TEMA 99. LA ESTIMACIÓN DE RECURSOS Y ESFUERZO EN EL DESARROLLO DE SISTEMAS DE INFORMACIÓN.

Actualizado 2020

1. CONCEPTO DE ESTIMACIÓN SW

Estimación es el proceso que predice el personal, coste, tiempo y esfuerzo necesarios para realizar todas las actividades de un proyecto y obtener los productos asociados al mismo.

La estimación es una previsión, siempre conlleva un margen de riesgo

Los factores de riesgo son:

- Complejidad del proyecto
- Tamaño del proyecto: cuanto más grande sea más difícil será estimarlo
- La no disponibilidad de información histórica, es más fácil estimar sobre algo que está documentado.

Lo recomendable es efectuar dos estimaciones, una al comienzo del proyecto, aun a sabiendas de que no será demasiado exacta, y otra una vez que se hayan establecido los requisitos del sistema.

2. MÉTRICAS

Una **métrica** de software es una **medida cuantitativa** del grado en el que un sistema, un componente o un proceso posee un determinado atributo

Tipos de métricas:

- **Las métricas del producto**, que miden diferentes aspectos del software obtenido, a menudo a partir de código fuente expresado en un lenguaje informático determinado
- **Las métricas del proceso**, que intentan medir determinados atributos que hacen referencia al entorno de desarrollo del software y tienen en cuenta la manera de construirlo.
- **Métricas de calidad (indirectas y subjetivas)** funcionalidad, complejidad, eficiencia, fiabilidad, facilidad de mantenimiento
- **Métricas de productividad (directas y objetivas)** líneas de código o los puntos de función,
- **Métricas orientadas al tamaño y a la función (funcionalidad)**
    - *Longitud – líneas de código* tienen la ventaja de ser una medida directa y objetiva, pero tienen el inconveniente de que sus valores no pueden ser medidos hasta que el proceso de codificación ha finalizado, es imprescindible disponer de información histórica y es dependiente del lenguaje de programación
        - LOC: líneas de código
        - KLOC: miles de líneas de código
        - DSI: instrucciones de código fuente realmente entregadas
        - NCSS: líneas de código fuente sin tener en cuenta los comentarios
        - NSLOC: nuevas líneas de código fuente
    - *Punto de Función* (PF, o FPA según la terminología sajona) es una unidad de medida empírica que mide el tamaño del software en términos de funcionalidad, y tiene la ventaja de que los puntos de función son medibles durante los primeros pasos del desarrollo, no es necesario disponer de información histórica, y es independiente del lenguaje de programación. Pero es indirecta y subjetiva

MÉTODO DE ALBRETCH

Dos factores:

- El tamaño del tratamiento de la información. esto es, una cierta medida de la Información tratada y proporcionada por el sistema.
- Un factor de complejidad técnica, que tiene en cuenta la importancia de diversos factores técnicos implicados en el desarrollo e implantación de los requisitos del sistema.

Cinco parámetros o elementos del dominio de información del software :

1. Ficheros lógicos internos. (ILF) mantenidos por la app son p.ej los ficheros maestros, las bases de datos, las tablas de usuario
2. Ficheros de interfaz externos. (IEF) mantenidos por otra app
3. Entradas externas. (EI) añaden o cambian información en un fichero lógico interno
4. Salidas externas. (EO) proporciona al usuario información orientada a la aplicación. OJO NO SON AYUDAS SI MENSAJES DE ERROR
5. Consultas externas. (EQ) peticiones de información a la aplicación SON LAS AYUDAS NO MENSAJES DE ERROR

Para cada uno de los elementos del dominio del software anteriormente considerados, se ha de indicar **su complejidad** como baja, media o alta (son tablas).

Una vez obtenido el grado de complejidad de cada uno de los elementos del dominio del software, **se aplican los factores de ponderación**

Se deben considerar también las **características generales del sistema**. En la formulación se dan **14 características funcionales** y se evalúa cada una de 0 a 5 en función del grado de influencia que tengan (el valor medio es, pues, 2,5). 0 sin influencia - 5 influencia crítica

Estas características generales afectan al resultado de los puntos de función de manera que la suma de los valores de éstas (que se encuentra siempre entre 0 y 70) se denomina **grado total de influencia (TDI).**

MÉTODO MARK II (SYMONS)

- Un sistema se configura como un conjunto de transacciones-tipo lógicas, esto es, **combinaciones lógicas de entrada/proceso/salida**.
- Las interfaces a éste nivel lógico se tratan como una entrada o salida más.
- Las consultas son una combinación más de entrada/proceso/salida.
- El concepto de fichero lógico es casi imposible de definir de manera no ambigua, especialmente en un entorno de base de datos. En su lugar aparece el concepto **de "entidad", que es un objeto. real o abstracto, del universo** del discurso acerca del cual el sistema proporciona información**.**

Las características generales del sistema que considera el método MARK II para calcular los valores de ajuste de la complejidad **son 19**, los catorce del método de Albretch más los cinco siguientes:

- **15. Interfaces con otras aplicaciones**
- **16. Características de seguridad especiales**
- **17. Facilidades especiales de formación de usuarios**
- **18. Utilización directa por terceras partes**
- **19. Requisitos de documentación.**

MODELOS EMPÍRICOS DE ESTIMACIÓN

Estos modelos de estimación emplean fórmulas matemáticas derivadas empíricamente, para predecir el esfuerzo de desarrollo del software (personas-mes), la duración del proyecto (meses) u otros datos, que son requeridos para el proceso de planificación del proyecto software

- Modelos estadísticos. Un ejemplo típico es el modelo de Walston-Felix.
- Modelos basados en teorías, como el modelo de Putnam.
- **Modelos compuestos**, que utilizan una combinación de intuición, análisis estadístico y juicio de expertos. El modelo más representativo es el **COCOMO**
- Modelos de estudio temporal. El más conocido es el modelo de Sterling. Este modelo mide el tiempo de trabajo útil por jornada y persona

CLASIFICACIÓN DE PRESSMAN

- **Modelos de recursos**. Se identifican tres tipos de modelos de recursos :

- Modelos simple-variable estáticos(usan un parámetro). Son modelos que responden a la siguiente ecuación general :

RECURSO = C1 x (CARACTERÍSTICA ESTIMADA) $^{C}{}_{2}$ C1 y C2 son constantes obtenidas de proyectos terminados y documentados, es decir, de datos históricos.

- Modelos multivariable estáticos (+ 1 parámetro). Estos modelos también hacen uso de *datos históricos para obtener relaciones empíricas*

- Modelos multivariable dinámicos (N drivers pero se puedne cambiar durante la estimación). Son modelos que proyectan los requerimientos de recursos como una función del tiempo, como, por ejemplo, el **modelo de Putnam.**

MODELO COCOMO (BOHEM)

(ES compuesto y de líneas de código)

2 versiones: COCOMO clásico 1981 y COCOMO 2000

- basa su estimación en el **número de instrucciones de código fuente de los diferentes programas que componen la aplicación (Delivered Source Instructions: DSI)** solo contabiliza las suministradas, es decir creadas por el personal del proyecto.
- es muy adecuado para ser aplicado a los desarrollos que siguen el ciclo de vida en cascada
- El periodo de desarrollo que cubre COCOMO abarca **desde el Análisis Funcional hasta la Implantación y Explotación del Sistema**. Las otras fases deben estimarse de manera separada.
- COCOMO supone que el Análisis de Requisitos del Sistema no se variará de manera sustancial a lo largo del proyecto.
- El Modelo COCOMO considera que un **mes-hombre consta de 152 horas de trabajo, distribuidas en 19 jornadas de 8 horas**

tres versiones :

- **Modelo Básico** : Es adecuado para estimaciones rápidas del orden de magnitud de los costes de desarrollo, pero su precisión es limitada ya que calcula el esfuerzo y el coste de desarrollo sólo en función de las líneas de código fuente estimadas Margen de error de un 29%, se suele tolerar hasta un 30%
- **Modelo Intermedio** : Además de las líneas de código fuente estimadas, tiene en cuenta una serie de factores como : limitaciones del equipo físico, calidad y experiencia del personal, uso de técnicas y herramientas actuales, etc., incluyendo estos factores en términos de su impacto global sobre la totalidad del proyecto En el 70% de las ocasiones el margen de error no supera el 20%
- **Modelo Avanzado o Detallado** : De manera análoga al intermedio, tiene en cuenta la influencia de los factores mencionados, si bien, a nivel de cada fase del proyecto (análisis, diseño y construcción). Principio de partición

tres modos de complejidad :

A. **Modo Orgánico** : Es válido para desarrollos relativamente pequeños y sencillos (proyectos de tamaño hasta un máximo de 50 KLDC), en los que un equipo de proyecto estable y con buena experiencia en la aplicación trabaja con unos requisitos poco rígidos.
B. **Modo Semiacoplado o Semilibre** : Es válido para desarrollos de tipo intermedio en cuanto a magnitud (300 KLDC) y complejidad. El equipo del proyecto tiene un nivel medio de experiencia y está formado por una combinación de personal experto e inexperto.
C. **Modo Empotrado o Rígido** : Con independencia del tamaño o magnitud del proyecto, este modo es imprescindible cuando hay que operar dentro de unas restricciones muy estrictas, y el producto debe explotarse en un entorno muy acoplado de hardware, software, normativa y procedimientos operativos.

El modelo COCOMO 2.000 contempla tres versiones :

incluye ya como **aspecto central el fenómeno de la reutilización del software**

- *Modelo de Composición de la Aplicación (Modelo AVM).*
  - se utiliza principalmente en aplicaciones de prototipado y aplicaciones basadas en generadores de pantallas
  - Este modelo se basa en los Puntos Objeto (PO) y en Factores de Reusabilidad y de Productividad.

- *Modelo de Diseño Preliminar (Modelo EDM).*
  - se debe utilizar en las etapas iniciales del proyecto cuando aún se conoce poco sobre el tamaño del producto, la plataforma y el personal
- *Modelo Post-Arquitectura (Modelo PAM*).
  - utilizar una vez que la arquitectura del sistema empieza a tomar forma, esto es, durante la fase de Diseño.,
  - se pueden utilizar las nuevas líneas de código (NSLOC) (New Source Lines of Code) cuando se trata de estimar los costes de una aplicación nueva, o las líneas de código adaptadas (ASLOC) (Adapted Source Lines of Code) cuando se trata de estimar una aplicación en la cual se utilizará básicamente el diseño o el código de otras aplicaciones ya disponibles

MODELO DE PUTNAM

(su herramienta es SLIM)

Es un modelo de recursos del tipo multivariable dinámico (n drivers varían a lo largo del tiempo) que asume una distribución específica del esfuerzo a lo largo de la vida de un proyecto de desarrollo de software.

Defiende que el esfuerzo de desarrollo se distribuye a lo largo de la duración del proyecto según la función de densidad de probabilidad de Rayleigh-Norden

Partiendo de la Ecuación del Software :

Producto (LDC, PF, etc) = Constante (Factor de proporcionalidad) x Esfuerzo x Tiempo

Putnam define:

$$L = C \times K^{1/3} \times T^{4/3}$$

L = Número de instrucciones fuente producidas

K = Esfuerzo durante todo el ciclo de vida, en personas-año.

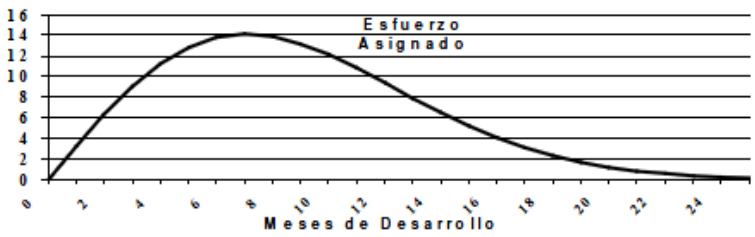
T = Tiempo de desarrollo, en años.

C = Constante dependiente de la tecnología, cuyos valores son :

2.000, para un entorno pobre de desarrollo del Sw. (Sin metodología).

8.000, para un entorno bueno de desarrollo. (Con metodología).

11.000, para un entorno excelente (Con herramientas y técnicas automáticas).



STAFFING SIZE

Es un conjunto de métricas para estimar el número de personas necesarias en un desarrollo Orientado a Objetos y para determinar el tiempo de su participación en el mismo.

Se consideran:

- ✓ 10-15 días para una clase en producción (incluyendo documentación y pruebas)
- ✓ 6-8 días para desarrollar prototipo (solo pruebas unitarias)

Es necesario discernir entre:

- ✓ **Número de clases clave**. Representan el dominio del negocio a desarrollar y son las que se definen en las etapas iniciales del análisis. Habitualmente entre el 20-40% de las clases.
- ✓ **Número de clases secundarias**. Son las clases no indispensables para el dominio del negocio. Se incluyen aquí las interfaces de usuario, las comunicaciones, etc…

MÉTODO KARNER – ESTIMACIÓN POR PUNTOS DE CASO DE USO

Es un método de estimación de esfuerzo para proyectos de software, a partir de sus casos de uso. Fue desarrollado por Gustav Karner en 1993, basándose en el método de puntos de función, y supervisado por Jacobson.

El método **utiliza los actores y casos de uso** para calcular el esfuerzo que significará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan **factores de entorno** y la **complejidad técnica** para ajustar el resultado.

Para obtener la estimación se realizarán los siguientes cálculos:

1. **UAW – Factor de peso de los actores sin ajustar:** Consiste en la evaluación de la complejidad de los actores con los que tendrá que interactuar el sistema. Este puntaje se calcula determinando si cada actor es una persona u otro sistema, a la forma en la que este interactúa con el caso de uso y la cantidad de actores de cada tipo.
2. **UUCW - Factor de peso de los casos de uso sin ajustar:** Este punto funciona muy similar al anterior, pero para determinar el nivel de complejidad se puede realizar mediante dos métodos: basado en transacciones o basado en clases de análisis.
3. **UUCP - Puntos de caso de uso sin ajustar:** Al inicio de un proyecto de software, cuando apenas se conocen los casos de uso y sus actores asociados, se puede proyectar una breve descripción de cada caso de uso, en el cual se describe de forma breve la funcionalidad - UUCP = UAW + UUCW
4. **UCP – Puntos de caso de uso ajustados:** Resulta de ajustar el UUCP con factores técnicos (TCF) y ambientales (EF). UCP = UUCP x TCF x EF
5. **E - Esfuerzo horas-hombre**: Este cálculo se realiza con el fin de tener una aproximación del esfuerzo, pensando solo en el desarrollo según las funcionalidades de los casos de uso. Anteriormente, se sugería utilizar 20 horas/persona por UCP, ahora existen unas tablas que se particularizan dependiendo de la tarea (CF). E = UCP x CF

3. HERRAMIENTAS AUTOMÁTICAS DE ESTIMACIÓN

Las herramientas automáticas de estimación permiten al planificador estimar costos y esfuerzos, así como llevar a cabo análisis del tipo, qué pasa si, con importantes variables del proyecto, tales como la fecha de entrega o la selección del personal.

Estas herramientas requieren los siguientes tipos de datos :

- ✓ Una estimación cuantitativa del tamaño: en función de líneas de código fuente (LDC) o de puntos de función (PF).
- ✓ Características de la calidad del proyecto como la complejidad técnica.
- ✓ Descripción de los actores y el entorno.

Las más relevantes son:

- BYL, WICOMO y DECPlan: son herramientas automáticas de estimación basadas en COCOMO, requieren que el usuario proporcione estimaciones LDC preliminares y producen estimaciones de duración del proyecto, del esfuerzo, del personal medio por mes, de la productividad y el coste.
- SLIM: se basa en la curva de Rayleigh-Norden para el ciclo de vida del software y en el modelo de estimación de Putnam. Permite al planificador calibrar el entorno local de desarrollo interpretando datos históricos.
- ESTIMACS: macro estimación que utiliza el modelo de Puntos de Función mejorado para adaptarse a distintos tipos de proyectos. Permite estimar el esfuerzo de desarrollo del sistema, el coste y el personal, la configuración del HW y el riesgo, entre otros
- SPQR/20: consta de un conjunto de sencillas preguntas y respuestas que permiten especificar el tipo de proyecto, el tipo de aplicación, la innovación, el entorno de trabajo, los requisitos del programa y del diseño, documentación, tiempo de respuesta, experiencia del personal, porcentaje de reutilización del código, lenguaje de programación, complejidad de los algoritmos, etc.

4. RELACIÓN CON OTROS TEMAS

Se recomienda leer los resúmenes sobre planificación y gestión de proyectos disponibles en el pack.

Tema 33 de la convocatoria del 2018: Herramientas de planificación y control de gestión de la función del directivo de sistemas y tecnologías de la información en la Administración. El cuadro de mando

Tema 36 convocatoria del 2018: Metodologías predictivas para la gestión de proyectos: GANTT, PERT.

El estudio de esta materia está muy relacionado con algunas de las preguntas que suelen introducir en el tercer ejercicio (supuesto práctico), donde se suele pedir aplicar alguna metodología de estimación de las vistas anteriormente para estimar los recursos a aplicar en un determinado proyecto TIC. Conviene echar un vistazo a alguno de los volcados (supuestos prácticos corregidos) donde se pregunta por dicha estimación.