**RESUMEN ISW**

**CAPITULO 21**
**CONCEPTOS DE GESTION DE PROYECTOS**

1. **EL ESPECTRO DE LA GESTION**

La gestión eficaz de la gestión de proyectos de software se enfoca sobre las cuatro P: personal, producto, proceso y proyecto.

### 1.1.1 EL PERSONAL

El factor humano es tan importante que el Software Engineering Institute ha desarrollado un modelo de madurez de la capacidad de gestión del personal (MMCGP).

El modelo de madurez de gestión de personal define las siguientes áreas clave prácticas para el personal de software: reclutamiento, selección, gestión del desempeño, entrenamiento, retribución, desarrollo de la carrera, diseño de la organización y el trabajo, y desarrollo de la cultura del equipo. Las organizaciones que logran altos niveles de madurez en el área de gestión de personal tienen una mayor probabilidad de implementar efectivas prácticas de ingeniería de software.

### 1.1.2 EL PRODUCTO

Antes de planear un proyecto se deberían establecer los objetivos y el ámbito del producto, considerar soluciones alternativas e identificar las restricciones técnicas y de gestión.

El desarrollador del software y el cliente se deben reunir para definir los objetivos y el ámbito del producto.

Los objetivos identifican las metas globales del producto, sin considerar cómo se las lograrán.

El ámbito identifica los datos primarios, las funciones y los comportamientos que caracterizan al producto y los intentos por enlazar tales características en una forma cuantitativa.

Una vez entendidos los objetivos y el ámbito del producto se consideran soluciones alternativas. Aunque se trata relativamente poco detalle, las alternativas posibilitan que los gestores y practicantes seleccionen un “mejor” enfoque, cúmplanlas restricciones que imponen las fechas límites de entrega, las restricciones presupuestarias, la disponibilidad del personal, las interfaces técnicas y miles de factores más.

### 1.1.3 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.

### 1.1.4 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.

## 2. PERSONAL

Los gestores argumentan que las personas son lo principal en los procesos de ingeniería de software, pero sus acciones con frecuencia contradicen sus palabras.

### 2.1. LOS PARTICIPANTES

El proceso de software lo integran participantes que pueden clasificarse dentro de una de cinco categorías:

1. Gestores ejecutivos: definen los aspectos del negocio que usualmente tienen una influencia significativa en el proyecto.
2. Gestores del proyecto: quienes planifican, motivan, organizan y controlan a los profesionales que realizan el trabajo del software.
3. Profesionales: quienes proporcionan las habilidades técnicas necesarias para realizar la ingeniería de un producto o aplicación.
4. Clientes: quienes especifican los requisitos para la ingeniería del software y otros elementos que tienen un interés mínimo en el resultado.
5. Usuarios finales: quienes interactúan con el software una vez que se libera para su uso productivo.

### 2.2. LIDERES DE EQUIPO

La gestión del proyecto es una actividad intensamente humana, por lo tanto, los profesionales competentes con frecuencia no son buenos líderes de equipo. Simplemente no tienen la mezcla correcta de habilidades con el personal.

Weinberg sugiere un modelo MOI de liderazgo, basado en la **M**otivación, **O**rganización e **I**nnovación.
Otra visión de las características que definen un gestor de proyecto eficiente resalta 4 rasgos clave: Resolución de problemas, Dotes de gestión, Incentivos e Influencia y fomento de la cultura de equipo

### 2.3. EL EQUIPO DE SOFTWARE

Existen casi tantas estructuras organizacionales de profesionales para el desarrollo de software como organizaciones que tiene el mismo fin. La estructura organizacional no puede ser fácilmente modificada. Las preocupaciones acerca de las consecuencias prácticas y políticas de cambio organizacional no están dentro del ámbito de responsabilidad del gestor del proyecto de software. Sin embargo, la organización de la gente directamente involucrada en un proyecto de software está dentro del ámbito del gestor del proyecto.
Los factores de proyecto que deberían considerarse cuando se planifica la estructura de los equipos de ingeniería del software son:

- La dificultad del problema que se resolverá.
- El tamaño del programa resultante en líneas de código o puntos de función.
- La vida del equipo.
- El grado en el que el problema puede separarse en módulos.
- La calidad y confiabilidad requeridas del sistema que se construirá.
- La rigidez de la fecha de entrega.
- El grado de sociabilidad que requiere el proyecto.

Constantine, sugiere 4 paradigmas organizacionales para los equipos de ingeniería de software:

- **Paradigma cerrado**, estructura un equipo a lo largo de una jerarquía tradicional de autoridad. Estos equipos pueden trabajar mejor cuando producen software muy similar a los proyectos anteriores, pero será menos probable que sean innovadores.

- **Paradigma aleatorio**, estructura un equipo libremente y depende de la iniciativa individual de los miembros del equipo. Cuando se requieren innovación o adelantos tecnológicos, los equipos que siguen el paradigma aleatorio serán excelentes, pero no son recomendables cuando se requiere desempeño ordenado.
- **Paradigma abierto,** el trabajo se desarrolla en colaboración. La sólida comunicación y la toma de decisiones basada en el consenso son las marcas características de los equipos de paradigma abierto. Las estructuras de equipo de paradigma abierto se adecuan bien a la solución de problemas complejos, pero no pueden desempeñarse de manera tan eficiente como otros equipos.
- **Paradigma sincrónico,** organiza a los miembros del equipo para trabajar en partes del problema con poca comunicación activa entre ellos.

### 2.4. EQUIPOS AGILES

La filosofía ágil alienta la satisfacción del cliente y la temprana entrega incremental de software; pequeños equipos de trabajo enormemente motivados; métodos informales; mínimos productos de trabajo de ingeniería del software y simplicidad global de desarrollo.
El enfoque ágil subraya la competencia individual en conjunción con la colaboración del grupo como factores de éxito cruciales para el equipo. Para aprovechar en forma eficiente las competencias de cada miembro del equipo y fomentar la colaboración eficaz a lo largo de un proyecto de software, los equipos ágiles son “auto organizados”. Un equipo auto organizado no necesariamente mantiene una sola estructura de equipo, sino que más bien aprovecha elementos de los paradigmas aleatorio, abierto y sincrónico.

### 2.5. CONFLICTOS DE COORDINACION Y COMUNICACIÓN

Existen muchas razones por las cuales los proyectos de software se vuelven problemáticos. La escala de muchos esfuerzos de desarrollo es grande, lo que conduce a complejidad, confusión y dificultades significativas en la coordinación de los miembros del equipo.

## 3. EL PRODUCTO

### 3.1. 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 ámbito del proyecto de software no debe ser ambiguo ni incomprensible a niveles de gestión y técnico. Se debe acotar un enunciado del ámbito del software, es decir, se establecen de manera explícita los datos cuantitativos, se anotan las restricciones o limitaciones y se describen los factores que reducen riesgos.

### 3.2. **Descomposición del problema**

La descomposición del problema, a veces llamada partición o elaboración del problema, es una actividad que se asienta en el núcleo del análisis de requisitos de software. La descomposición se aplica en dos grandes áreas:

- La funcionalidad que debe entregarse.
- El proceso que se empleará para entregarlo.

Un problema complejo se divide en problemas menores que resultan más manejables. Ésta es la estrategia que se aplica cuando comienza la planificación del proyecto. Las funciones de software se evalúan y refinan para proporcionar más detalles antes del comienzo de la estimación. Puesto que las estimaciones de costo y planificación temporal están funcionalmente orientadas, con frecuencia es útil cierto grado de descomposición.

## **4.** 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.

### 4.1. **Combinación del producto y el proceso**

La planeación del proyecto comienza con la combinación del producto y del proceso. Cada función que el equipo de software someterá a ingeniería debe pasar a través del conjunto de actividades del marco de trabajo definidas para una organización de software.

### 4.2. **Descomposición del proceso**

Un equipo de software debe tener un grado significativo de flexibilidad al elegir el modelo de procesos de software que sea mejor para el proyecto y las tareas de ingeniería de software que integren el modelo de procesos una vez elegido.

Enfoque secuencial lineal: para realizar un proyecto relativamente pequeño similar a otros que se hayan realizado.
Modelo de desarrollo rápido de aplicaciones: utilizado cuando existen restricciones de tiempo muy ceñidas.
Estrategia Incremental: utilizado cuando la fecha límite es tan ceñida que la funcionalidad completa no puede alcanzarse.
Proyectos con otras características conducirán a la búsqueda de otros modelos.

Una vez elegido el modelo de proceso, el marco de trabajo respectivo se adapta a él. Podrá aplicarse el marco de trabajo genérico: comunicación, planificación, modelado, construcción y despliegue. Funcionará para modelos lineales, iterativos e incrementales, así como evolutivos e incluso para modelos concurrentes o de ensamble de componentes.

La descomposición del proceso comienza cuando el gerente de proyecto pregunta ¿Cómo lograremos esta actividad del marco de trabajo?

**5.** EL PROYECTO

La gestión de un proyecto de software exitoso requiere entender que puede salir mal. John Reel define10 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.

La lista de las señales mencionada, son las causas que conducen a la regla 90-90. El primer 90% de un sistema absorbe el 90% del esfuerzo y tiempo asignados. El último 10% toma el otro 90% del esfuerzo y tiempos asignados.

¿Cómo actúa un gestor para evitar los problemas mencionados? Sugiere un enfoque de sentido común de 5 partes para proyectos de software:

1. Comience con el pie derecho: Entender bien el problema para establecer bien los objetivos y expectativas. Construir el equipo correcto y darle a éste autonomía, autoridad y tecnología.
2. Mantenga el ímpetu: El gestor de proyecto debe proporcionar incentivos, el equipo debe resaltar la calidad en cada tarea que realiza y los gestores ejecutivos deben hacer lo posible por mantenerse fuera del camino del equipo.
3. Rastree el progreso: En un proyecto de software el progreso se rastrea conforme se elaboran los productos de trabajo(código fuente-modelos) y se aprueban como parte de una actividad de aseguramiento de calidad
4. Tomar decisiones inteligentes: Las decisiones del gestor de proyecto y del equipo de software deben encaminarse para mantenerlo simple.
5. Realice un análisis de resultados: Establezca un mecanismo consistente para extraer lecciones aprendidas por cada proyecto. Evalúe la planificación real y la prevista, recolecte y analice métricas, obtenga realimentación por parte del equipo y de los clientes.

RESUMEN

Gestión de proyectos de software: actividad protectora dentro de la ingeniería del software. Comienza antes de iniciar cualquier actividad técnica y continúa a lo largo de la definición, el desarrollo y el soporte del software de computadora.
Esta actividad abarca medidas y métricas, estimación y planificación análisis de riesgos, segumiento y control.
Las 4 P: Personal, Producto, Proceso y Proyecto.
Personal: debe estar organizado en equipos eficientes, motivados para hacer un trabajo de software de alta calidad y coordinados para lograr una comunicación eficaz.

Producto: los requisitos del producto se deben comunicar del cliente al desarrollador, ser divididos en sus partes constitutivas y distribuirse para que trabaje el equipo de software.
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.
Proyecto: debe estar organizado en una forma que permita triunfar al equipo de software.
El elemento central de todos los proyectos de software es el PERSONAL. Los ingenieros de software pueden organizarse en diferentes estructuras de equipo que van desde las jerarquías de control tradicionales hasta los equipos de paradigma abierto.
Se pueden aplicar varias técnicas de coordinación y comunicación para apoyar el trabajo del equipo.

**CAPITULO 23**
**ESTIMACION PARA PROYECTOS DE SOFTWARE**

**INTRODUCCION**
Gestión del proyecto del software: comienza con un conjunto de actividades que en grupo se llaman PLANIFICACION DEL PROYECTO. Antes de que el proyecto comience, el gestor del proyecto y el equipo de software deben estimar el trabajo que habrá de realizarse, los recursos que se requerirán y el tiempo que transcurrirá desde el principio hasta el final.
Una vez que se completen estas actividades, el equipo de software debe establecer un Plan De Proyecto, que defina las tareas y fechas clave de la ingeniería del software, que identifique quién es el responsable de dirigir cada tarea y especifique las dependencias entre tareas que pueden ser determinantes del progreso.

## 1. OBSERVACIONES ACERCA DE LA ESTIMACION

La planificación requiere que los gestores técnicos y los miembros del equipo de software establezcan un compromiso inicial.
Aunque la estimación es tanto un arte como una ciencia, esta importante actividad no necesita realizarse en una forma improvisada.

Puesto que la estimación coloca los cimientos para las demás actividades de planificación del proyecto y ésta proporciona la ruta para la ingeniería del software exitosa, se estaría mal aconsejando si se embarcara sin ella.
El riesgo de estimación se mide por el grado de incertidumbre de las estimaciones cuantitativas establecidas para recursos, costos y programa de trabajo

## 2. EL PROCESO DE PLANIFICACION DEL PROYECTO

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor estimar razonablemente recursos, costo y programa de trabajo. Además, las estimaciones deben intentar definir los escenarios de mejor y peor caso de modo que los resultados del proyecto se puedan acotar.
El plan del proyecto se debe adaptar y actualizar conforme avance el proyecto.

## 3. AMBITO DEL SOFTWARE Y FACTIBILIDAD

El ámbito del software describe las funciones y características que se entregarán a los usuarios finales, los datos que son entrada y salida, el contenido que se presenta a los usuarios como consecuencia de emplear el software, así como el desempeño, las restricciones, las interfaces y la confiabilidad que acotan al sistema.
El ámbito se define al usar una de las dos técnicas siguientes:

- Después de una comunicación con todos los participantes se desarrolla una descripción narrativa del ámbito del software.
- Los usuarios finales desarrollan un conjunto de casos de uso.

Una vez identificado el AMBITO es razonable preguntar ¿Es posible construir el software para satisfacer el ámbito? ¿El proyecto es factible?
El AMBITO no es suficiente para contestar las preguntas mencionadas. Una vez que el ámbito se comprende el equipo de software y otros deben trabajar para determinar si se puede hacer dentro de las siguientes dimensiones:

- Tecnología
- Finanzas
- Tiempo
- Recursos

## 4. RECURSOS

La segunda tarea de la planificación es la estimación de los recursos necesarios para completar el esfuerzo de desarrollo de software.

### 4.1. RECURSOS HUMANOS

El planificador comienza evaluando el ámbito del software y seleccionando las habilidades requeridas para completar el desarrollo. Se especifican tanto la posición organizacional como la especialidad.
El número de personas que requiere un proyecto de software solo se determina después de que se ha hecho una estimación del esfuerzo de desarrollo.

### 4.2. RECURSOS DE SOFTWARE REUTILIZABLES

La ingeniería del software basada en componentes enfatiza la reutilización, es decir, la creación y reutilización de bloques de construcción de software. Tales bloques, usualmente llamados componentes, deben catalogarse para consultarlos con facilidad,

estandarizarse para facilitar su aplicación y validarse para integrarlos fácilmente.

Bennatan sugiere 4 categorías:

- Componentes ya desarrollados: El software existente se puede adquirir de un tercero o se desarrolló internamente para un proyecto previo (están listos y validados).
- Componentes experimentales: Especificaciones, diseño, código o datos de prueba existentes que se desarrollaron para proyectos previos similares.
- Componentes de experiencia parcial: Especificaciones, diseño, código o datos de prueba existentes que se desarrollaron para proyectos previos están relacionados con el proyecto actual, pero requerirán modificaciones sustanciales.
- Componentes nuevos: El equipo de software debe construir los componentes de software específicamente para las necesidades del proyecto actual.

### 4.3. RECURSOS DEL ENTORNO

El entorno que soporta un proyecto de software, con frecuencia denominado entorno de ingeniería de software (EIS), incorpora hardware y software. El hardware proporciona una plataforma que soporta herramientas (software) con que se producen los productos de trabajo basados en una buena práctica de la ingeniería del software.

El planificador del proyecto de software debe especificar cada elemento de hardware.

## 5. ESTIMACION DE PROYECTOS DE SOFTWARE

La estimación de costo y esfuerzo nunca será una ciencia exacta. Sin embargo, la estimación del proyecto se puede transformar de una práctica oscura en una serie de pasos sistemáticos que proporcionan estimaciones con riesgo aceptable.

Para lograr estimaciones confiables de costo y esfuerzo se tienen varias opciones:

- Demorar la estimación hasta más tarde en el proyecto. (Poco práctica, las estimaciones deben realizarse por adelantado).
- Basar las estimaciones en proyectos similares que ya hayan sido completados (Puede funcionar si el proyecto en curso es muy similar a los previos).
- Emplear técnicas de descomposición relativamente simples para generar estimaciones de costo y esfuerzo del proyecto.
- Utilizar uno o más modelos empíricos en la estimación de costo esfuerzo.

Las últimas 2 opciones son enfoques viables, deben aplicarse juntas, cada una empleada como una marca de verificación para la otra.

## 6. TECNICAS DE DESCOMPOSICION

La estimación del proyecto de software es una forma de resolver problemas, en la mayoría de los casos complejos como para considerar una sola pieza. Por ello, se descompone el problema.

### 6.1. TAMAÑO DEL SOFTWARE

La precisión de la estimación de un proyecto de software se manifiesta en varios factores:

1. El grado con el cual el planificador ha estimado adecuadamente el tamaño del producto que se construirá..
2. La habilidad para traducir la estimación del tamaño en esfuerzo humano, programa de trabajo y dinero.

3. El grado en el cual el plan del proyecto refleja las habilidades del equipo de software.
4. La estabilidad de los requisitos del producto y el entorno que soporta el esfuerzo de ingeniaría de software.

Putnam y Myers sugieren 4 enfoques al problema del tamaño:

1. Tamaño de lógica difusa: El planificador identifica el tipo de aplicación, establece su magnitud.
2. Tamaño de punto de función: El planificador desarrolla estimaciones de las características del dominio de la información.
3. Tamaño de componentes estándar: El planificador del proyecto estima el número de ocurrencias de cada componente estándar y luego aplica datos de proyectos históricos para determinar el tamaño de entrega por componente estándar.
4. Tamaño del cambio: El planificador estima el número y tipo de las modificaciones que se deben lograr

## 6.2. ESTIMACION BASADA EN EL PROBLEMA

Las estimaciones de LDC y PF son distintas técnicas de estimación, aunque ambas tienen varias características en común

Las técnicas de estimación LDC y PF difieren en cuanto al detalle requerido para descomposición y el objetivo de la partición.

Mientras mayo sea el grado de partición es más probable que se desarrolle una estimación razonablemente precisa de LDC.

## 6.3. ESTIMACION BASADA EN EL PROCESO

La técnica más común para estimar un proyecto es basar la estimación en el proceso que se empleará. Es decir, el proceso se descompone en un conjunto relativamente pequeño de tareas y se estima el esfuerzo requerido para lograr cada tarea.

La estimación basada en el proceso comienza con un bosquejo de las funciones del software obtenidas a partir del ámbito del proyecto. Cada función requiere realizar un grupo de actividades del marco de trabajo.

Una vez que se combinan las funciones del problema y las actividades del proceso, el planificador estima el esfuerzo que se requerirá para lograr cada actividad del proceso de software en cada función. Luego se aplica la tasa de trabajo promedio (costo/unidad de esfuerzo) al esfuerzo estimado para cada actividad del proceso.

## 6.4. ESTIMACION CON CASOS DE USO

Los casos de uso permiten que un equipo de software comprenda el ámbito del software y los requisitos.

Razones por las cuales, desarrollar un enfoque de estimación con casos de uso, es problemático:

- Los casos de uso se describen empleando muchos formatos y estilos diferentes, no existe un formato estándar.
- Los casos de uso representan una visión externa de visión externa (del usuario), del software y con frecuencia están escritos con diferentes grados de abstracción.
- Los casos de uso no abordan la complejidad de las funciones ni de las características que se describen.
- Los casos de uso no describen el comportamiento complejo que involucran muchas funciones y características.

A diferencia de las LDC o PF, el caso de uso de una persona tal vez requiera meses de esfuerzo mientras que el de otra quizá se implemente en un día o dos.

## 7. MODELOS EMPIRICOS DE ESTIMACION

Los datos empíricos que apoyan la mayoría de los modelos de estimación proceden de una manera limitada de proyectos. Por esta razón ningún modelo de estimación es apropiado para todas las clase de software ni en todos los entornos de desarrollo.
Un modelo de estimación debe calibrarse para reflejar las condiciones locales. El modelo debe probarse mediante la aplicación de los datos recopilados a partir de proyectos completados, colocar los datos en el modelo y luego comparar los resultados reales con los predichos.

## 8. ESTIMACION PARA PROYECTOS ORIENTADOS A OBJETOS

Enfoque planteado explícitamente para software OO:
1 Desarrollar estimaciones aplicando la descomposición del esfuerzo, análisis de PF y cualquier otro método que sea aplicable en aplicaciones convencionales.
2 Aplicar el modelado de análisis OO.
3 Determinar el número de clases clave.
4 Categorizar el tipo de interfaz para la aplicación.
5 Multiplicar el número total de clases por el número promedio de unidades de trabajo por clase.
6 Comprobar de manera cruzada la estimación.

**RESUMEN**

El planificador del proyecto de software debe estimar 3 factores antes de que un proyecto comience:

- Cuánto tiempo tomará.
- Cuánto esfuerzo requerirá.
- Cuánto personal está involucrado

También debe predecir los recursos (hard y soft) que se requerirán y el riesgo involucrado.
La descripción del AMBITO ayuda al planificador a desarrollar estimaciones empleando una o más técnicas que se clasifican en dos amplias categorías:

- Descomposición
- Modelo empírico

Las técnicas de descomposición requieren un bosquejo de las principales funciones del software, seguido por estimaciones de:

- Número de LDC
- Valores seleccionados dentro del dominio de información
- Número de casos de uso
- Número de personas-mes requeridos para implementar cada función
- Número de personas-mes requeridos para cada actividad de ingeniería de software

Las técnicas empíricas usan expresiones para esfuerzo y tiempo obtenidas empíricamente para predecir estas cantidades del proyecto.

**CAPITULO 24**
**CALENDARIZACION DE PROYECTOS DE SOFTWARE**

## CONCEPTOS BASICOS

Aunque existen muchas razones por las cuales el software se entrega con retraso, la mayoría se encuadra en una o más de las siguientes causas:

- Una fecha límite irrealizable establecida por alguien externo al grupo de ingeniería del software e impuesta a los gestores y profesionales del grupo.
- Cambios en los requisitos del cliente que no se reflejan en modificaciones a la calendarización.
- Una subestimación razonable de la cantidad de esfuerzo o de recursos que se requerirán para alcanzar el trabajo.
- Riesgos que no se consideraron cuando empezó el proyecto.
- Dificultades técnicas que no pudieron preverse.
- Dificultades humanas imprevisibles.
- Falta de comunicación entre el personal del proyecto, lo que genera demoras.
- Una falla en la gestión del proyecto porque no reconoció el retraso ni emprendió una acción para corregir el problema.

Pasos a seguir ante una situación de fecha límite:

1. Realizar una estimación detallada usando datos de proyectos previos. Determinar esfuerzo y duración estimados.
2. Aplicar un modelo de proceso incremental. Desarrollar una estrategia de ingeniería de software que entregará la funcionalidad crítica en la fecha límite impuesta, pero demorará otra.
3. Reunirse con el cliente y con la estimación detallada, explicarle por qué la fecha límite impuesta es irrealizable. Asegurarse de señalar que todas las estimaciones están basadas sobre el desempeño en proyectos previos.
4. Ofrezca la estrategia de desarrollo incremental como alternativa.

## CALENDARIZACION DE PROYECTO.

A Fred Brooks se le preguntó cómo se retrasan los proyectos de software en la calendarización y contestó: DE UN DIA A LA VEZ

La realidad de un proyecto técnico es que cientos de pequeñas tareas deben realizarse para lograr una meta mayor.
Si las tareas con “trayectoria crítica” se retrasan en la calendarización, la fecha de terminación del proyecto se pone en riesgo.

Objetivo del Gestor: definir todas las tareas del proyecto, construir una red que bosqueje sus interdependencias, identificar las tareas cruciales dentro de la red y luego seguir su progreso para garantizar que la demora se produce “un día a la vez”.

Calendarización del proyecto de software: es una actividad que distribuye estimaciones de esfuerzo a través de la duración planificada del proyecto al asignar el esfuerzo a tareas específicas de ingeniería de software. La calendarización evoluciona a lo largo del tiempo.

La calendarización para proyectos de ingeniería de software se puede ver desde 2 perspectivas:

- Ya se ha establecido una fecha final para la liberación de un sistema basado en computadora.
- Supone que se han comentado límites cronológicos aproximados, pero que a la fecha final la establece la organización de ingeniería del software.

**PRINCIPIOS BASICOS**

Compartimentación: El proyecto debe dividirse en compartimentos en varias actividades, acciones y tareas manejables.(Descomponer tanto el producto como el proceso).
Interdependencia: Se debe determinar la interdependencia de cada actividad, acción o tarea compartimentada.
Asignación de tiempo: A cada tarea por calendarizar se le debe asignar cierto número de unidades de trabajo, fecha de inicio, fecha de terminación.
Validación del esfuerzo: Todo proyecto tiene un número definido de personas en el equipo de software. Conforme ocurre la asignación de tiempo, el gestor de proyecto debe asegurarse de que, en un tiempo dado, no se han asignado más que el número de personas calendarizado.
Definición de responsabilidades: Toda tarea calendarizada se la debe asignar a un miembro específico del equipo.
Definición de resultados: Toda tarea calendarizada debe tener un resultado definido. En proyectos de software generalmente es un producto de trabajo.
Definición de hitos: Un hito se logra cuando se ha revisado la calidad de uno o más productos de trabajo y se ha aprobado

**RELACION ENTRE EL PERSONAL Y EL ESFUERZO**

| MITO: “Si nos retrasamos en la calendarización, siempre podemos incorporar más programadores y recuperarnos más adelante en el proyecto” |
|---|

Desgraciadamente, agregar más personas en etapas tardías tiene un efecto perturbador sobre éste, lo que provoca que la calendarización se desfase aún más.
Las personas que se agregan tienen que aprender el sistema, y la gente que les enseña es la misma que estaba haciendo el trabajo. Durante la enseñanza no se realiza trabajo y el proyecto experimenta mayores retrasos.

**DEFINICION DE UN CONJUNTO DE TAREAS PARA EL PROYECTO DE SOFTWARE**

Es difícil desarrollar una taxonomía completa de tipos de proyectos de software, en la mayoría de las organizaciones del ramo se encuentran los siguientes proyectos:

1. Proyectos de desarrollo del concepto, los cuales se inician para explorar algunas aplicaciones o conceptos de negocio s de alguna nueva tecnología.
2. Proyectos de desarrollo de nuevas aplicaciones, los cuales se llevan a cabo como consecuencia de una solicitud específica del cliente.
3. Proyectos de mejora de la aplicación, éstos ocurren cuando el software existente experimenta grandes modificaciones en la función, el desempeño o las interfaces visibles para el usuario final.
4. Proyectos de mantenimiento de aplicación, los cuales corrigen, adaptan o extienden el software existente en formas que no sean obvias inmediatamente para el usuario final.

5. Proyectos de reingeniería, éstos se llevan a cabo con la finalidad de reconstruir un sistema existente en todo o en parte.

CALENDARIZACION

Técnicas que se aplican para la calendarización: PERT, CMP, GANTT.

Seguimiento de la calendarización
La calendarización del proyecto proporciona un mapa de carreteras al gestor del proyecto de software. Si se ha realizado de manera adecuada, la calendarización del proyecto define las tareas e hitos que se deben seguir y controlar conforme avance el proyecto. El seguimiento se puede hacer de diferentes maneras:

- Con la realización periódica de reuniones para valorar el estado del proyecto en las cuales cada uno de los miembros del equipo informa del progreso y los problemas.
- Con la evaluación de los resultados de todas las revisiones realizadas a lo largo del proceso de ingeniería del software.
- Con la determinación de si se han logrado los hitos formales del proyecto en la fecha programada.
- Al comparar la fecha de inicio real con la fecha prevista para cada tarea del proyecto.
- Al reunirse de manera informal con los trabajadores para obtener su evaluación subjetiva del progreso, hasta la fecha y los problemas que se vislumbran.
- Con el uso del análisis del valor obtenido para evaluar el progreso cuantitativamente.

**FILMINAS ESTIMACIONES DE SOFTWARE**

**Problemática de la estimación**

- Averiguar lo que costará desarrollar una aplicación (personas, dinero, tiempo).
- Momento en que se desea conocer el costo.
- Siempre se quiere muy pronto.

Cada vez que se avanza más en el proyecto, y se obtiene más información, la estimación es más precisa. El problema es realizar predicciones.

**Proceso de estimación propuesto**

Medir lo que el usuario quiere: Especificar y cuantificar lo que el usuario necesita.
Estimar lo que costará:

- Experiencia individual
- Experiencia de empresa

Debe tenerse en cuenta la experiencia en proyectos previos. Luego de que se hayan desarrollado varios proyectos, se contará con características claves de negocio.

**Métodos Utilizados**

- Basados en la experiencia
  -Juicio experto: Puro, Delphi.
  -Analogía
  -Distribución de la utilización de recursos en el ciclo de vida
- Basados exclusivamente en los recursos
- Método basado exclusivamente en el mercado
- Basados en los componentes del producto o en el proceso de desarrollo
- Métodos algorítmicos

**JUICIO EXPERTO PURO**
Un experto estudia las especificaciones y hace su estimación.
Se basa fundamentalmente en los conocimientos del experto.

Si desaparece el experto la empresa deja de estimar.
Si el experto no está, el equipo no sabe cómo realizar las estimaciones, se debe tratar de transmitir el conocimiento a todo el equipo.

**Estructurando el Juicio Experto**

- **Hacer una WBS con una granularidad aceptable**
- **Usar el método de "Optimista, pesimista y habitual" y su fórmula (0+4h+p)/6.**
- **Use un checklist y un criterio definido para asegurar cobertura.**

**JUICIO EXPERTO WIDEBAND DELPHI**
Un grupo de personas son informadas y tratan de adivinar lo que costará el desarrollo tanto en esfuerzo como en duración.
Las estimaciones en grupo suelen ser mejores que las individuales.
La reunión se prepara 24hs antes, se junta un grupo de expertos para enfrentar opiniones y estimar. Cada uno de los expertos envía su estimación al coordinador. Luego se tabulan las estimaciones (son anónimas).

Se dan especificaciones a un grupo de expertos.
Se les reúne para que discutan tanto el producto como la estimación.
Remiten sus estimaciones individuales al coordinador.
Cada estimador recibe información sobre su estimación, y las ajenas pero de forma anónima.
Se reúnen de nuevo para discutir las estimaciones.
Cada uno revisa su propia estimación y se la envía al coordinador.
Se repite el proceso hasta que la estimación converge de forma razonable.

¿Cuántas rondas son aceptables? Generalmente 3 reuniones son aceptables.

**Analogía**
Consiste en comparar las especificaciones de un proyecto con las de otros proyectos.

Comparar componentes similares
Cuando se compara por analogía hay mucho error, no son fiables.

**Analogía-Factores**

Tamaño: ¿Mayor o menor?
Complejidad: ¿Más complejo de lo usual?
Usuarios: Si hay más usuarios habrán más complicaciones
Otros factores:
Sistema operativo, entornos (la primera vez más).
Hardware, ¿Es la primera vez que se va a utilizar?
Personal del proyecto, ¿nuevos en la organización?

Generalmente lo que más afecta es la complejidad y `para quién se estima.

**Estrategia de estimación**

- Cuente primero
- Use un método para convertir las "cuentas" en estimaciones
- Use el juicio de experto como último resorte.

| Primero se debe contar y fijar la medida. |
|---|

Segundo se debe transformar la cuenta en estimación.
NO USAR UN SOLO MÉTODO SE RECOMIENDA USAR POR LO MENOS 2, Y SE SUELEN USAR DE A PARES.
Generalmente se usa el juicio experto y uno algorítmico.

**Calibraciones y Historical Data**

- Las calibraciones son solamente usadas para convertir las "cuentas" a estimados (líneas de código a esfuerzo, use cases a calendarios, requerimientos a números de test cases, etc.).
- Estimar, siempre involucra algún tipo de calibración sea directa o implícita.
- Las estimaciones pueden ser calibración usando cualquiera de estos tres tipos de datos:

1. Industry Data: datos de otras organizaciones que aportan al mercado y desarrollan productos con algún grado de semejanza y que permite una comparación básica. (En general están es estándares)
2. Historical Data: datos de la organización de proyectos que se desarrollaron y ya se cerraron.(Se guardaron datos-estimaciones)
3. Project Data: datos del proyecto pero de etapas anteriores a la que se está estimando.(Cada vez que se abre una fase estimarla)

**Actividades Omitidas**

- Una de las fuentes de error más común en las estimaciones es omitir actividades necesarias para la estimación del proyecto:
- Requerimientos faltantes
- Actividades de desarrollo faltantes (documentación técnica, participación en revisiones, creación de datos para el testing, mantenimiento de producto en previas versiones).
- Actividades generales (días por enfermedad, licencia, cursos, reuniones de la compañía)
- Uso de buffers o colchón.
- "Nunca tenga temor de que las estimaciones creadas por desarrolladores sean demasiado pesimistas, dado que los desarrolladores siempre generan schedules demasiado optimistas".

Lo que más afecta a la estimación es:

1. Requerimientos no relevantes
2. Requerimientos mal relevados

Siempre guardar una cantidad de días extra por algo que salga mal o falte: buffer o colchón.

Factores que afectan a la estimación: Nos olvidamos de los requerimientos NO FUNCIONALES, sólo tenemos en cuenta los funcionales.

**Estimación de tamaño**

- El número que más se busca en las estimaciones de software es el tamaño del software a ser construido.
- El tamaño puede ser expresado en LDC (líneas de código), puntos de función, número de requerimientos, número de web pages u otra medida.

**Pocker Estimation**

- Popular entre los ágiles practitiones, publicado por Mike Cohn.
- Combina opinión de experto, analogía y desegregación.
- Participantes en planning pocker son desarrolladores
  - Las personas más competentes en resolver una tarea deben ser quienes la estiman.
- El misterio de Fibonacci.

**Pocker Planning**

Defina la lista de actividades, módulos, use cases, etc.
Acuerde y consensue en la más simple, ejemplo tarea z.
Asigne tamaño/complejidad 1 a la tarea z
Ejecute un wide Band delphi para estimar complejidad/tamaño del resto de la lista, comparado con la más simple y solo asignando valores de COMPLEJIDAD de la serie de Fibonacci. Se puede hacer como una ronda de cartas.
Estime el esfuerzo requerido para llevar a cabo la tarea z y aplique el multiplicador a todas las actividades.

**Estimación basada en casos de uso**

- Basada en un conjunto de casos de uso y su complejidad asociada.
- Se agregan valores que modifican el tamaño.
- Se agregan valores que modifican el esfuerzo.

- **Estimación de tamaño:**
  - Complejidad.
  - Actores.
  - Optimización.
  - Reusabilidad.
  - Tecnología.
  - Salidas.
  - Dominio.
- Criterios:
  - Generales.
  - Particulares.

- **Estimación de esfuerzo:**
  - Madurez de la organización.
  - Experiencia en la tecnología.
  - Conocimiento en el proceso.
- Esfuerzo distribuido en el ciclo de vida:
  - Ajuste por solapamiento.
- Criterios
  - Generales.
  - Particulares**.**

**Problemas en las estimaciones**

  - La estimación de tamaño es el paso intelectual más difícil (pero no imposible), y muchas veces se saltea y se trabaja directamente en la estimación del cronograma (tiempo).

- Los clientes y desarrolladores a menudo no reconocen que el desarrollo de software es un proceso gradual, que requiere el refinamiento de las estimaciones tempranas.
- Las organizaciones a menudo no registran, guardan y analizan sus datos históricos de performance de desarrollo de proyecto.
- A menudo es difícil generar cronogramas realistas aceptados por clientes y gerentes.

**Proyectos de mantenimiento**

- Cuando se está estimando el tamaño para un proyecto de mantenimiento se debe tener en cuenta que insertar una nueva funcionalidad será factible si la arquitectura existente del producto puede acomodarse a dicha función. Caso contrario, el esfuerzo de mantenimiento incrementará el retrabajo de la arquitectura.
- Puede existir una sobre estimación si se utilizan modelos de estimación calibrados para nuevos proyectos.
- A menudo los proyectos de mantenimiento tienen fecha de entrega fija así como un número de personal asignado. Es otro elemento con el que hay que lidiar en el momento de hacer la estimación.

**No aplican los métodos de estimación para proyectos nuevos en proyectos de mantenimiento.**

**Estimaciones en Proyectos pequeños**

- Proceso pequeño: trabajan una o dos personas en un tiempo menor a 6 meses.
- Los datos industriales de modelos de estimación que existen no están calibrados para este tipo de proyecto, de modo que son inadecuados y lo mejor es utilizar los datos históricos de la organización.
- Las estimaciones de los proyectos pequeños dependen en gran medida de las capacidades de los individuos que hacen el trabajo. El Personal Software Process es de gran ayuda para estos proyectos.