Unidad 1

Definiciones

Ingeniería de software

Es la aplicación práctica del método científico en el diseño y creación de programas informáticos, así como en la documentación necesaria para su desarrollo, operación y mantenimiento.

Método

El método para la ingeniera de software indica la forma en la que se debe construir software. Incluye el análisis de requisitos, el diseño, la elaboración del programa, las pruebas y el mantenimiento.

Herramientas

Se conoce como herramientas al soporte necesario para el proceso y los métodos. Permiten realizar una tarea con mayor exactitud o calidad. También mejorar la productividad, la eficiencia y la calidad del producto.

Proceso

El **proceso**, en ingeniería de software, se define como el marco de trabajo que abarca en su totalidad a todas las actividades del desarrollo de software. Tiene como objetivo proporcionar una estructura organizada para gestionar y coordinar las distintas etapas del desarrollo del producto, desde su concepción hasta su implementación y el mantenimiento posterior.

El **proceso software** se compone de 4 procesos principales:

1. Proceso de selección de un modelo de ciclo de vida.
2. Proceso de gestión del proyecto.
3. Procesos orientados del desarrollo del software
4. Procesos integrales del proyecto.

La elección de un **proceso software** permite planificar los plazos de un proyecto, la cantidad de personas asignadas al mismo, así como los costos que tendrá su desarrollo.

Un **proceso software** se usa como:

* Guía sobre la documentación a crear y entregar al cliente.
* Base para evaluar que herramientas, técnicas y metodologías serán las apropiadas para brindar soporte.
* Marco para entender y estimar asignaciones y consumos de recursos durante todo el ciclo de vida del producto.
* Descripción que permite evaluar como los sistemas llegan a ser lo que son.

Paradigma

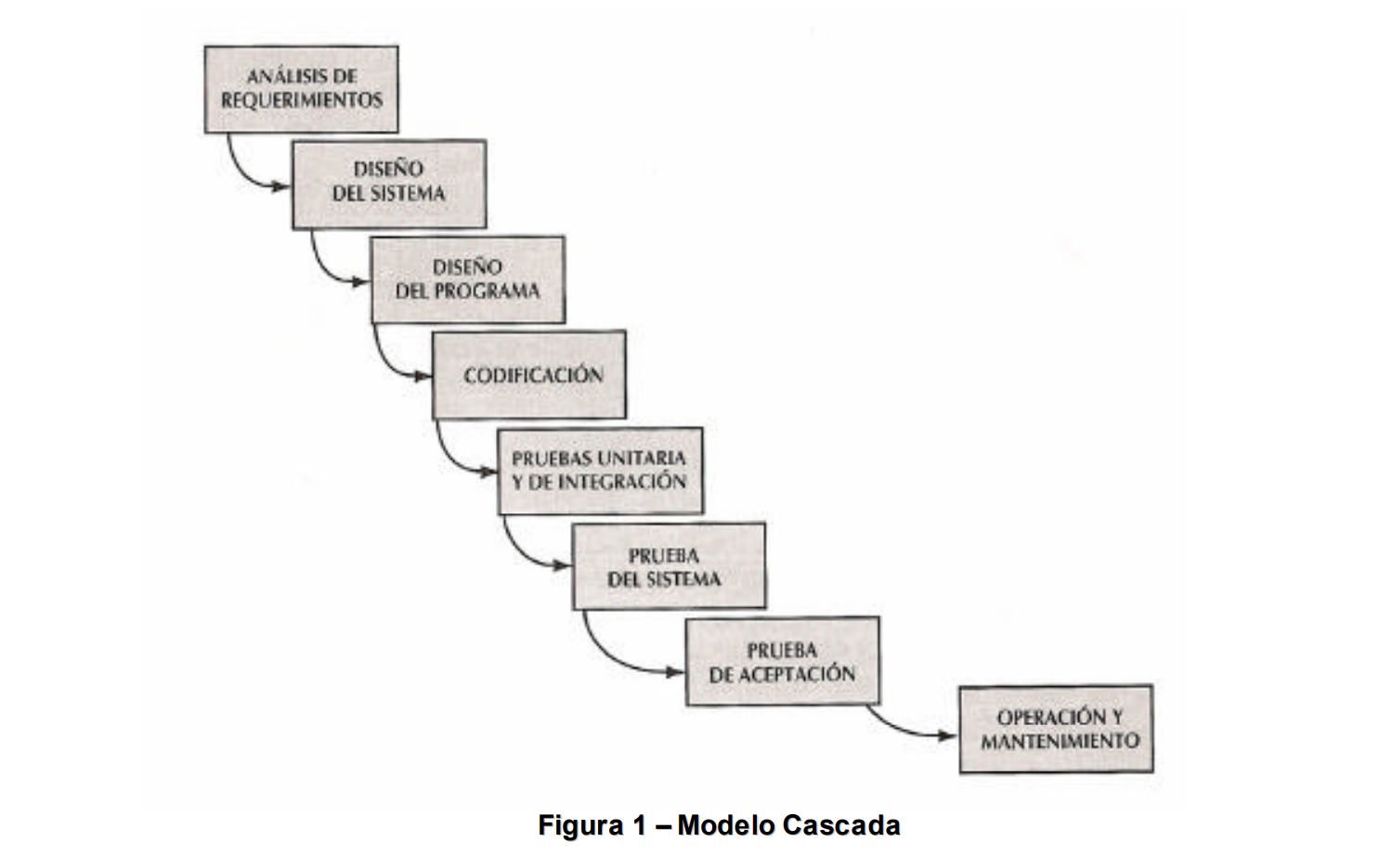
El paradigma representa un enfoque particular para la construcción del software específicamente.

Modelos de ciclo de vida

Modelo en cascada

El modelo de ciclo de vida en cascada divide el desarrollo en etapas que deben completarse una tras otra. Por ejemplo, el equipo de desarrollo solo puede comenzar la etapa de diseño del sistema después de completar la etapa de toma de requerimientos.

Las etapas del modelo en cascada son las siguientes:



**Ingeniería y modelado de Sistemas**

Establece los requisitos que debe cumplir el software teniendo en cuenta la interconexión con otros sistemas.

**Análisis de los requisitos del software**

Se procesan los requisitos centrándose en el software a crear. Se crea la documentación en función de los requisitos. Para crear el software necesario para satisfacer sus necesidades, es necesario comprender el dominio del problema.

**Diseño**

El diseño se subdivide en cuatro pasos: estructurado de datos, arquitectura de software, representaciones de interfaz y proceso algorítmico. El diseño traduce requisitos en un sistema que debe ser evaluado por calidad antes de comenzar con la codificación.

**Codificación**

Al diseñarse el sistema en la etapa previa, la creación de código se realiza mecánicamente.

**Pruebas de integración**

Una vez creado el código, empieza las pruebas de este. Se debe verificar que el sistema cumple con los requisitos planteados, y corroborar la funcionalidad de este.

**Mantenimiento**

El sistema puede cambiar por corrección de errores, por entrega de nuevos requisitos o por cambios en el entorno externo. El mantenimiento implica volver a aplicar las etapas previas sobre el sistema existente.

Las limitaciones de este modelo son las siguientes:

1. El modelo es demasiado rígido para contemplar caminos alternativos cuando se cumple una etapa.
2. El modelo asume que todos los requisitos se definen antes de empezar el diseño y no tienen modificaciones posteriormente.
3. Cuando se definen los requisitos se selecciona el hardware que hará de soporte del sistema. Como consecuencia, el producto a su salida contara con hardware obsoleto.
4. El software no puede probarse hasta que el proyecto este avanzado por lo que un error grave puede complicar y/o encarecer el desarrollo del producto.
5. La naturaleza lineal del modelo produce bloqueos ya que ciertos miembros requieren que otros miembros completen sus tareas para poder seguir con las suyas.
6. El modelo no especifica cómo el resultado de una etapa se transforma en uno nuevo. Por lo tanto, el modelo no proporciona instrucciones sobre cómo manejar los cambios que puede experimentar el producto en una etapa y cómo estos cambios afectan las etapas anteriores.
7. El modelo no tiene en cuenta el proceso evolutivo del software por lo que no comprende los procesos de avance y retroceso del mismo. Tampoco comprende el desarrollo a prueba y error de prototipo de hasta llegar a una solución.

Los beneficios del modelo en cascada son:

1. Etapas se organizan lógicamente. De esta manera, no se puede avanzar una etapa hasta que la anterior no se haya completado.
2. Cada etapa incluye un proceso de revisión de errores y la aceptación del resultado final. Como resultado, los errores que pasan de una etapa a otra deben ser mínimos.

Modelo en cascada V

Diagrama

Descripción generada automáticamente

Este modelo es una variación del modelo en cascada que se centra en que cada etapa tenga otra asociada referida a pruebas que, después de la etapa de codificación, verifiquen si el análisis de requerimientos, el diseño del sistema o el diseño del programa son correctos. Si las pruebas fallan, se redirige el ciclo hacia la etapa fallida para solucionar el problema antes de continuar con el proceso.

En este modelo, a diferencia del modelo en cascada, los errores y sus soluciones son representadas como parte del flujo de creación de software.

Las pruebas que sugiere este modelo son:

1. Prueba unitaria/integración: Valida el diseño del programa. Dirigida por el desarrollador.
2. Prueba del sistema. Valida el diseño del sistema. Dirigida por el desarrollador.
3. Prueba de aceptación: Valida que la toma de requerimientos sea correcta. Es dirigida por el cliente.

Modelo incremental

Diagrama

Descripción generada automáticamente

El modelo de ciclo de vida incremental construye el sistema de manera gradual, agregando nuevas funcionalidades en cada incremento. Este enfoque permite desarrollar el sistema progresivamente hasta que cumpla con todas las funcionalidades requeridas.

El primer paso es crear el núcleo del producto, que incluye los requisitos prioritarios. Las versiones subsecuentes se desarrollan a partir de este núcleo, incorporando mejoras y características solicitadas por el cliente en base a la evaluación de cada incremento.

Las ventajas de este modelo son:

* Los primeros incrementos requieren menos personal, ya que el producto inicial es pequeño, y se puede incorporar más personal a medida que el proyecto evoluciona.
* Mayor feedback del cliente con cada incremento completado.

Modelo evolutivo

El objetivo del modelo evolutivo es entregar mejoras continuas del sistema empezando con una base chica e integrando y mejorando las funcionales a lo largo del tiempo. Se basa en crear un producto lo más rápido posible para recibir comentarios y confirmar su viabilidad.

El producto creado por el modelo iterativo es funcional desde el principio, a diferencia del modelo incremental, pero es por su naturaleza flexible es menos maduro.

Método selección de ciclo de vida

|  |  |  |  |
| --- | --- | --- | --- |
|  | Cascada | Incremental | Espiral |
| DIVISIBLE | NO | SI | SI |
| ALCANCE | SI | SI | NO |

Cuatro “P” de administración de proyectos

Administrar es un concepto que puede referirse al acto de graduar el uso de algo para obtener un rendimiento mejor, ya sea una sustancia, una acción o simplemente el tiempo. Administrar el tiempo, las finanzas personales y el tiempo en el trabajo son algunos ejemplos.

Los pilares de la administración de proyectos son Personal, producto, proceso y proyecto.

Personal

Se basa en tener el recurso humano motivado y capacitado. Existen 2 categorías de motivación, los **internos** (lo que lo mueve a una persona por adentro, sus objetivos y metas) y los **externos** como el dinero y los beneficios materiales como una computadora para trabajar, suele ser algo temporal y suele variar en el tiempo.

People-CMM mide el modelo de madurez de una persona dentro de la empresa. Tiene como objetivo motivas, desarrollar, mejorar y conservar la fuerza de trabajo.

Hay que identificar los roles dentro del personal para particionar responsabilidades y tareas:

**Líderes de equipo**

Los líderes de equipo en el desarrollo de software son responsables de manejar y gestionar un grupo de personas con el objetivo de llevar a cabo el desarrollo y mantenimiento de un producto. Para desempeñar eficazmente su papel, deben poseer habilidades blandas bien desarrolladas, lo que les permite mantener motivado a su equipo y gestionar adecuadamente los requisitos del cliente.

**Equipo software**

Los equipos de software deben tener una estructura y va a depender de diversas variables como la organización de la empresa, la dificultad y el tamaño del producto a desarrollar, el tiempo disponible y la rigidez de la fecha de entrega.

Existen diferentes formas de organizar a un equipo de software. A continuación, se mencionarán brevemente tres maneras:

* **Descentralizado democrático**: No tiene jefe permanente por lo que las decisiones y los enfoques ante los problemas son tomados por consenso de grupo. La comunicación es horizontal entre los miembros.
* **Descentralizado controlado**: Tiene jefe definido el cual coordina tareas específicas y jefes secundarios que tienen responsabilidades. La resolución de problemas es horizontal pero siempre bajo la guía del líder. La comunicación se vuelve vertical en la jerarquía de control.
* **Centralizado controlado**: El jefe de equipo se encarga de la resolución de problemas a alto nivel y la coordinación interna del equipo. La comunicación es completamente vertical.

Producto

La solución materializada a un problema que el equipo de desarrollo se encarga de resolver.

Proceso

Proporciona el marco conceptual para abarcar un proyecto. Debe elegirse el que mejor se pueda adaptar al personal, al producto y al entorno.

Se debe decidir la cantidad de actividades de manera flexible según el tamaño del proyecto. No todos los proyectos requieren la misma capacidad de procesos para llevarse a cabo exitosamente.

Proyecto

Los proyectos deben planificarse y supervisarse para que puedan manejarse. Determinar cómo planificar, supervisar y controlar un proyecto es responsabilidad del gerente del proyecto. Además, se debe especificar cómo se notifica en un momento determinado sobre el estado actual del proyecto en relación con los requisitos y el tiempo restante.

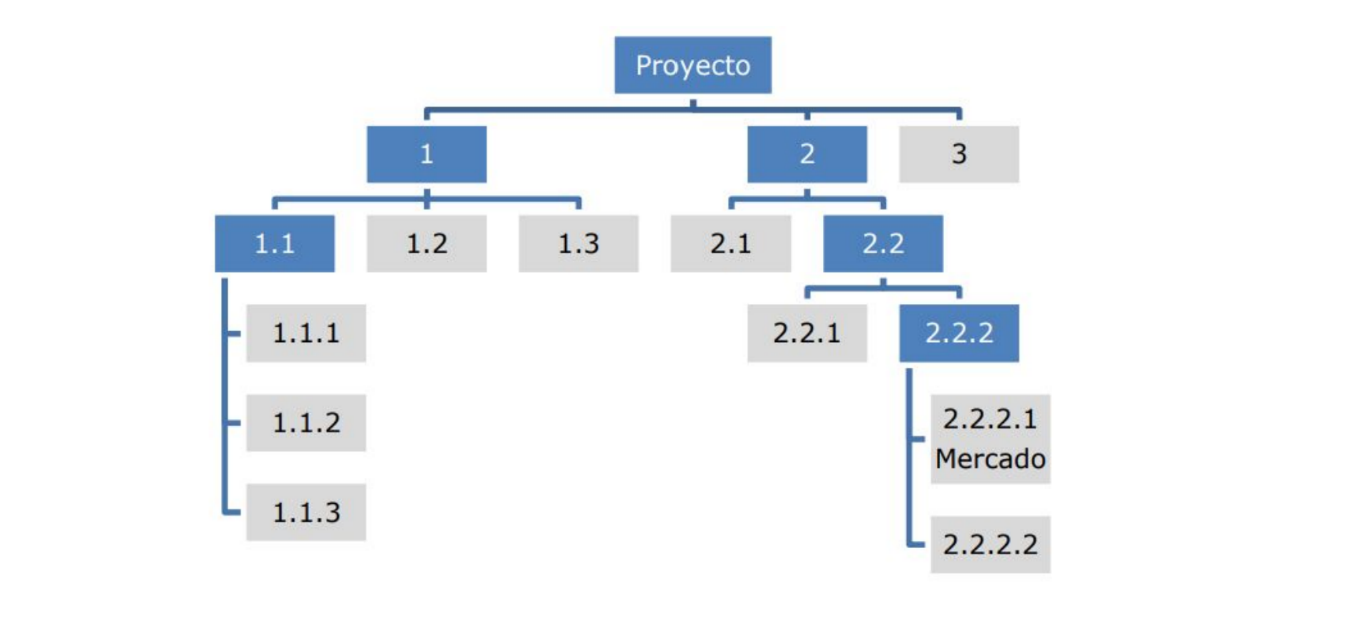
La **fecha resultante** es la fecha para completar un proyecto si comienza mañana. Corre desde mañana y tiene una entrega de 90 días, por ejemplo

La fecha coordinada para la entrega de un proyecto con una fecha de inicio definida se conoce como **fecha estipulada**. Funciona desde la fecha de entrega hasta hoy.

Descomposición

La Estructura de desglose de trabajo (ETD/WBS) es un organigrama jerárquico del proyecto que se divide en partes más pequeñas para facilitar la planificación y el control.

El "Paquete de trabajo" es la composición más pequeña y suele estar agrupado en "Paquetes de control".



Tipos de evaluación

1. Juicio experto: Se basa en la estimación de un miembro del equipo que, por su experiencia y conocimiento en el área, estima una feature.
2. Estimación análoga: Se estima una tarea o proyecto en base a comparar con una actividad anterior.
3. Análisis de reserva: Se estima una tarea pensando en dejar un tiempo extra para cubrirse de posibles complicaciones.
4. Paramétrica: Se estima una tarea en base a comparar con una tarea similar llevada a cabo en un proyecto anterior.
5. Por tres valores: Según una fórmula matemática que toma un tiempo optimista y otro pesimista calculo un tiempo intermedio que será el ideal de la tarea.

Cronograma

Se puede acortar los tiempos definidos en el cronograma por medio de dos tácticas. Una es el **fast-tracking** o ejecución rápida que consiste en realizar tareas en paralelo a costa de aumentar los riesgos de errores. La otra metodología es la compresión o **crashing** que consiste en agregar recursos para acortar la duración del proyecto.

Secuenciar actividades

Tipos de dependencias

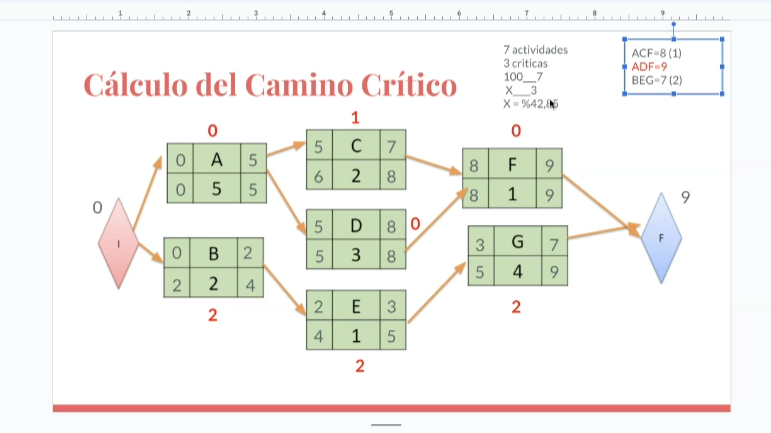
Los tipos de dependencias entre actividades son

1. Obligatorias: Son dependencias que están inherentemente asociadas al trabajo a realizar.
2. Discrecionales: Enfoque preferido elegido por sobre otros para realizar el trabajo. Pueden ser cambiados a placer.
3. Internas: Se basa en las necesidades del proyecto y pueden ser negociadas por el equipo del proyecto.
4. Externas: Se basan en necesidades o deseos externos al proyecto.

Cálculo del camino critico

El camino critico es aquel cuya holgura para realizarse sea igual a cero

La holgura son la cantidad de días se puede demorar una actividad sin comprometer la fecha de entrega del proyecto.



Riesgos

Estrategia reactiva: Cuando un riesgo se convierte en un problema, el gerente de proyecto toma medidas. En una estrategia reactiva, el proyecto se monitorea solo para evaluar riesgos altamente probables. Desafortunadamente, muchos gerentes adoptan este enfoque, reaccionando únicamente cuando los problemas ya han surgido.

Estrategia proactiva: Una estrategia proactiva comienza mucho antes de que se inicie el trabajo técnico. Los riesgos potenciales se identifican y se evalúan en términos de su probabilidad e impacto. Estos riesgos se clasifican por su importancia, y el equipo de software elabora una estrategia de gestión de riesgos. El objetivo principal es evitar los riesgos, pero como no todos son evitables, el equipo también desarrolla un plan de contingencia. Este plan permite responder de manera controlada y efectiva cuando los riesgos se materializan.