Universidad Complutense

Facultad de Informática

eGorilla

Documento de Estimación

Asignatura: *Ingeniería del Software*

Curso Académico: *2008/2009*

Grupo: *4º B*

Índice

1. Introducción.

A la hora de desarrollar un sistema de información realizaremos las siguientes tareas básicas:

-Desarrollo (Análisis, diseño, codificación, pruebas, mantenimiento,…)

-Gestión (Estimación, planificación, seguimiento y control)

Dentro de las actividades de gestión hay tres actividades fundamentales. Las actividades de estimación que tienen que ver con el coste, esfuerzo, recursos, etc.

Una vez obtenida la estimación del proyecto podremos realizar la planificación de nuestro proyecto.

La planificación de proyecto será un proceso de selección de una estrategia para la obtención de unos productos finales así como la definición de las actividades a realizar, para conseguir esos objetivos teniendo en cuenta la posible concurrencia y solapamiento de las actividades y la asignación de recursos a los mismos

Las actividades de seguimiento y control engloban aquellas actividades para llevar a cabo una buena configuración del sistema.

En este documento realizaremos la primera actividad de gestión que es la estimación del proyecto. Utilizaremos modelos empíricos de estimación que producen sus ecuaciones de estimación deducidas del estudio de un conjunto de proyectos terminados. En ellos se aplican una serie de ecuaciones matemáticas como:

$$\text{Esfuerzo de desarrollo} = a * (KSDI^{b}) \rightarrow \text{COCOMO}$$

Este método proporciona unas tablas donde se dice como se distribuye el esfuerzo y el tiempo entre las actividades del proyecto.

Concretamente utilizaremos el método COCOMO II post-arquitectura.

Para la estimación del coste del proyecto utilizando el método COCOMO II, lo hemos realizado utilizando los PUNTOS DE FUNCIÓN.

2. Puntos de función.

Los puntos de función es una técnica de estimación que trata de medir el tamaño de un desarrollo en términos de funcionalidad proporcionada al usuario final, el objetivo es proporcionar una medida del tamaño de un sistema de manera que otras métricas puedan ser expresadas en términos de ratios. Con los puntos de función no medimos el esfuerzo, es una medida de tamaño.

Lo primero que obtenemos es la cuenta de los tipos de función y los DET (Data Element Type), RET (Record Element Type) y FTR (File Type Referenced) de estos tipos de función y con ellos obtendremos la complejidad.

2.1. Tipos de función de datos.

Ficheros lógicos internos (ILF)

Éstos serán datos que están relacionados lógicamente o información de control mantenida dentro de los límites de la aplicación.

Parámetros de conexión→(Num_descargas_sim, Lim_subida, Lim_bajada, Puerto) DET=4, RET=1

Complejidad: BAJA

Parámetros de disco→(Dir_Llegada, Dir_Compartidos)

DET=2, RET=1

Complejidad: BAJA

Parámetros de Servidor→(IpServidor, PuertoServidor,NombreServidor, Descripcion)

DET=4, RET=1

Complejidad: BAJA

Parámetros de usuario→(NmbUsuario)

DET=1

Complejidad: BAJA

cliente.properties→ (nombre de propiedad, valor de la propiedad)

DET=2, RET=1

Complejidad: BAJA

cliente_default.properties→(nombre de propiedad_defecto, valor de propiedad_defecto)

DET=2, RET=1

Cliente_Archivo→nombre_fichero, hash_archivo

DET=2, RET=1

Complejidad: BAJA

Ficheros de interfaz externos (EIF)

Serán grupos de datos identificables por el usuario, o información de control.

2.2. Tipos de función de procedimientos

Entradas externas (EI)

Aquí tenemos información de control que procede de fuera de los límites de la aplicación. Podría ser información del usuario o de otra aplicación.

PeticionBusqueda -> 1 DET (cadena de busqueda) 1 FRT (lista de archivos)

Complejidad: BAJA

PeticionDescarga -> 1 DET (hash de fichero) 2 FRT (los dos Ficheros lógicos que hay en servidor, lista de clientes y lista de archivos)

Complejidad: BAJA

Propiedades del cliente:

Parámetros de conexión→(Num descargas simultáneas, Limite velocidad subida, Limite velocidad bajada, Puerto)

DET =4

Parámetros de disco→(Directorio de Llegada, Directorio de Compartidos)

DET=2

Parámetros de servidor→(IpServidor, PuertoServidor, NombreServidor, Descripcion)

DET=4

Parámetros de usuario→(NmbUsuario)

DET=1

El ILF cliente.properties sobre el que se realiza el mantenimiento y el ILF cliente_default.properties que solamente se lee (por si el cliente desea restaurar las propiedades por defecto).

FTR=2

TOTAL: DET=11 FTR=2

Complejidad: MEDIA

Salidas externas (EO)

Será información de control que sale de los límites de la aplicación mediante un proceso lógico (cálculo). No contabilizará la ayuda o los mensajes de error.

Consultas externas (EQ)

Obtiene una combinación de entrada/salida como resultado de una recuperación de datos.

Un ejemplo de consultas externas es cuando el usuario busca un fichero.

Solicitud_Cliente = fichero_buscado

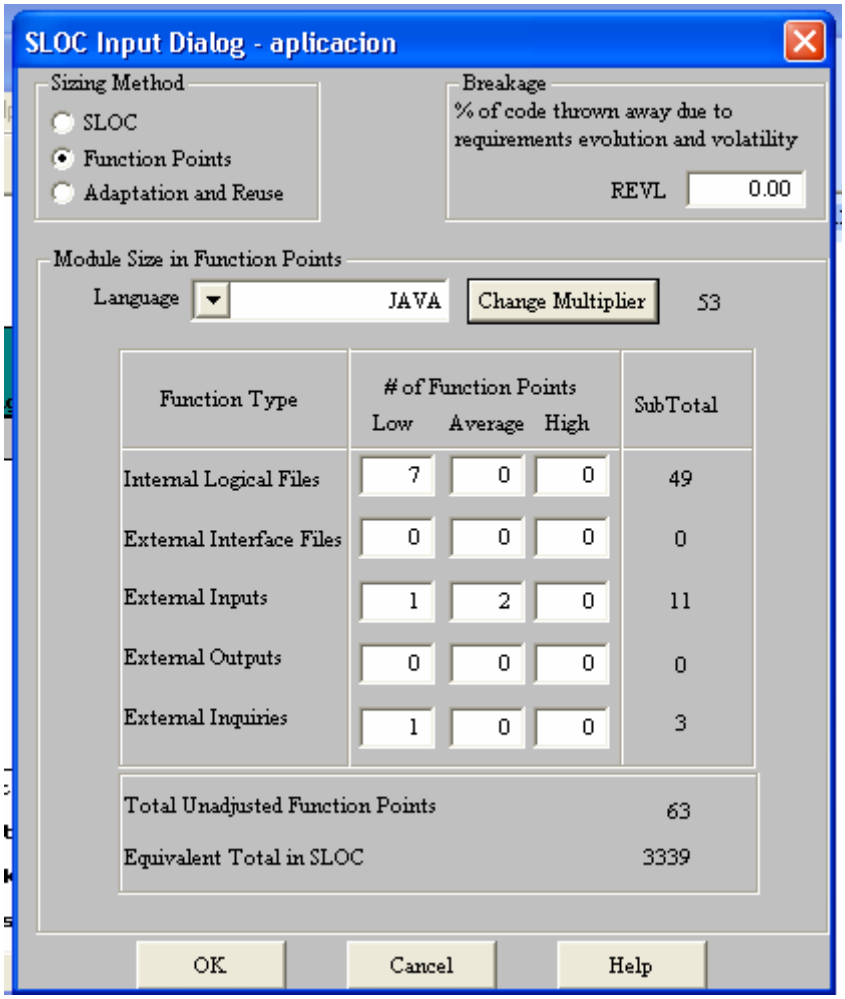
DET = 1 FTR = 1

Complejidad = BAJA

2.3. Puntos de función sin ajustar.

Con la información anterior obtenemos los puntos de función sin ajustar realizando el siguiente cálculo:

PARAMETRO	COMPLEJIDAD	*	PESO		TOTAL	
ENTRADA	ALTA	*	6	=		
	MEDIA (1)	*	5	=	5	
	BAJA (2)	*	3	=	6	
SALIDA	ALTA	*	7	=		
	MEDIA	*	5	=		
	BAJA	*	4	=		
CONSULTA	ALTA	*	6	=		
	MEDIA	*	4	=		
	BAJA (1)	*	3	=	3	
FICH. LOG. INT	ALTA	*	15	=		
	MEDIA	*	10	=		
	BAJA (7)	*	7	=	49	
FICH. INTERF	ALTA	*	10	=		
	MEDIA	*	7	=		
	BAJA	*	5	=		
TOTAL					=	63



Cálculo de los puntos de aplicación mediante herramienta software.

3. COCOMO II.

3.1. Cálculo del $MM_{nominal}$

$MM_{nominal}$= A x (Tamaño)^{B} .Siendo:

Tamaño=53 (java) X 63 = 3339 LOC. En miles de líneas de código =3,339

A=3,0.

B=factor de escala.

3.1.1. Cálculo del factor de escala.

Drivers de Escala.

Precedentes (PREC):

En gran parte sin precedentes.

Bajo - 3,24.

Flexibilidad en el Desarrollo (FLEX):

Alguna relajación.

Nominal – 3,64.

Resolución de Riesgos (RESL):

Alguna.

Bajo – 3,38.

Cohesión del Equipo (TEAM):

Interacciones cooperativas básicas.

Nominal – 2,73.

Madurez del Proceso (PMAT):

Nominal – 2,97.

$B=0,91 + 0,01 \Sigma W_i = 0,91 + 0,01(3,24 + 3,64 + 3,38 + 2,97 + 2,73)=1,06$

$MM_{nominal}$= A x (Tamaño)^{B} = 10,63

3.2. Cálculo del $MM_{ajustado}$ .Post-Arquitectura.

$MM_{ajustado} = MM_{nominal} X (\Pi_{\iota}^{17} EM_i)$.

3.2.1. Drivers de Coste.

1. Fiabilidad requerida del software (RELY):
   Fiabilidad moderada.
   Nominal – 1,00
2. Tamaño de la base de datos (DATA):
   No utilizamos base de datos por tanto:
   Nominal – 1,00
3. Complejidad del producto (CPLX):
   Dividida en 5 áreas.
   Funcionamiento de control
   Muy alta
   Funcionamiento computacional
   Nominal
   Funcionamiento de Dispositivos dependientes
   Alta
   Funcionamiento del sector de datos
   Nominal.
   Funcionamiento del GUI
   Bajo
   La media es:
   Alta – 1,17
4. Reusabilidad requerida (RUSE):
   Reutilización en el mismo proyecto
   Nominal – 1,00

5. Documentación de acuerdo a las necesidades del ciclo de vida (DOCU):

Necesidades cubiertas para el correcto desarrollo del ciclo de vida del software.

Nominal – 1,00

6. Restricción de tiempo (TIME):

   Menos del 50% por tanto valor nominal.

   Nominal – 1,00

7. Restricción de almacenamiento principal (STOR):

   Menos del 50% por tanto valor nominal.

   Nominal – 1,00

8. Volatilidad de plataforma (PVOL):

   La volatilidad de la plataforma es baja, donde el termino plataforma se usa para referirnos a la complejidad del software y hardware que el producto necesita para realizar sus tareas.

   Muy bajo pero como la tabla no tiene valor para este parámetro, utilizaremos como tal, bajo.

   Bajo - 0,87

9. Capacidad de analistas (ACAP):

   La habilidad de análisis y diseño, la eficiencia y minuciosidad y la habilidad para comunicar y cooperar, de los miembros del equipo es considerada Baja.

   Baja – 1,19.

10. Capacidad de programadores (PCAP):

    La capacidad de los programadores como equipo se considera

    Nominal – 1,00

11. Continuidad del personal (PCON):

    La continuidad del personal de proyecto ha sido baja. Hemos tenido 3 bajas y 2 incorporaciones (33% del personal)

    Baja – 1,12

12. Experiencia en aplicaciones (AEXP):

    El nivel de experiencia del equipo de desarrollo, en sistemas software de similar complejidad se puede considerar bajo (6 meses).

    Baja – 1,10

13. Experiencia de plataforma (PEXP):

La experiencia en este tipo de plataformas muy baja (menos de 2 meses).

Muy baja – 1,19.

14. Experiencia de lenguajes y herramientas (LTEX):

La experiencia con las herramientas software de la aplicación es alta (3 años desarrollando en java).

Alta – 0,91.

15. Uso de herramientas de software (TOOL):

La herramienta utilizada, en nuestro caso netbeans, posee herramientas de diseño y mantenimiento del ciclo de vida de la aplicación, por tanto la capacidad es muy alta.

Muy Alta - 0,78

16. Desarrollo en múltiples lugares (SITE):

Se mide en dos factores:

-Localización del lugar:

Mismo edificio o complejo, en nuestro caso la Facultad de Informática.

Muy alta

-Soporte de comunicación:

Mediante email por tanto un valor nominal.

Nominal

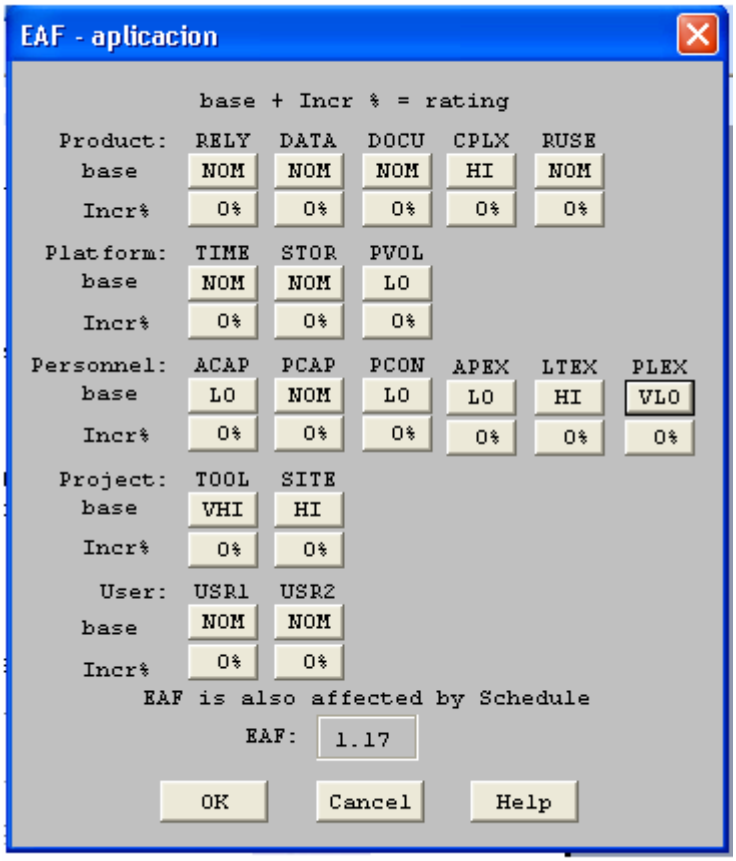
La media se queda por tanto en un valor alto

Alto – 0,93

17. Esfuerzo de calendario (SCED):

Los cambios en cuanto a ampliación o disminución del calendario de entrega del proyecto, respecto a un calendario nominal son de coste nominal. Las fechas se han mantenido por tanto valor nominal.

Nominal – 1,00



$MM_{ajustado} = MM_{nominal}$ X $(\Pi_{i}^{17} EM_i)$.=10,63 X 1,17 = 12,44 personas X mes.

Si el mes tiene 160 horas de trabajo y partimos de que nosotros trabajamos 2 horas diarias , nuestro mes tiene 40 horas por lo tanto nuestro MM es 50 personas X mes.

4. Métricas.

4.1. Productividad

La productividad mide el número de líneas de código que cada miembro de desarrollo produce por mes. Gracias a este dato podremos ver el rendimiento del grupo de desarrollo y tomar medidas a llevar a cabo para aumentar el rendimiento de la plantilla.

La métrica de productividad se calcula como Nº líneas de código (LOC)/esfuerzo (personas X mes).

En nuestro caso somos 12 personas y el desarrollo del proyecto se ha realizado en 8 meses (2 cuatrimestres), por tanto nuestro esfuerzo es 12 personas X 8 meses, lo que equivale a *96 personas X mes*.

Productividad = 3339 LOC/ 96 personas X mes =**34,7 LOC / persona X mes.**

4.2. Fiabilidad

Lo mediremos en número de errores en tiempo de ejecución / hora, nos dará una medida de cómo es de fiable nuestra aplicación.

Nos permitirá comprobar el posible desarrollo positivo de la aplicación en cuanto a que la fiabilidad vaya aumentando.

Esperamos que en la entrega final, la fiabilidad, sea alta.

En este momento la fiabilidad es baja.

4.3. Número de páginas de documentación/líneas de código.

Con este dato podremos medir si hemos desarrollado suficiente documentación en el desarrollo de la aplicación. Un valor bajo, que indique que el número de páginas de documentación por líneas de código sea bajo nos dará una idea de la cantidad necesaria de documentación a generar.

Nº de paginas / 100 LOC= 258 páginas/33,39 = 7,7 páginas por cada 100 líneas de código.

4.4. Número de errores detectados en las pruebas previas a la entrega.

Esta métrica nos será de utilidad para contemplar si nuestro desarrollo ha dado lugar a muchos errores, para así poder subsanarlos antes de que se conviertan en defectos de la aplicación.

Esta métrica la podremos calcular cuando hagamos la primera y segunda batería de pruebas.

4.5. Número de líneas de código con factor de ajuste.

Esta métrica nos permite ver el número de líneas de código de la aplicación, esta medida nos permitirá hacer comparaciones con otras aplicaciones similares para poder sacar conclusiones .

4.5.1. Cálculo del factor de ajuste.

Para calcular el factor de ajuste, primero determinaremos el grado de influencia asignando valores entre (0-5), según la respuesta que ofrece nuestro sistema a cada una de las siguientes 14 preguntas:

Nota: El valor asignado entre 0 y 5 indicará lo siguiente:

0 Sin Influencia

1 Influencia Mínima

2 Influencia Moderada

3 Influencia Apreciable

4 Influencia Significativa

5 Influencia Muy Fuerte

1. Comunicación de datos

Esta parte es crítica para nuestro sistema pero utilizamos un solo protocolo de comunicación por lo que lo valoramos con 4.

2. Funciones distribuidas

En nuestra aplicación existe un procesamiento distribuido on-line en ambas direcciones, por lo que la influencia es 4

3. Rendimiento

Se definen requisitos de rendimiento, pero no se definen acciones especiales. El grado del factor es 1

4. Configuraciones fuertemente utilizadas

Existen restricciones operativas. El grado del factor es 1

5. Frecuencia de transacciones

No existe definición de periodo punta 0.

6. Entrada de datos *on-line*

De acuerdo con lo descrito. El factor es 5.

7. Eficiencia del usuario final

De cara al usuario final no se ha dado una gran importancia a todo lo relacionado con la interfaz, scrolling, ventanas, ayuda aunque sí que se ha tenido en cuenta, por lo que damos un factor 2.

8. Actualización *on-line de los ILF*

Actualización *on-line* de todos los archivos lógicos internos es importante, pero no se ha realizado un diseño especialmente contra la perdida de datos por lo que el valor de factor asignado es de 4.

9. Procesamiento complejo

La aplicación no realiza procesamiento excesivamente complejo de cara por ejemplo, a la optimización de las entradas de datos de diferentes usuarios por lo que el factor asignado es de 0.

10. Reusabilidad

No hay excesiva preocupación en cuanto a la reusabilidad del código por lo que el factor es 0.

11. Facilidad de implementación

No se hecho ninguna consideración con vistas a facilitar la implementación o instalación de la aplicación. Factor 0

12. Facilidad de operación

No se ha hecho ninguna consideración específica. Factor 0

13. Instalación en distintos lugares

La aplicación al estar hecha en Java, esta pensada para que se pueda instalar en múltiples entornos con distinto software (Linux, Windows),

incluso diferente hardware ,por lo que el factor asignado es 3.

14. Facilidad de cambios

Se valora con un 0 porque el usuario no mantiene tablas ni realiza consultas sobre ellas. Factor 0.

La suma de todas las características determina el nivel de influencia y es 26, por lo que:

Factor de ajuste = (nivel de influencia * 0.01) + 0.65

Factor de ajuste = (24 * 0.01) + 0.65

Factor de ajuste = 0.89

4.5.2. Cálculo del número de líneas de código

Con el factor de ajuste y los puntos de función podemos calcular el valor de los puntos de función ajustados

Puntos de Función del Proyecto (PFP) = Factor de Ajuste (AF) * (PF)

PFP = 0.89 * 63=39,69.

Número de líneas de código= PFP * 53 (java) = 2103 LOC

5. Análisis de resultados.

Resultado:

$MM_{ajustado} = MM_{nominal} X (\Pi_{i}^{17} EM_i).=10,63 X 1,17 = 12,44$ personas X mes.

Conclusiones:

Si el mes tiene 160 horas de trabajo y partimos de que nosotros trabajamos 2 horas diarias , nuestro mes tiene 40 horas por lo tanto nuestro MM es 50 personas X mes.

A la vista de los resultados obtenidos podemos ver, que teniendo en cuenta, que el desarrollo del proyecto se ha realizado en un transcurso de tiempo de 8 meses, el número de miembros necesarios para la realización del mismo, es de aproximadamente según el método de COCOMO II de 6 miembros.

Existe un desfase de 6 personas del número de integrantes reales, con respecto a la estimación.

Este desfase se debe a que además de desarrollar software, también generamos documentación, tales como *especificación de requisitos, gestión de riesgos, diseño del sistema,…*lo cual resta un número significativo de personal desarrollando en determinados momentos del proceso de elaboración de la aplicación, este parámetro es tenido en cuenta en COCOMO II, pero con poca influencia en el factor de escala, además hemos generado documentación que no hemos programado.

También hemos de tener en cuenta que existen periodos en los que el desarrollo se ha visto paralizado por dedicación de la plantilla al estudio, para la realización de los exámenes de febrero o realización de otras prácticas.

Otro factor que ha influido ha sido la gran cantidad de líneas programadas que no se han usado en la aplicación final pero que han sido parte del desarrollo de la misma.