❑ Vinculo proceso-proyecto-producto en la gestión de un proyecto de desarrollo de software.

**GESTION DE PROYECTOS**

Comienza antes de iniciar cualquier actividad técnica y continúa a lo largo de la definición, el desarrollo y el soporte del software de computadora.

**El producto**

El desarrollador de sw y el cliente se reúnen para definir los objetivos (qué) y el ámbito del producto (datos primarios, funciones y comportamientos del producto).

Luego se consideran soluciones alternativas, que posibilitan que los gestores y practicantes seleccionen un mejor enfoque, que cumplan las restricciones de fechas limites de entrega, presupuestaria, disponibilidad del personal, interfaces técnicas.

Los requisitos del producto se deben comunicar del cliente al desarrollador, ser divididos en partes constitutivas y distribuirse para que trabaje el equipo de software.

**EL PRODUCTO**

**Ámbito del software**

La primera actividad de gestión de un proyecto de software es la determinación del ámbito de software. El ámbito se define al responder las siguientes preguntas:

* **CONTEXTO** ¿Cómo encaja el software que se desarrollará en un sistema más grande, producto o contexto de negocios, y qué restricciones se imponen como resultado del contexto?
* **OBJETIVOS DE INFORMACION** ¿Qué objetos de datos visibles al usuario se producen como resultado del software? ¿Qué objetos de datos se requieren de entrada?
* **FUNCION Y DESEMPEÑO ¿**Qué funciones realiza el software para transformar los datos de entrada en salida? ¿Existen algunas características de desempeño especiales que deban abordarse?

**El proceso**

Un proceso de software proporciona el marco de trabajo desde el cual se puede establecer un plan detallado para el desarrollo del software. Un pequeño número de actividades del marco de trabajo es aplicable a todos los proyectos de software, sin importar su tamaño o complejidad. Algunos conjuntos de tareas diferentes (tareas, hitos, productos de trabajo y puntos de control de calidad) permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software, así como a los requisitos del equipo del proyecto. Finalmente, las actividades protectoras (control de calidad del software, la gestión de configuración del software y la medición) cubren el modelo de proceso. Las actividades protectoras son independientes de cualquier actividad del marco de trabajo y ocurren durante todo el proceso.

Debe adaptarse al personal y al problema. Se selecciona un marco de trabajo de proceso común, se aplica un paradigma de ingeniería de software adecuado y se elige un conju8nto de tareas para llevar a cabo el trabajo.

**EL PROCESO**

El problema que se presenta es seleccionar el modelo de proceso apropiado para que un equipo de proyecto someta al software a ingeniería.

El gestor de proyectos debe decidir cuál modelo de proceso es más apropiado para:

* Los clientes que han solicitado el producto y el personal que hará el trabajo.
* Las características del producto mismo.
* El ambiente del proyecto en el que trabaja el equipo de software.

Cuando se ha seleccionado un modelo de procesos entonces el equipo define un plan de proyecto preliminar con base en el conjunto de actividades del marco del trabajo del proceso.

Una vez que se establece el plan preliminar, comienza le descomposición del proceso.

**El proyecto**

Los proyectos de software se realizan de manera planificada y controlada por una razón principal: es la única forma conocida de gestionar la complejidad.

Para evitar el fracaso del proyecto, un gestor de proyecto de software y los ingenieros de software que construyen el producto deben eludir un conjunto de señales de advertencia comunes, comprender los factores de éxito críticos que conducen a una buena gestión del proyecto y desarrollar un enfoque de sentido común para planificar, supervisar y controlar el proyecto.

Debe estar organizado en una forma que permita triunfar al equipo de software.

**EL PROYECTO**

La gestión de un proyecto de software exitoso requiere entender que puede salir mal. John Reel define 10 señales que indican que un proyecto de sistemas de información está en peligro:

1. El personal de software no entiende las necesidades de sus clientes.
2. El ámbito del producto está mal definido.
3. Los cambios se gestionan mal.
4. La tecnología elegida cambia.
5. Las necesidades comerciales cambian, o están mal definidas.
6. Los plazos de entrega no son realistas.
7. Los usuarios se resisten.
8. Se pierde el patrocino.
9. El equipo de proyecto carece de personal.
10. Los gestores evitan las mejores prácticas y las lecciones aprendidas.

**Métricas**

Es una forma de medir y una escala, definidas para realizar mediciones de uno o más atributos.

Es un buen medio para entender, monitorizar, controlar, ordenar y probar el desarrollo de software y los proyectos de mantenimiento.

Su objetivo es entender que ocurre durante el desarrollo y mantenimiento, permitir controlar que es lo que ocurre en nuestro proyecto, poder mejorar nuestros procesos y productos.

### Métricas de Software

* **Orientadas al tamaño**: provienen de la normalización de las medidas de calidad y/o productividad considerando el tamaño de software que se haya producido. Se seleccionan las LOC como valor de normalización, no se aceptan universalmente como la mejor forma de medir proceso de software, muchos dicen que son algo fácil de contar y que están presentes en todos los proyectos pero otros dicen que no son algo estable y que las LOC dependen del lenguaje de programación. Errores x KLOC, Defectos x KLOC, etc.
* **Orientadas a la función**: usan como valor de normalización una medida de la funcionalidad que entrega la aplicación, la métrica orientada a la función más usada es el Punto de Función. Los partidarios afirman que el PF es independiente del lenguaje y que se basa en datos que es más probable que se conozcan temprano en la evolución del proyecto. Los opositores dicen que el cálculo se basa más en datos subjetivos que objetivos y que el PF no tiene significado físico, es solo un número.

### Métricas de calidad de Software

El primer objetivo en el proyecto es medir errores y defectos, las métricas que provienen de estas medidas proporcionan una indicación de la efectividad de las actividades de control y de la garantía de calidad.

Medidas de calidad:

* Corrección
* Facilidad de mantenimiento
* Integridad
* Facilidad de eso

### Dominio de las métricas

La aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo de software y sus productos para suministrar información relevante a tiempo, así el líder de proyecto junto con el empleo de estas técnicas mejorará el proceso y sus productos. Su dominio se divide en métricas de Proyecto, de Proceso o de Producto.

* **Métricas de Proceso:** mayor enfoque sobre la calidad lograda como consecuencia del proceso repetible o administrado. Se recompilan en el curso de todos los proyectos y durante largos periodos.

Se mide el proceso para conocer y mejorar sus costos, disminuir el tiempo de construcción, mejorar la performance, etc. Su propósito es proporcionar un conjunto de indicadores de proceso que conduzcan a la mejoría de los procesos de software de largo plazo. Por ejemplo defectos que detectan y reportan los usuarios.

Medidas directas del proceso: costo, esferzo…

* **Métricas de Proyecto:** Esfuerzo/Tiempo por tarea, Errores no cubiertos por hora de revisión, Fechas de entrega reales vs programadas. Su objetivo es minimizar el tiempo de desarrollo (disminuyendo riesgos, problemas potenciales) y valorar la calidad del producto sobre una base actual.

Se mide el proyecto porque un proyecto debería ser entregado con las capacidades funcionales y no requeridas por el cliente, las restricciones impuestas, el presupuesto y el tiempo planificado.

Las métricas del proyecto se consolidan para crear métricas de proceso que sean públicas para toda la organización del software.

* **Métricas de Producto:** pueden ser Dinámicas (recogidas por las mediciones hechas en un programa en ejecución) o Estáticas (recogidas por las mediciones hechas en las representaciones del sistema como diseño, el programa o documentación)

Se miden para garantizar que los artefactos producidos (componentes y modelos) cumplan con los requerimientos del cliente, estén libres de errores, cumplan con los criterios de calidad.

Medidas directas del producto: LOC, cantidad de defectos…

Medidas indirectas: funcionalidad, calidad, complejidad, eficiencia, etc.

### Buenas métricas

* Simples y calculables: fácil aprender a derivar la métrica y fácil de calcular
* Empírica e intuitivamente persuasivas para el ingeniero
* Consistentes y efectivas: sin ambigüedad
* Consistentes en el uso de unidades y dimensiones
* Independiente del lenguaje de programación
* Dar retroalimentación efectiva.

### Métricas básicas para un Proyecto

Métricas apropiadas para el nivel apropiado:

* Desarrollador: esfuerzo, numero de defectos encontrados en revisión por pares, duración estimada y actual de una tarea.
* Equipo de desarrollador: tamaño del producto, numero de tareas planificadas y completadas, % de test ejecutados, numero de defectos encontrados, distribución de esfuerzo
* Organización: tiempo calendario, performance actual y planificada del esfuerzo, performance actual y planificada del presupuesto, defectos en reléase, precisión de estimaciones de Schedule y esfuerzo.

Métricas básicas:

* Tamaño como contar CU, SP, reportes, LOC
* Esfuerzo como horas-hombre
* Tiempo como calendario en días
* Defectos.

Los proyectos se atrasan en calendario pero se recuperan en esfuerzo.

Se seleccionan métricas que provean información para que las personas puedan tomar mejores decisiones, las métricas dan visibilidad de los problemas y las personas resuelven los problemas no las métricas

### Indicadores

Ellos nos dan las tendencias. Principales (sugieren tendencias o eventos futuros) o Secundarios (proveen información sobre las salidas)

* Estratégicos: permiten tener una visión profunda de la eficiencia del proceso, que los gestores evalúen que funciona y que no
* Tácticos: permiten evaluar el estado del proyecto, seguir la pista de los riesgos potenciales, detectar áreas de problemas antes que se conviertan críticas, ajustar el flujo y las tareas de trabajo.

### Eficiencia de un Proceso de Software

Se mide de forma indirecta y se incluyen medidas de: errores detectados antes de la entrega del software, errores detectados e informados por el usuario, esfuerzo humano y tiempo consumido, ajuste con la planificación.