



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA



Departamento de Informática
Universidad Técnica Federico Santa María

Estimaciones

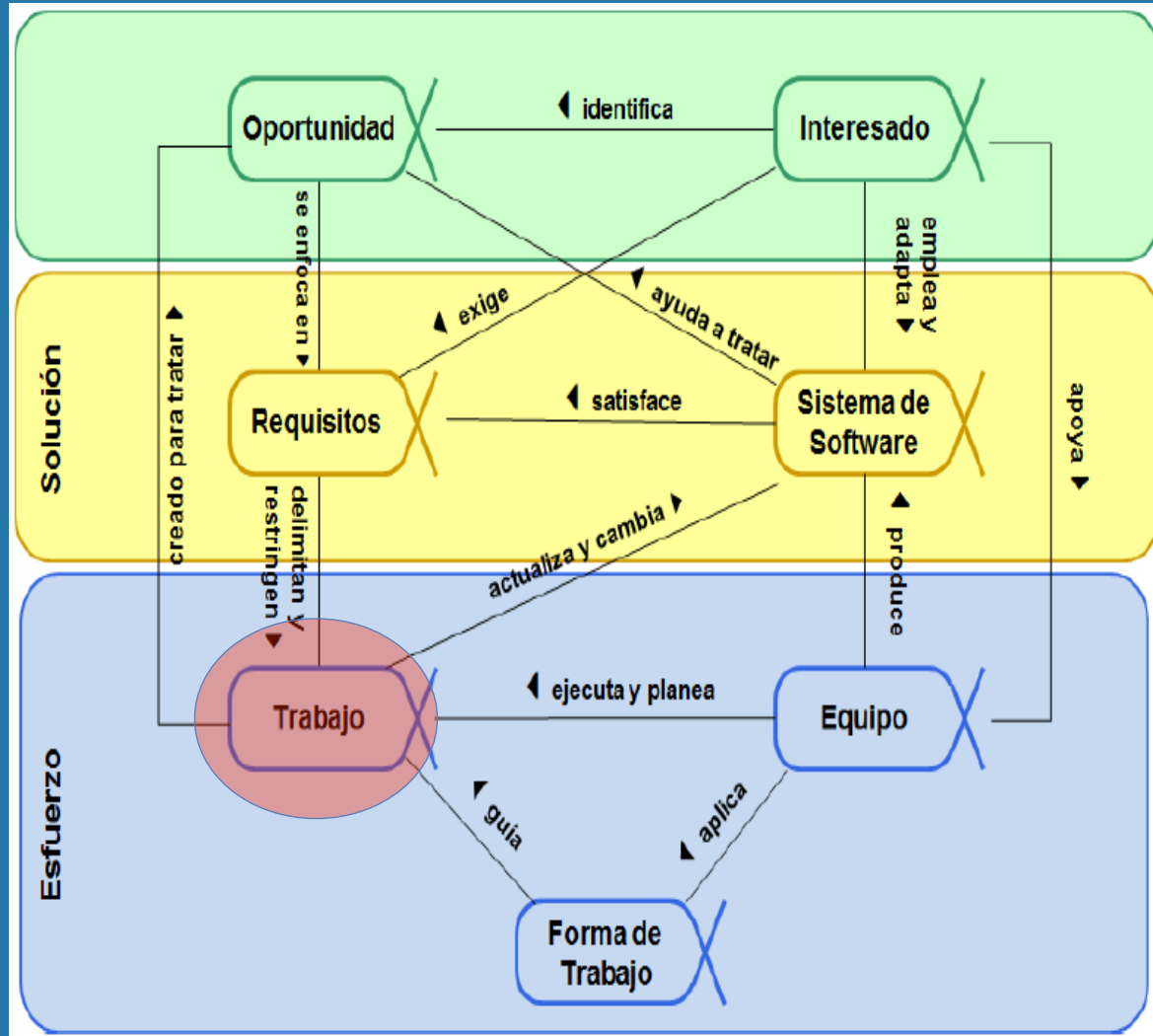
Ingeniería de Software

Hernán Astudillo & Cristian Orellana
Departamento de Informática
Universidad Técnica Federico Santa María

Contexto

- Partido de Chile
 - Se desea realizar un asado con los amig@s y, para lo anterior, se debe estimar la cuota por persona
 - ¿Se quiere comer carne? ¿Pollo? ¿Ensalada? ¿Otros?
 - ¿Posta paleta? ¿Asado carnicero? ¿Alitas de pollo? ¿Pechuga?
 - ¿Bebidas? ¿Cervezas? ¿Carbón?
 - Al parecer, para estimar la cuota por persona se deben considerar muchos factores
 - No es lo mismo colocar \$5.000 p/p que \$10.000
- En ISW ocurre lo mismo
 - Existen atributos y características que se deben considerar al momento de estimar
 - Con el objetivo de predecir (anticipar con cierto grado de certeza)

Contexto



Productividad de Software

Productividad de Software

[1]

- En general, entradas / salidas
- ¿Qué tipos de salidas?
 - ¿Lines of Codes (LOC)?
 - ¿Tokens?
 - ¿Function Points (FP)?
 - ¿Story Points?
 - ¿Orientación a Objetos?
 - ¿WebMet?
- ¿Qué tipos de entrada?
 - ¿\$?
 - ¿Esfuerzo?
- ¿Productividad?
 - ¿LOC/\$, LOC/HH, FP/HH, SP/\$?

Productividad de Software [2]

- Concepto no fácil de definir
 - Pero es relativamente fácil de mejorar
- Reducción del esfuerzo
 - Mejores habilidades, herramientas, métodos, estructuras organizacionales, planificación y control
 - “Necesitamos mejorar el proceso”
 - Hacer lo mismo o similar

Productividad de Software

[3]

- Factores que afectan la productividad
 - Humanos
 - Procesos
 - Producto
 - Tecnología
- Modelos de productividad
 - Objetivos
 - Generales
 - Significativos
 - Independientes

Productividad en estándares internacionales

- No existe un único modelos que integre la productividad completamente
 - [Cheikhi et al. 2012]

ISO 25010 (antes 9126)

[1]

- La *productividad* en la norma ISO se define como un atributo de calidad
 - Productividad: “capacidad del producto de software para habilitar a los usuarios a gastar los montos apropiados de recursos en relación a la efectividad lograda en un contexto específico de uso”
 - Eficacia: “capacidad del producto de software para habilitar a los usuarios para alcanzar los objetivos específicos con exactitud y completitud en un contexto específico”

ISO 25010 (antes 9126)

[2]

Quality characteristic	Derived Measures and purposes	Base Measures
Productivity	- Task time (How long does it take to complete a task?)	Task time
	- Task efficiency (How efficient are the users?)	Task effectiveness/Task time Task completion/ Task time
	- Economic productivity (How cost-effective is the user?)	Task effectiveness/Total cost of the task
	- Productive proportion (What proportion of the time is the user performing productive actions?)	Productive time (Task time - help time - error time - search time) / Task time
	- Relative user efficiency (How efficient is a user compared to an expert?)	Ordinary user's task efficiency/expert user's task efficiency

Effectiveness	- Task effectiveness (What proportion of the goals of the task is achieved correctly?)	
	- Task Completion (What proportion of the tasks is completed?)	Number of tasks completed/number of tasks attempted
	- Error frequency (What is the frequency of errors?)	Number of errors made by the user/time or number of task

IEEE std. 1045

- Objetivo: estandarizar la forma en que se *mide* la productividad y esfuerzo de software
- La productividad es expresada en términos de salida/entrada
- Estándar útil para medir la *productividad en artefactos de software producidos* como salida y *no para el uso* del producto de software

Estimación del esfuerzo

Estimación del esfuerzo

[1]

- Estimación: predicción de tiempo y costo requerido para completar u proyecto.
 - Arte v/s ciencia
- Costo de estimar mal
 - Subestimación: pérdida económica
 - Sobreestimación: baja competencia

Estimación del esfuerzo

[2]

- Enfoques de la estimación
 - Opiniones de expertos/juicios
 - Analogías
 - Descomposiciones
 - Modelos matemáticos
- Normalmente se usa una combinación de modelos, datos históricos, experiencia y sentido común

¿Qué es una “buena” estimación?

- Definición común: “una buena estimación es aquella que entrega resultados con un 25% de error el 75% de las veces”
- “Una buena estimación es aquella en donde el project manager tiene suficiente información para que pueda tomar las decisiones correctas con el objetivo de satisfacer necesidades”

Modelos de Estimación

- Criterios de calidad (de modelos)
 - Validación
 - Objetividad
 - Fácil de usar
 - Robusto
 - Transportable
- Tipos de modelos
 - Históricos
 - Estadísticos
 - Teóricos
 - Compuestos

Modelos históricos

- Modelos históricos o experimentales
 - Tipos de modelos más utilizados
 - Los expertos hacen sus juicios basados en
 - Experiencia
 - Intuición
 - Información histórica accesible

Modelos compuestos

- Combinan ecuaciones analíticas, ajustes estadísticos y juicios de expertos
- P.ej. COCOMO (COConstructive COst MOdel)

Modelos estadísticos

- Usan análisis estadísticos para determinar parámetros y sus relaciones (p.ej. regresión)
 - [Chulani et al. 1998] aplican análisis Bayesiano para la mejor predicción del rendimiento en el modelo COCOMO II (COCOMO = COConstructive COst MOdel)

Modelos teóricos [1]

- Están basados en teorías de cómo el ser humano trabaja durante el desarrollo de software
- A su vez, también están basados en leyes matemáticas que el proceso de desarrollo de software sigue
- P.ej. modelo Putnam

Modelos teóricos [2]

- Size= tamaño del producto. Puede ser LOC, ESLOC (Effective Source Lines of Code)

$$\frac{B^{1/3} \cdot \text{Size}}{\text{Productivity}} = \text{Effort}^{1/3} \cdot \text{Time}^{4/3}$$

- B= es un factor, una función del tamaño del proyecto que va desde 0.16 para productos con 5000 - 15000 LOC hasta 0.39 para productos con +70000 LOC
- Productivity= Proceso de productividad del software
- Effort= total de esfuerzo aplicado en el proyecto medido en personas/año
- Time= el tiempo requerido del proyecto en años

Causas de una mala estimación

- Frecuentes cambios en los requisitos
- Tarea omitidas
- Los usuarios no entienden completamente los requisitos
- Pobre comunicación entre analistas y usuarios
- Pobre o imprecisa definición del problema

Un enfoque para estimar

- Entrada: definición del problema
 - Paso 1: Estimación del tamaño (*Function Points*)
 - Paso 2: Transformación del tamaño (FP a KLOC)
 - Paso 3: Esfuerzo y programación de estimación (COCOMO)
 - Paso 4: Esfuerzo y programación de distribución de tareas (COCOMO)
 - Paso 5: Normalización del calendario de actividades (ESTERLING)
- Salida: Programación y costos

Relación entre planificación y estimación

- La estimación es la base de la planificación, pero ambos entregan diferente información
- En general, planificar para satisfacer ciertos objetivos es confundido con objetivos de estimación
- No separar estimación y planificación resulta en estimaciones no reales y malos planes

Comunicando planes y estimaciones

- Generalmente los stakeholders no se comunican correctamente
- En varias ocasiones, la estimación es requerida cuando la planificación realmente es solicitada
- No siempre se puede satisfacer los objetivos, debido a que algunas veces es necesarios clarificar cuáles pueden ser sacrificados (compromisos)

Estimación como probabilidad

- Error común: presentar la estimación como un simple valor numérico
- Si se desea mostrar una simple estimación, se debe estar seguro de que es realmente una estimación y no un objetivo
- Una estimación presentada de la siguiente forma “18-24 semanas” es más útil

COCOMO

COCOMO

- El modelo constructivo de costos (COCOMO) es utilizado en proyecto de software para estimar los costos en función de tres sub-modelos: básico, intermedio y avanzado
 - Básico: estima el costo del proyecto (pequeño-mediano) en función de número de líneas de código estimadas
 - Intermedio: se utiliza para estimaciones más complejas. Éste incluye 15 atributos (dentro de 4 categorías) del software para determinar el costo del proyecto
 - Avanzado: incorpora las características del modelo intermedio y se analiza cada paso del proceso de ingeniería de software

COCOMO - Atributos

(1) Atributos del producto

- RELY: garantía de funcionamiento requerida al software
- DATA: tamaño de la base de datos
- CPLX: complejidad del producto

(2) Atributos de la máquina

- TIME: restricción de tiempo de ejecución
- STOR: restricción del almacenamiento principal
- VIRT: volatilidad de la máquina virtual
- TURN: tiempo de respuesta del ordenador

(3) Atributos del personal

- ACAP: capacidad del analista
- AEXP: experiencia en la aplicación
- PCAP: capacidad del programador
- VEXP: experiencia en máquina virtual
- LEXP: experiencia en lenguaje de programación

(4) Atributos del proyecto

- MODP: prácticas de programación modernas
- TOOL: utilización de herramientas software
- SCED: plan de desarrollo requerido.

COCOMO

- Por otro lado, COCOMO define tres modos de desarrollo o tipos de proyectos:
 - Orgánico: proyectos relativamente sencillos, menores de 50 KDLC líneas de código, en los cuales se tiene experiencia de proyectos similares y se encuentran en entornos estables
 - Semi-acoplado: proyectos intermedios en complejidad y tamaño (menores de 300 KDLC), donde la experiencia en este tipo de proyectos es variable, y las restricciones intermedias
 - Empotrado: proyectos bastante complejos, en los que apenas se tiene experiencia y se engloban en un entorno de gran innovación técnica. Además se trabaja con unos requisitos muy restrictivos y de gran volatilidad

Ejemplo COCOMO


- Proyecto: Gestión de Inventarios
- Para este proyecto, se usará el modelo intermedio, dado que realiza las estimaciones con bastante precisión.
- Luego las formulas serán las siguiente
 - $E = \text{Esfuerzo} = a \cdot \text{KLDC} \cdot \text{FAE}$ (persona x meses)
 - $T = \text{Tiempo de duración del desarrollo} = c \cdot \text{Esfuerzo}$ (meses)
 - $P = \text{Personal} = E/T$ (personas)

Ejemplo

- Para calcular el Esfuerzo, se necesita hallar la variable KDLC (Kilo-líneas de código), donde los PF son 261,36 (dato conocido) y las líneas por cada PF equivalen a 32 (Visual Basic, dato conocido)
- Luego, si se sabe que son 32 LDC por cada PF, por el hecho de ser Visual Basic el resultado de los KDLC será el siguiente:
 - $KLDC = (PF \cdot \text{Líneas de código por cada PF}) / 1000 = (261,36 \cdot 32) / 1000 = 8,363 \text{ KDLC}$

Ejemplo

- Dado lo anterior, para este caso el tipo de proyecto orgánico será el más apropiado, ya que el número de líneas de código no supera los 50 KLDC, y además el proyecto no es muy complejo. Entonces, los coeficientes que usarán serán los siguientes:



PROYECTO SOFTWARE	a	e	c	d
Orgánico	3,2	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,20	2,5	0,32

Ejemplo

- Y por otro lado, se debe obtener la variable FAE, la cual se obtiene mediante la multiplicación de los valores evaluados en los diferentes 15 atributos.
- En la siguiente slide se mostrará la tabla

Ejemplo

- $F\Delta E = 1,15 * 1,00 * 0,85 * 1,11 * 1,00 * 1,00 * 1,07 * 0,86 * 0,82 * 0,70 * 1,00 * 0,95 * 1,00 * 0,91 * 1,08 = 0,53508480$

CONDUCTORES DE COSTE	VALORACIÓN					
	<i>Muy bajo</i>	<i>Bajo</i>	<i>Nominal</i>	<i>Alto</i>	<i>Muy alto</i>	<i>Extr. alto</i>
Fiabilidad requerida del software	0,75	0,88	1.00	1,15	1,40	-
Tamaño de la base de datos	-	0,94	1.00	1,08	1,16	-
Complejidad del producto	0,70	0,85	1.00	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	1.00	1,11	1,30	1,66
Restricciones del almacenamiento principal	-	-	1.00	1,06	1,21	1,56
Volatilidad de la máquina virtual	-	0,87	1.00	1,15	1,30	-
Tiempo de respuesta del ordenador	-	0,87	1.00	1,07	1,15	-
Capacidad del analista	1,46	1,19	1.00	0,86	0,71	-
Experiencia en la aplicación	1,29	1,13	1.00	0,91	0,82	-
Capacidad de los programadores	1,42	1,17	1.00	0,86	0,70	-
Experiencia en S.O. utilizado	1,21	1,10	1.00	0,90	-	-
Experiencia en el lenguaje de programación	1,14	1,07	1.00	0,95	-	-
Prácticas de programación modernas	1,24	1,10	1.00	0,91	0,82	-
Utilización de herramientas software	1,24	1,10	1.00	0,91	0,83	-
Limitaciones de planificación del proyecto	1,23	1,08	1.00	1,04	1,10	-

Ejemplo

- Justificación
 - Atributos de producto
 - Fiabilidad requerida del software: Si se produce un fallo por el pago de un pedido, o fallo en alguna reserva, etc... puede ocasionar grandes pérdidas a la empresa (Valoración Alta).
 - Tamaño de la base de datos: La base de datos del producto será de tipo estándar (Valoración Nominal).
 - Complejidad del producto: La aplicación no va a realizar cálculos complejos (Valoración Baja).

Ejemplo

- Justificación
 - Atributos de la máquina
 - Restricciones del tiempo de ejecución: En los requerimientos se exige alto rendimiento (Valoración Alta)
 - Restricciones del almacenamiento principal: No hay restricciones al respecto (Valoración Nominal)
 - Volatilidad de la máquina virtual: Se usarán sistemas de la “Familia Windows” (Valoración Nominal)
 - Tiempo de respuesta de la máquina: Deberá ser interactivo con el usuario (Valoración Alta)

Ejemplo

- Justificación
 - Atributos del personal
 - Capacidad del analista: Capacidad alta relativamente, debido a la experiencia en análisis en proyecto similar (Valoración Alta)
 - Experiencia en la aplicación: Se tiene cierta experiencia en aplicaciones de esta envergadura (Valoración muy alta)
 - Capacidad de los programadores: Teóricamente deberá tenerse una capacidad muy alta por la experiencia en anteriores proyectos similares (Valoración muy alta)
 - Experiencia en S.O. utilizado: Con Windows 8 Professional la experiencia es a nivel usuario (Valoración Nominal)
 - Experiencia en el lenguaje de programación: Es relativamente alta, dado que se controlan las nociones básicas y las propias del proyecto (Valoración Alta)

Ejemplo

- Justificación
 - Atributos del proyecto
 - Prácticas de programación modernas: Se usarán prácticas de programación mayormente convencional (Valoración Nominal)
 - Utilización de herramientas software: Se usarán herramientas estándar que no exigirán mucha formación, de las cuales se tiene cierta experiencia (Valoración Alta)
 - Limitaciones de planificación del proyecto: Existen pocos límites de planificación. (Valoración Baja)

Ejemplo

- Cálculo del esfuerzo del desarrollo
- $E = a \cdot KLDC \cdot FAE = 3,25 \cdot (8.363) \cdot 0,53508480 = 15,91$ personas /mes
- Cálculo del tiempo de desarrollo
- $T = c \cdot \text{Esfuerzo}^d = 2,5 \cdot (15,91)^{0,38} = 7,15$ meses
- Productividad
- $PR = LDC/\text{Esfuerzo} = 8363/15,91 = 525,64$ LDC/ personas mes
- Personal promedio
- $P = E/T = 15,91/7,15 = 2,22$ personas

Ejemplo

- Según estas cifras será necesario un equipo de 3 personas trabajando alrededor de 7 meses. Pero, debido a que el desarrollo del proyecto debe realizarse en un plazo 3 meses, incrementaremos a 6 personas el número de personas del equipo de proyecto (ya que $15,91/3$ nos da alrededor de este resultado).
- Así pues tendremos un equipo formado por 1 Jefe de proyecto, 2 Analistas, 2 programadores y 1 Responsable de calidad.

COCOMO II [1]

- Este modelo permite realizar estimaciones en función del tamaño del software, y de un conjunto de factores de costo y de escala.
- Los factores de costo describen aspectos relacionados con la naturaleza del producto, hardware utilizado, personal involucrado, y características propias del proyecto.
- El conjunto de factores de escala explica las economías y deseconomías de escala producidas a medida que un proyecto de software incrementa su tamaño.

COCOMO II [2]

- COCOMO II (http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html) posee tres modelos denominados Composición de Aplicación, Diseño Temprano y Post-Arquitectura.
- Cada uno de ellos orientados a sectores específicos del mercado de desarrollo de software y a las distintas etapas del desarrollo de software.

Agile COCOMO II

- Agile COCOMO II (<http://csse.usc.edu/csse/research/AgileCOCOMO/>)
 - Facilita la exactitud comparativa de estimación de costo de proyectos de software
 - Fue desarrollado para reducir la complejidad de la estimación de costos usando COCOMO II mientras provee información exacta de estimación de costo y esfuerzo

Buenas prácticas

Consejos para una buena estimación

- Utilizar más de una persona para estimar
 - Esto brindará diferentes perspectivas para poder estimar
- Utilizar más de un enfoque
 - Utilizar más de una técnica de estimación permitirá converger en una estimación final
- Saber cuando parar
 - La estimación no es una ciencia exacta
- Presentar las estimaciones como rango
- Describir los rangos como probabilidades



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA



Departamento de Informática
Universidad Técnica Federico Santa María

Estimaciones

Ingeniería de Software

Hernán Astudillo & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María