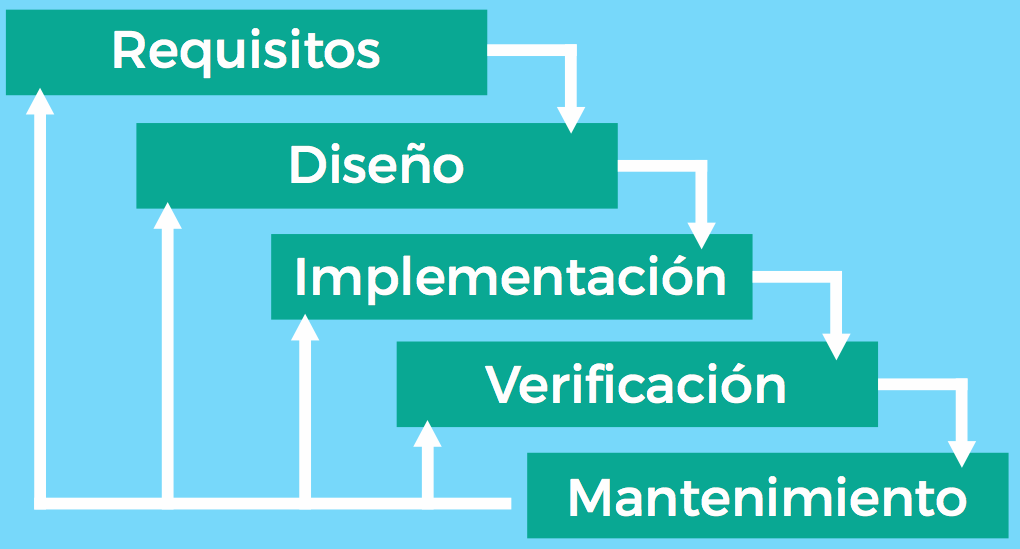
2. Consultar los diferentes modelos de proceso de software cada uno con sus diagramas, características, herramientas, fases y definición de componentes.

**MODELO CASCADA**

**Definición**

Su visión es muy simple, el desarrollo de software se debe realizar siguiendo una secuencia de fases. Cada etapa tiene un conjunto de metas bien definidas y las actividades dentro de cada una contribuyen a la satisfacción de metas de esa fase o quizás a una sub secuencia de metas de la misma.

**Diagrama**



**Fases**

**Requisitos**: Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software.

el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software debe comprender el ámbito de la información del software así como la función, el rendimiento y las interfaces requeridas.

**Diseño**: El diseño del software se enfoca en cuatro atributos distintos del programa; la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz.

**Implementación**: El diseño debe traducirse en una forma legible para la maquina.

**Verificación:** Una vez que se ha generado el código comienza la prueba del programa.

**Mantenimiento:** El software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debidos a que se haya encontrado errores, a que el software deba adaptarse a cambios del entorno externo.

**Características**

* Cada fase genera documentación para la siguiente. Esta documentación debe ser aprobada.
* Una fase no comienza hasta que la anterior ha terminado.
* Requiere disponer de unos requisitos completos y precisos al principio del desarrollo.
* Es una visión del proceso de desarrollo de software como una sucesión de etapas que producen productos intermedios.
* Para que el proyecto tenga éxito deben desarrollarse todas las fases.
* Si se cambia el orden de las fases, el producto final será de inferior calidad.

**PROCESO UNIFICADO ÁGIL**

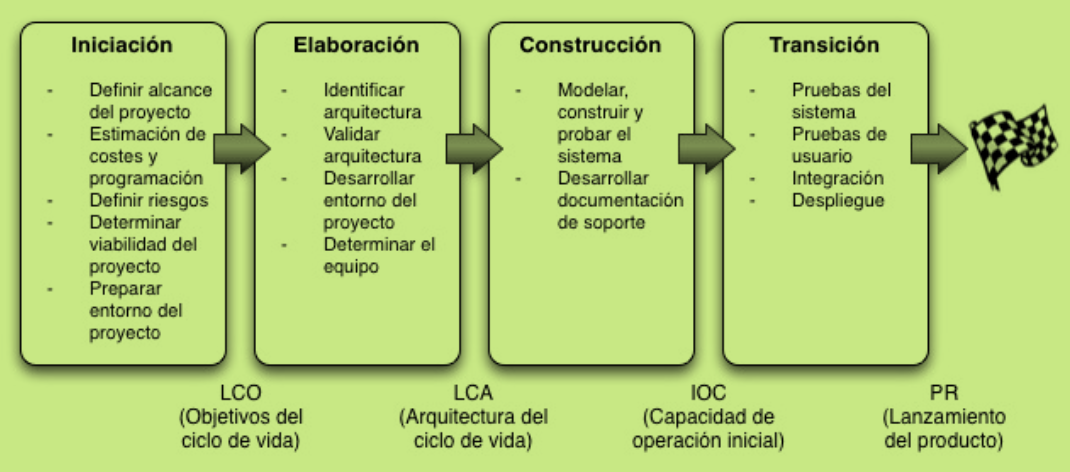
**Definición**

En general, el Proceso Unificado Ágil supone un enfoque intermedio entre XP (eXtreme Programming) y el Proceso Unificado de Rational, y tiene la ventaja de ser un proceso ágil que incluye explícitamente actividades y artefactos a los que la mayoría de desarrolladores ya están acostumbrados.

**Fases**

* **Iniciación**. El objetivo de esta fase es identificar el alcance inicial del proyecto, una arquitectura potencial para el sistema y obtener, si procede, financiación para el proyecto y la aceptación por parte de los promotores del sistema.
* **Elaboración**. Mediante esta fase se pretende identificar y validar la arquitectura del sistema.
* **Construcción**. El objetivo de esta fase consiste en construir software desde un punto de vista incremental basado en las prioridades de los participantes.
* **Transición**. En esta fase se valida y despliega el sistema en el entorno de producción.

**Diagrama**

****

**Características**

* **Modelado**. Su objeto es entender la lógica de negocio de la aplicación, el dominio del problema del proyecto e identificar una solución viable para el dominio del problema.
* **Implementación**. Transformar los modelos en código ejecutable.
* **Pruebas**. Realizar una evaluación de los objetivos para asegurar la calidad.
* **Despliegue**. Planear la entrega del sistema y ejecutar el plan para hacer que el sistema quede disponible para los usuarios finales.
* **Gestión de la configuración**. Gestionar el acceso a los artefactos del proyecto.
* **Gestión del proyecto**. Dirige las actividades que tienen lugar dentro del proyecto.
* **Entorno**. Apoyar el resto del esfuerzo asegurando que los procesos, métodos y herramientas están disponibles para el equipo cuando los necesitan.

**DESARROLLO EN INCREMENTAL**

**Definición**

Surgió el enfoque incremental de desarrollo como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema.

**Fases**

* Requerimientos
* Definición de tareas
* Diseño de incrementos
* Desarrollo de incrementos
* Validación de incrementos
* Integración de incrementos
* Entrega del producto

**Diagrama**



**División del Proceso**

* Análisis
* Diseño
* Código
* Prueba

**Clasificación**

* Inicio
* Desarrollo
* Estandarización
* Innovación
* Sustentabilidad

**MODELO EN ESPIRAL**

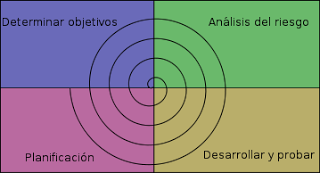
**Definición**

El modelo espiral en el desarrollo de software es un modelo meta del ciclo de vida del software donde es el fuerzo del desarrollo es iterativo, tan pronto culmina un esfuerzo del desarrollo por ahí mismo comienza el otro.

**Fases**

* Determinar o fijar objetivos
* Análisis de riesgos
* Desarrollar, Verificar y Validar
* Planificar

**Diagrama**



**Características**

* Comunicación con el cliente
* Planificación
* Análisis de riesgos
* Ingeniería
* Construcción y adaptación
* Evaluación del cliente

**PROGRAMACIÓN EXTREMA**

La programación extrema, o Extreme Programming (XP), es una metodología de desarrollo ágil, una de las más exitosas en tiempo reciente. Su autor principal es Kent Beck, quien eligió algunas características de otras metodologías y las relacionó de forma que cada una complementara a la otra.

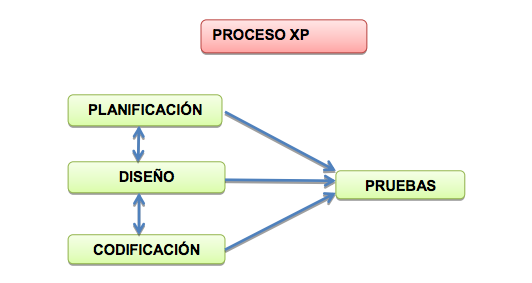
Así, la XP se puede definir como un conjunto de pasos de diversas metodologías, acopladas de manera que sean pasos flexibles a seguir utilizadas con el uso común, para realizar un desarrollo más agradable y sencillo.

Esta metodología tiene como base la simplicidad y como objetivo principal la satisfacción del cliente.

**Fases**

* **Planificación del proyecto**
  + Historias de usuario
  + Release planning.
  + Iteraciones:
  + Velocidad del proyecto
  + Programación en pareja
  + Reuniones diarias.
* **Diseño**
* Diseños simples
* Glosarios de términos.
* Riesgos
* Funcionalidad extra.
* Tarjetas C.R.C.
* **Codificación.**
* **Pruebas.**

**Diagrama**

****

**Características**

* Comunicación: Es muy importante que haya una comunicación constante con el cliente y dentro de todo el equipo de trabajo, de esto dependerá que el desarrollo se lleve a cabo de una manera sencilla, entendible y que se entregue al cliente lo que necesita.
* Simplicidad: En la XP se refiere que ante todo y sin importar qué funcionalidad requiera el usuario en su sistema, éste debe ser fácil. El diseño debe ser sencillo y amigable al usuario, el código debe ser simple y entendible, programando sólo lo necesario y lo que se utilizará.
* Retroalimentación: Es la comunicación constante entre el desarrollador y el usuario.
* Coraje: Se refiere a la valentía que se debe tener al modificar o eliminar el código que se realizó con tanto esfuerzo; el desarrollador debe saber cuando el código que desarrolló no es útil en el sistema y, por lo mismo, debe ser eliminado. También se refiere a tener la persistencia para resolver los errores en la programación.

**MODELO BASADO EN REUTILIZACIÓN**

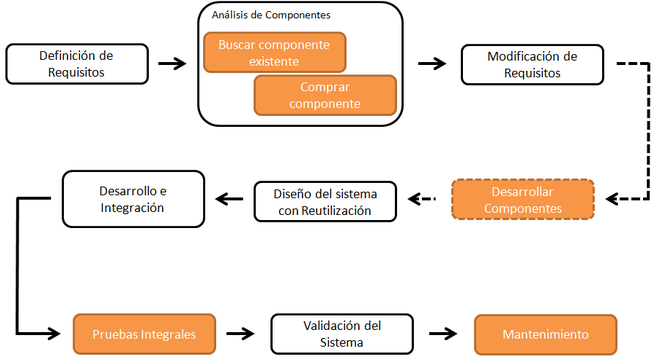
Es el proceso de creación de sistemas de software a partir de un software existente, en lugar de tener que rediseñar desde el principio.

Podemos definirla como el empleo de elementos de software u otros de nivel superior, creados en desarrollo anteriores, para de este modo reducir los tiempos y simplificar el desarrollo del software, mejorando la calidad y reduciendo su costo.

**Conceptos de Reutilización de Software**

* La reutilización de software aparece como una alternativa para desarrollar aplicaciones y sistemas SW de un manera más eficiente, productiva y rápida.
* La idea es reutilizar elementos y componentes SW en lugar de tener que desarrollarlos desde un principio.
* Surge formalmente de 1968
* La idea principal era producir componentes de software como si de componentes eléctricos se tratara.
* El objetivo es reutilizar lo existente sin tener que volver a rediseñarlo desde el principio.

**Diagrama**



3. Realizar un cuadro comparativo con cada uno de los modelos de proceso de desarrollo de software donde se evidencian: ventajas y desventajas.

|  |  |  |
| --- | --- | --- |
| **MODELO DE DESARROLLO** | **VENTAJAS** | **DESVENTAJAS** |
| MODELO CASCADA | * Se tiene todo bien organizado y no se mezclan las fases. * Ayuda a localizar errores en las primeras etapas del proyecto a un bajo costo. * Ayuda a minimizar los gastos de la planificación porque permite realizarla sin problemas. | * Gran dependencia en los requerimientos iníciales * Difícilmente un cliente va a establecer al principio todos los requerimientos necesarios. * El modelo genera pocos signos visibles de progreso hasta el final. * Inicio de la codificación muy tarde en el ciclo de vida del proyecto |
| MODELO UNIFICADO ÁGIL | * El personal sabe lo que esta haciendo: no obliga a conocer detalles. * Simplicidad: apuntes concisos. * Agilidad: procesos simplificados del RUP. * Centrarse en actividades de alto valor: esenciales para el desarrollo. * Herramientas independientes: a disposición del usuario. | * El AUP es un producto muy pesado en relación al RUP. * Como es un proceso simplificado, muchos desarrolladores eligen trabajar con el RUP, por tener a disposición mas detalles en el proceso. |
| DESARROLLO EN ESPIRAL | * No requiere definición completa de los requerimientos del software a desarrollar para alcanzar su funcionalidad * En la terminación de un producto desde la primera iteración es muy factible aprobar los requisitos * Al sufrir retrasos se corre un riesgo menor por que se comprueban los conflictos presentados tempranamente. | * Existe complicación cuando se evalúa los riesgos. * Se requiere la participación continua por parte del cliente. * Se pierde tiempo al volver producir inicialmente una especificación completa de los requerimientos cuando se modifica o mejora el software. |
| DESARROLLO INCREMENTAL | * Con un paradigma incremental se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial. * También provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del software. * El modelo proporciona todas las ventajas del modelo en Cascada realimentado. * Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos. | * El modelo incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido y/o de alto índice de riesgos. * Requiere de mucha planeación, tanto administrativa como técnica. * Requiere de metas claras para conocer el estado del proyecto. |
| DESARROLLO BASADO EN REUTILIZACIÓN | * Reducir el tiempo de desarrollo. * Reducir los costos. * Incrementar la productividad. * No tener que reinventar las soluciones. * Facilitar la compartición de productos del ciclo de vida. * Mayor fiabilidad * Mayor eficiencia (Aunque al principio pueda parecer que no) * Consistencia y la familiaridad, los patrones dentro del software serán más consistentes, tendiendo a facilitar el mantenimiento del producto. | * Necesidad de invertir antes de obtener resultados. * Carencia de métodos adecuados. * Necesidad de formar al personal. * Convencer a los “manager”. * Dificultad para institucionalizar los procesos. |
| PROGRAMACIÓN EXTREMA | * Da lugar a una programación sumamente organizada. * Ocasiona eficiencias en el proceso de planificación y pruebas. * Cuenta con una tasa de errores muy pequeña. * Propicia la satisfacción del programador. * Fomenta la comunicación entre los clientes y los desarrolladores. * Facilita los cambios. * Permite ahorrar mucho tiempo y dinero. * Puede ser aplicada a cualquier lenguaje de programación. * El cliente tiene el control sobre las prioridades. * Se hacen pruebas continuas durante el proyecto. * La XP es mejor utilizada en la implementación de nuevas tecnologías. | * Es recomendable emplearla solo en proyectos a corto plazo. * En caso de fallar, las comisiones son muy altas. * Requiere de un rígido ajuste a los principios de XP. * Puede no siempre ser más fácil que el desarrollo tradicional. |

# Bibliografía

Ruvalcaba, M. (2 de agosto de 2000). *Procesos de desarrollo de software*. Recuperado el 19 de agosto de 2017, de SG Buzz: https://sg.com.mx/revista/1/procesos-software#.WZhz75MjHUo

https://iswugaps2extremeprogramming.wordpress.com/2015/09/14/ventajas-y-desventajas/

https://www.uv.mx/universo/486/infgral/infgral\_15.html

http://ingenieriadesoftware.mex.tl/63758\_aup.html

https://www.ecured.cu/Modelo\_en\_cascada

http://jurifa-ingenieriadesoftware.blogspot.com.co/2012/09/reutilizacion-de-software.html