Métricas de productividad de software para la gestión de proyectos

Consultora Ana Cristina Domingo Troncho
Alumno Javier Luque Claveras
Asignatura TFC-Gestión de proyectos
Fecha 13/01/2015

INDICE DE CONTENIDO

Contenido

1. INTRODUCCION

1.1 Justificación del TFC

El sector informático se caracteriza por el cambio constante, por un estado de progreso, perfeccionamiento y adaptación a nuestras necesidades. De ahí que muchos proyectos de esta área no se finalicen como se esperaba debido a una planificación improvisada, no hay tiempo para su preparación, y se excede en tiempo de ejecución debido a una mal ejecución de los diferentes puntos a realizar o a la ausencia en la planificación inicial de puntos necesarios que no se tuvieron en cuenta.

En la preparación de todo proyecto existe una estimación de costes de los diferentes puntos a realizar. Las métricas del software pueden ser de: productividad, calidad, técnicas, orientadas al tamaño, orientadas a la función u orientadas a la persona. Este documento tratará sobre las métricas del software, que se centran en el rendimiento del proceso de la ingeniería del software.

1.2 Objetivos del TFC

Iniciaremos el trabajo con una breve introducción a los conceptos más básicos relacionados con la medición de un proyecto de *software*: medida, métrica, indicador y atributos, y un listado y definición de los factores más importantes a la hora de diseñar aplicar una métrica.

Serán analizadas las diferentes métricas existentes (proceso, proyecto y software), centrándonos preferentemente en las métricas de productividad. En este caso se analizarán los diferentes aspectos que se pueden tener en cuenta como el tamaño, las de función, de puntos características y puntos objeto. Se estudiarán también los diferentes estándares oficiales.

Finalizaremos con la presentación del modelo COCOMO, de los primeros que se crearon y que continua vigente manteniéndose actualizado según los requisitos que la sociedad moderna necesita y requiere.

1.3 Enfoque y método a seguir

Para la realización del proyecto se ha decidido por un desarrollo en cascada. Para ello se requiere de una dedicación diaria de trabajo y la elaboración de una planificación detallada de trabajo, indicando que actividades se van a realizar y en qué orden.

1.4 Planificación del proyecto

La fecha de inicio del proyecto será el 13 de Octubre y su fecha fin el día de su entrega (el 13 de Enero). Debido al tamaño del proyecto la dedicación será diaria, siguiendo punto por punto el siguiente listado de actividades a realizar:

TAREA	DURACIÓN	COMIENZO	FIN
Definición del alcance del proyecto	81 días	13/10/2014	13/01/2015
Punto 1: Introducción	9 días	13/10/2014	21/10/2014
Justificación del TFC	2 días	13/10/2014	14/10/2014
Objetivos del TFC	2 días	15/10/2014	16/10/2014
Enfoque y método a seguir	1,5 días	17/10/2014	18/10/2014
Planificación del proyecto	1,5 días	18/10/2014	19/10/2014
Productos obtenidos	1 dia	20/10/2014	20/10/2014
Descripción de la memoria	1 dia	21/10/2014	21/10/2014
Punto 2: Conceptos Básicos	12 días	22/10/2014	11/11/2014
Mesures	3 días	22/10/2014	24/10/2014
Métricas	3 días	25/10/2014	27/10/2014
Indicadores	2 días	28/10/2014	29/10/2014
Atributos internos y atributos externos	2 días	30/10/2014	31/10/2014
Factores a tener en cuenta	2 días	01/11/2014	02/11/2014
Punto 3: Métricas	9 días	03/11/2014	11/11/2014
Métricas de proceso	3 días	03/11/2014	05/11/2014
Métricas del proyecto	3 días	06/11/2014	08/10/2014
Métricas del software	3 días	09/11/2014	11/11/2014
Entrega PAC 2			
Punto 4: Métricas de Productividad	21 días	12/11/2014	09/12/2014
Definición y características	2 días	12/11/2014	13/11/2014
Utilización	2 días	14/11/2014	15/11/2014
Recursos	2 días	16/11/2014	17/11/2014
Métricas orientadas al tamaño	3 días	18/11/2014	20/11/2014
Métricas de puntos de función (Albretch)	3 días	21/11/2014	23/11/2014
Relación tamaño y puntos de función	3 días	24/11/2014	26/11/2014
Métricas de puntos características (MKII)	3 días	27/11/2014	29/11/2014
Puntos objeto	3 días	30/11/2014	02/12/2014
Punto 5: Estándares para la medición	7 días	03/12/2014	09/12/2014
Método Objetivo-Pregunta-Métrica (GQM)	2 días	03/12/2014	04/12/2014
Métricas de calidad (IEEE 1061-1998)	2 días	05/12/2014	06/12/2014
PSM y el estándar ISO/IEC 15939	2 días	07/12/2014	08/12/2014
Otros	1 dia	09/12/2014	09/12/2014
Entrega PAC 3			
Punto 6: Métricas de recursos (COCOMO)	33 días	10/12/2014	13/01/2015
Definición y caracteristicas	3 días	10/12/2014	12/12/2014
Modelos de estimación	3 días	13/12/2014	15/12/2014
Punto 7: Conclusiones	10 días	15/12/2014	24/12/2014
Punto 8: Glosario	10 días	25/12/2014	03/12/2014
Punto 9: Bibliografía	10 días	04/12/2014	13/01/2015
Entrega final			

Figura 1: calendario del proyecto

Para la ejecución del TFC se propone seguir el siguiente diagrama de Gantt:

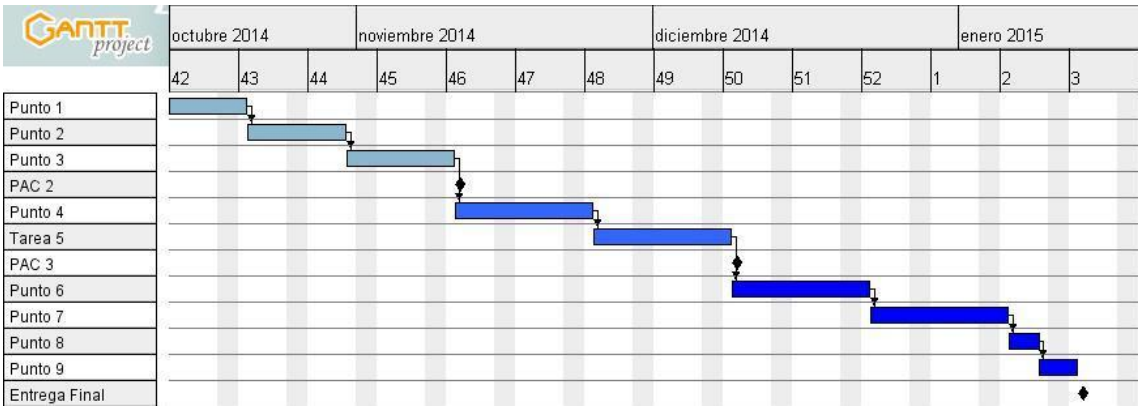

Figura 2: diagrama de Gantt

1.5 Productos obtenidos

Como resultado de este proyecto se obtendrá la presente memoria.

1.6 Descripción de la memoria

Este TFC se ha dividido en 9 puntos principales. A continuación comentaremos brevemente cada uno de ellos:

Introducción: se justifica el proyecto, se indica el objetivo del mismo, el enfoque utilizado y se detalla la planificación que se ha seguido para su realización y ejecución.

Conceptos básicos: recopilación de las definiciones más importantes para facilitar la comprensión del proyecto. Los conceptos explicados en este apartado serán medidas, métricas, indicadores y atributos externos e internos.

Métricas: descripción de las principales métricas utilizadas a la hora de analizar la productividad de la gestión de un proyecto, estas serán métricas de proceso proyecto y *software*.

Métricas de productividad: uno de los principales motivos por los cuales se ha escrito este TFC.

Estándares para la medición: plasmaremos los acuerdos a los que se ha llegado para la medición de las métricas de la gestión de un proyecto, detallando los estándares más importantes.

Métricas de recursos: explicación de modelos de estimación, principalmente COCOMO.

Conclusiones: En este punto se elaborará un conjunto de conclusiones personales extraídas de la propia elaboración de este TFC.

Glosario: catálogo de definiciones

Bibliografía: detalle de la bibliografía utilizada

2. Conceptos Básicos

2.1 Medidas

Proporciona una indicación cuantitativa de la extensión, cantidad, dimensión, capacidad y medida de algunos atributos de un proceso o producto.

Esta definición trasladada a la gestión de proyectos de ingeniería del *software*, seria la cantidad de líneas de código existentes. A la hora de medir deben estar bien definidos los métodos y pautas a seguir para determinar que es y que no es una línea de código.

2.2 Métricas

Una métrica es una medida cuantitativa en la que un sistema, componente o proceso posee un atributo dado. De la misma manera que una medida registra el valor de una característica individual, la métrica nos permite interrelacionar medidas teniendo la capacidad de comparar los diferentes datos (medidas).

2.3 Indicadores

Métrica o combinación de métricas que proporcionan un análisis más detallado del proceso del *software*, del proyecto del *software* o del producto en sí.

Existen los indicadores de producto, de proyecto (uno o más productos) y de proceso (uno o más proyectos).

2.4 Atributos internos y atributos externos

Los atributos internos de un producto, proceso o recurso son aquellos que podemos medir sólo en términos del mismo producto, proceso o recurso. Pueden ser medidos de manera directa (por ejemplo el tamaño del equipo, 3 programadores).

Los atributos externos de un producto, proceso o recurso son aquellos que pueden ser medidos respecto a cómo dicho producto, proceso o recurso se relaciona con su entorno (por ejemplo la productividad del equipo, sería la relación de los 3 trabajadores del equipo con el trabajo realizado según tiempo empleado y complejidad del mismo).

En la siguiente tabla se muestra un conjunto de ejemplos de los dos tipos de atributos:

ENTIDADES	ATRIBUTOS	
Recursos	Internos	Externos
Trabajadores	Edad, sueldo, preparación	Productividad, experiencia, inteligencia
Equipo	Medida, estructuras	Productividad, calidad
Software	Precio, velocidad, tamaño	Usabilidad, seguridad, fiabilidad
Hardware	Precio, velocidad, memoria	Usabilidad, seguridad, fiabilidad
Oficinas	Tamaño, temperatura, luz, condiciones	Adaptabilidad, calidad
Procesos	Internos	Externos

Construcción de especificaciones	Tiempo, esfuerzo, cantidad de modificaciones	Calidad, coste, estabilidad
Diseño específico	Tiempo, esfuerzo	Coste, coste efectivo
Pruebas	Tiempo, errores encontrados	Estabilidad, coste, coste efectivo
Productos	Internos	Externos
Especificaciones	Tamaño, modularidad, funcionalidad	Comprensibilidad, mantenimiento
Diseños	Modularidad, medida, utilización	Calidad, complejidad, mantenimiento
Código	Funcionalidad, complejidad algorítmica, control de flujo	Seguridad, usabilidad, mantenimiento
Datos de prueba	Tamaño, nivel de protección	Calidad, cantidad

Tabla 1: Atributos internos y externos

2.5 Factores a tener en cuenta

Existen tres factores importantes a la hora de determinar la productividad en el desarrollo de un proyecto del *software* y que son comunes a cualquier otro tipo de proyecto, estos son:

Factores del proyecto: de aquí cabe destacar los principales que son el de personal (empleados) y el ambiente de desarrollo (metodología). Un empleado con una amplia experiencia y preparación académica *a priori* será mucho más productivo que un empleado con escasa experiencia y estudios. Un equipo de trabajo será mucho más eficiente si trabajan con una misma metodología, unas normas claras de documentación del trabajo que se va realizando, *backups*, gestión de versiones, patrones de codificación comunes…

Factores administrativos: se tiene en cuenta la cohesión y estabilidad del grupo de proyecto, la participación y la implicación de cada participante, y también los costes y los recursos fijos, limitaciones de funcionalidad o limitaciones de tiempo.

Factores del producto: se refiere a los aspectos que se pueden entender como más críticos, como puede ser el tiempo de desarrollo, el tiempo de puesta en marcha, (pruebas y funcionamiento), nivel de calidad, posibilidades de integración, escalabilidad y usabilidad, nivel de innovación y su complejidad en general.

3. Métricas

3.1 Métricas de proceso

Los datos de las métricas de proceso se acumulan proyecto por proyecto durante largos períodos de tiempo, de esta manera se obtiene una visión global sobre ciertos indicadores de proceso, pudiendo mejorar aquellos en los que se aprecia un bajo nivel de eficacia o productividad.

Se utilizan para mejorar la eficiencia del proceso del ciclo de vida del *software* (*Software Development Life Cycle*, *SDLC*). Para obtener estás métricas, inicialmente concretaremos los atributos específicos de un proceso, desarrollaremos un conjunto de métricas significativas sobre la información que disponemos de estos atributos, finalmente con las métricas se obtendrían los indicadores que ayudarían a obtener una estrategia que mejoraría el proceso para futuras situaciones.

3.2 Métricas del proyecto

Las métricas del proyecto son utilizadas por parte del director de proyecto y su equipo para controlar de una manera global la situación y transcurso del proyecto. Son las más relevantes a nivel de equipo de proyecto, de vital importancia sobre todo a la hora de la planificación y adecuando la estrategia que se tomará para llevar a cabo el proyecto. Uno de los aspectos más importantes que debe vigilarse es el de las desviaciones, el tiempo de más que se tarda en llevar a cabo una o más tareas, en este sentido estas métricas llevan a cabo la función de vigilar estas desviaciones para evitarlas o al menos minimizarlas.

Su objetivo es doble:

- Minimizar el calendario de tareas (evitando desviaciones)
- Evaluar periódicamente el estado de las tareas, modificando el enfoque técnico en caso necesario para mejorar la calidad

Cada proyecto debe medir:

Inputs → recursos necesarios para la realización de las tareas (trabajadores, *hardware,* entrono de trabajo)

Outputs → documentos, archivos…que se van entregando a medida que se van realizando las tareas

Results → medidas que indiquen la eficiencia de los resultados finales

3.3 Métricas del software

El objetivo de las métricas del software es la valoración cuantitativa de la calidad del *software*. Estas mediciones evalúan la calidad del análisis y los modelos de diseño, el código fuente y los casos de prueba utilizados para la confirmación del correcto funcionamiento.

Se clasifican en dos categorías:

- Directas: no depende de ninguna métrica de otro atributo (líneas de código, velocidad de ejecución, número de errores).
- Indirectas: se obtienen a través de métricas directas (complejidad, eficiencia, fiabilidad).

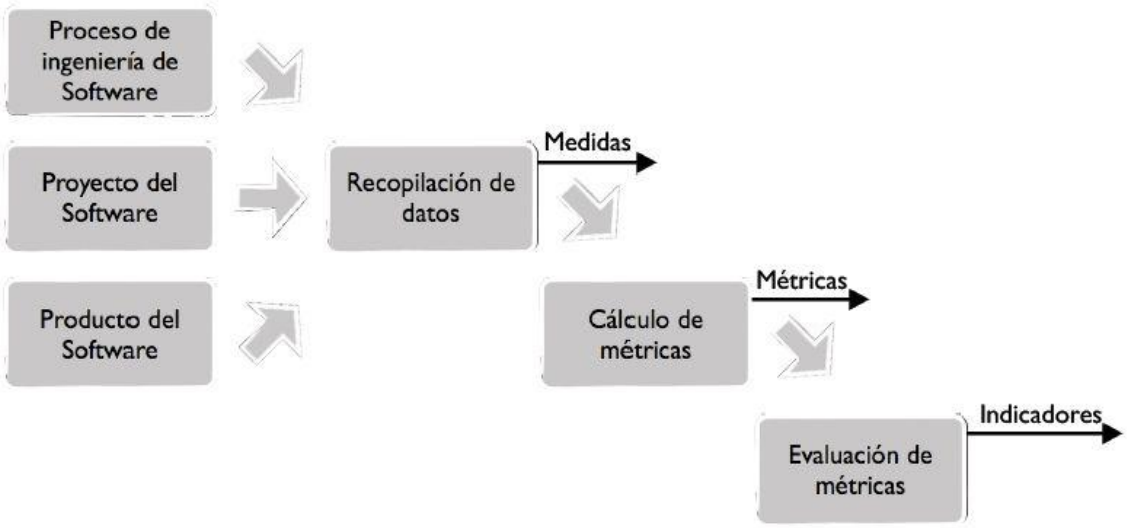

Figura 3: Proceso Recopilación de métricas de software

4. Métricas de productividad

4.1 Definición

La medición puede considerarse como una acción muy subjetiva y polémica puesto que quien esté creando las reglas de medición, juzgará por él mismo que métricas son apropiadas para el proceso y para el producto y decidirá cómo utilizar los datos obtenidos.

A pesar de la controversia que se podría generar existen motivos por los cuales es necesaria la medición. Ayudan a: indicar la calidad del producto, evaluar la productividad de los empleados que participan, justificar el uso de nuevas herramientas o de formación adicional.

Una de los objetivos prioritarios a la hora de iniciar la planificación de un proyecto es la fijar un control de los recursos, esfuerzos y tiempos utilizados para llevarlo cabo.

Las métricas de productividad tienen como objetivo medir el rendimiento del *software* en sí, como de productivo va a ser, centrándose en la reutilización del código, su fiabilidad, la funcionalidad del cliente, la distribución de los tiempos y tareas y la entrega a tiempo del producto.

4.2 Clasificación

Las métricas de productividad se clasifican en dos grupos:

4.2.1 Métricas orientadas al tamaño

Son medidas directas del *software* y del proceso. Indican en que tiempo se va a llevar a cabo el proyecto y cuantas personas se van a necesitar.

Se obtienen las siguientes fórmulas:

Productividad = KLDC / esfuerzo

Calidad = errores / KLDC

Documentación= págs. Documentación / KLDC

Costo = costo monetario / KLDC

* kldc = cantidad de líneas de código en millares; esfuerzo = personal/mes

4.2.2 Métricas orientadas a la función

Son medidas indirectas del software y se centran en la funcionalidad o utilidad del programa. Miden la cantidad de funciones que se van a lograr, y el número de: entradas del usuario, salida del usuario, peticiones del usuario, archivos, interfaces externas…

Las principales características son:

- Independiente de la tecnología

- Sencillo

- Enfocado a la funcionalidad proporcionada (que podré hacer con este producto)

- Basado en las necesidades del usuario

- Consistencia

4.3 Métricas de puntos características

La importancia adquirida por los puntos de función se puede constatar tanto en la enorme difusión de su aplicación, como en la diversidad de los métodos desarrollados. Existen fundamentalmente dos factores de dificultad a la hora de utilizar puntos de función: la comparación de medidas realizadas con diferentes métodos y la elección del método más adecuado para el usuario y el proyecto. A continuación explicaremos brevemente cuatro de los métodos más utilizados: FPA, COSMIC-FFP, IFPUG y MKII.

4.3.1 Método FPA

Es un método para medir el tamaño funcional de un sistema de información. Mide el tamaño funcional mirando las transacciones (funcionales) y los ficheros de datos (lógicos) más importantes para el usuario final.

El FPA evalúa con fiabilidad:

– El valor comercial de un sistema para el usuario.

– El tamaño del proyecto, costo y tiempo de desarrollo.

– Calidad y productividad del programador MIS.

– Esfuerzo de adaptación, modificación y mantenimiento.

– Posibilidad de desarrollo propio.

– Beneficios de implementación en 4GL.

4.3.2 Método COSMIC-FFP

COSMIC-FFP se creó para contar con un método más preciso para la estimación y medición de las características del *software* que sean igualmente fiables para todos los tipos del mismo, con los siguientes objetivos principales:

- Conocer la talla funcional del *software* como componente importante de la medición y de las características que se necesiten.

- Estimar los requerimientos en la talla del *software*, como paso a calcular costos del proyecto.

4.3.3 Método IFPUG

Orientado a medir el *software* cuantificando la funcionalidad que proporciona al usuario basándose en el diseño lógico. Principalmente lo que se busca es:

- Medir la funcionalidad que el usuario necesita y recibe.

- Medir el desarrollo y el mantenimiento del *software* independientemente del de la tecnología utilizada para su implementación.

Es importante resaltar que el proceso de contar los Puntos Función deberá ser lo suficientemente simple para minimizar el sobre-esfuerzo necesario del proceso de medir.

4.3.4 Método MKII

Los puntos de función MKII - FPA se obtienen sobre el producto de dos atributos: el tamaño de procesamiento de la información y el ajuste de complejidad técnica.

En el tamaño del procesamiento de la información en vez de utilizar cinco componentes: entradas, salidas, interfaces, consultas y ficheros lógicos, el sistema se examina desde el punto de vista de las transacciones lógicas, consistiendo cada una de ellas en entrada, proceso y salida.

En el ajuste de complejidad técnica se amplían la lista de características generales. Se incluyen las 5 características específicas y se permite introducir otras.

5. Estándares para la medición

5.1 Método Objetivo-Pregunta-Métrica (GQM)

Método definido por Basili y Weiss en 1984 y extendido posteriormente por Rombach. Es un enfoque que proporciona una manera útil para definir métricas tanto del proceso como de los resultados de un proyecto. Según este método, un programa de medición puede ser más satisfactorio si se diseña orientado a las metas u objetivos que se quieren alcanzar.

Las metas son generales, abstractas e intangibles. Responden a la pregunta de ¿qué queremos alcanzar?, y la respuesta es cualitativa (por ejemplo reducir el tiempo de entrega). Los objetivos por su parte son precisos, tangibles y concretos.

Las metas se descomponen en un conjunto de objetivos bien definidos, con la intención de que alcanzando los objetivos alcanzaremos la meta. Responden a la pregunta de ¿cuánto queremos alcanzar?, y la respuesta es cuantitativa (por ejemplo reducir el tiempo de entrega en un 10% al final de año).

GQM define una meta, descompone esta meta en preguntas y define métricas que intentan dar información para responder a estas preguntas. Se puede aplicar a todo el ciclo de vida del producto, procesos y recursos. GQM sigue un proceso de seis pasos donde, los tres primeros tratan de identificar las métricas a partir de las metas del negocio y los tres últimos se basan en la recopilación de los datos de las medidas y su utilización eficaz en la toma de decisiones.

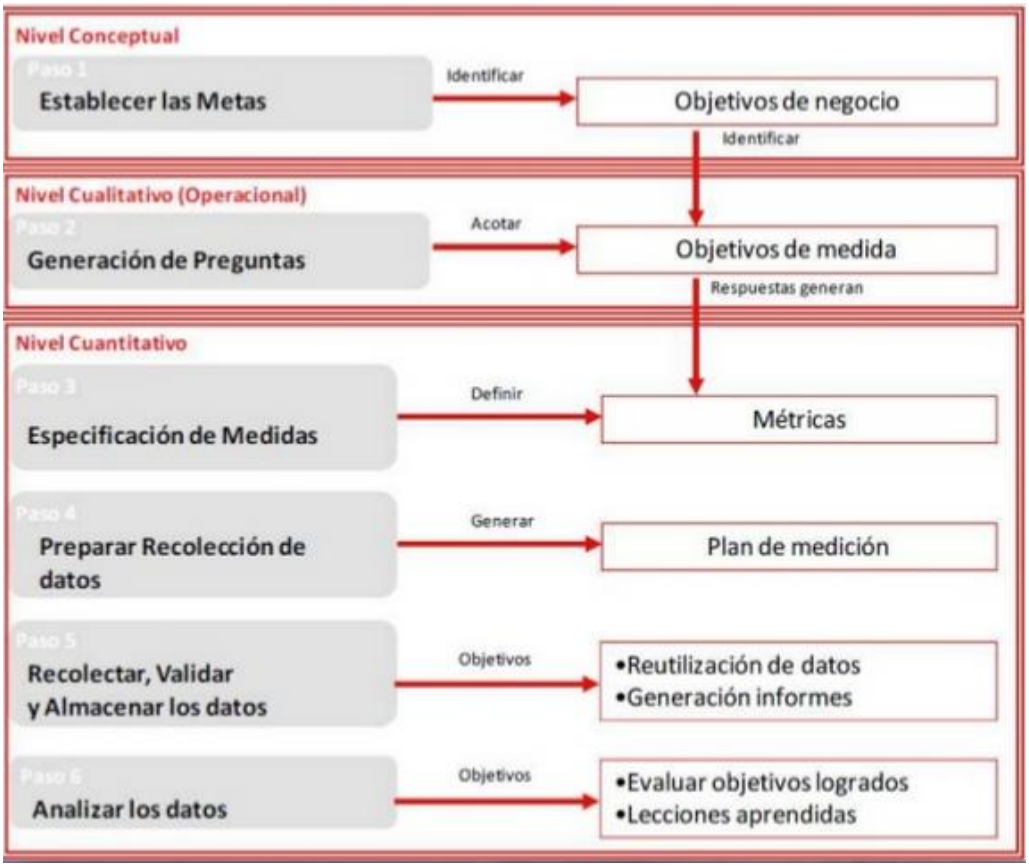

Figura 4: Esquema método GQM

Nivel Conceptual → los objetivos identifican lo que queremos lograr respecto a los productos, procesos o recursos.

Nivel Cualitativo → las preguntas nos ayudan a comprender cómo satisfacer el objetivo. Tratan de caracterizar al objeto de la medición con respecto a un aspecto de calidad concreto y tratan de determinar la calidad de dichos objetos desde el punto de vista seleccionado, analizando el grado de cumplimiento de los objetivos específicos.

Nivel Cuantitativo → se asocia un conjunto de datos a cada pregunta, con el fin de proporcionar una respuesta de manera cuantitativa. Los datos pueden ser: objetivos si dependen únicamente del objeto que se está midiendo y no del punto de vista desde el que se captan (por ejemplo, el número de versiones de un documento), subjetivos: si dependen tanto del objeto que se está midiendo como del punto de vista desde el que se captan (por ejemplo, el nivel de satisfacción del usuario).

5.2 Métricas de calidad

5.2.1 Definición y características

La calidad del *software* es, según Pressman, la "concordancia con los requisitos funcionales y de rendimiento, con los estándares de desarrollo y con las características implícitas que se espera del *software* desarrollado profesionalmente"

No existe una definición única de calidad, ya que:

- Es un concepto relativo (es una compleja mezcla de factores que varía para las diferentes aplicaciones y los clientes que las solicitan).
- Es un concepto multidimensional, referido a muchas cualidades.
- Está ligada a restricciones (por ejemplo, el presupuesto).
- Está ligada a compromisos aceptables (por ejemplo, plazos de fabricación).
- No es ni totalmente subjetiva ni objetiva.

Puede resultar transparente cuando está presente y reconocible cuando está ausente. Actualmente, la calidad del *software* debe tenerse en cuenta a dos niveles:

A nivel de empresa: para conseguir *software* de calidad, las organizaciones deben tener una estructura organizativa apropiada para fomentar el trabajo por la calidad de todas las personas y departamentos de la empresa, además de fomentar procesos específicos para asegurar la calidad.

A nivel de proyecto: se trata de llevar a la práctica en las actividades cotidianas las disposiciones fijadas en el sistema de calidad. Se aplica durante todo el proceso de ingeniería del *software*, es decir, en análisis, diseño, codificación y prueba.

5.2.2 Estándares

Los estándares de calidad de *software* son normas emitidas por organismos específicos, que sirven para establecer un marco con el que comparar si un proceso de desarrollo es o no de calidad. Establecen el nivel máximo y mínimo aceptables para un indicador, el cual determina el punto en el que se está cumpliendo la calidad.

Los principales estándares aplicados al desarrollo del software son: ISO, SPICE y CMM.

5.2.2.1 Estándar ISO-9000

Se refiere a los sistemas de gestión de calidad y especifica la manera en que se debe documentar efectivamente los elementos de los sistemas de calidad, hay que destacar que no especifica la tecnología a usar.

Sus principales ventajas:

- Estandariza las actividades de personal mediante la documentación.

- Incrementa la satisfacción del cliente

- Mide y monitoriza el desempeño de procesos.

- Disminuye la repetición de procesos.

- Incrementa eficacia y eficiencia de la organización al lograr objetivos.

- Mejora continuamente en los procesos, productos, eficacia…

- Reduce los errores de producción o prestación de servicios.

A su vez, la serie ISO 9000 proporciona las siguientes series:

- ISO 9001: diseño, manufactura, instalación y sistemas de servicio.

- ISO 9002: producción e instalación.

- ISO 9003: inspección y examen de productos finales

- ISO 9004: proporciona una guía interna para desarrollar sistemas propios de calidad.

5.2.2.2 Estándar SPICE (ISO 15504)

Conjunto de normas y técnicas para el proceso de desarrollo de *software* y funciones relacionadas con la gestión empresarial (*Software Process Improvement and Capability Determination*). Es una importante iniciativa internacional, similar a CMMI, para apoyar el desarrollo de una norma internacional para la evaluación de procesos de software. Clasifica las organizaciones en seis niveles de madurez, de 0 a 5 (de incompleto a optimizado).

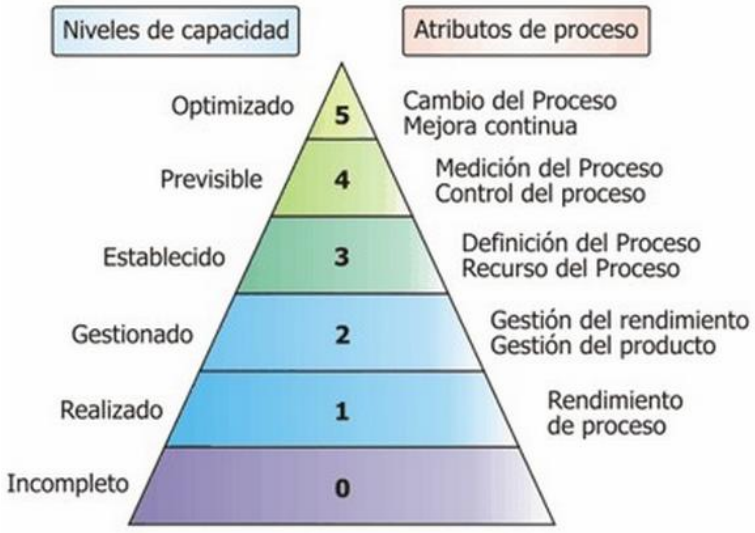

Figura 5: Esquema método GQM

5.2.2.3 Estándar CMMI

Modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software (*Capability Maturity Model Integration*). Clasifica las empresas en niveles de madurez, estos niveles sirven para conocer la madurez de los procesos que se realizan para producir *software*. Surge como la integración del CMM (*Capability Maturity Model*) v.2.0 y de la ISO 15504 *Draft Estándar* v.1.00.

Este modelo de procesos tiene dos representaciones: continua y por etapas, siendo la diferencia entre estas la evaluación por niveles de la capacidad de procesos o de la madurez de la organización, respectivamente.

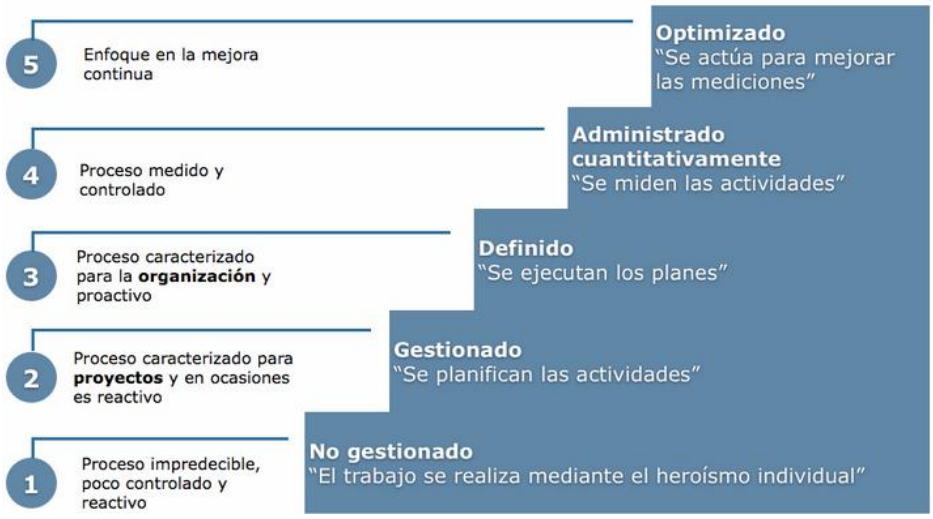

Figura 6: Niveles método CMMI

A continuación explicaremos brevemente cada nivel:

Nivel 1: las organizaciones en este nivel no disponen de un ambiente adecuado para el desarrollo de *software*, no existe planificación. El éxito de los proyectos se basa en la mayoría de los casos en el esfuerzo personal, aunque a menudo se producen fracasos y casi siempre retrasos y sobre costos.

Nivel 2: se definen claramente puntos de control en cada etapa principal del proyecto, aunque no se llega a saber con precisión como se desenvuelve el proyecto en cada etapa.

Nivel 3: los procesos comunes para desarrollo y mantenimiento del software están documentados de manera suficiente en una biblioteca accesible a los equipos de desarrollo. Las personas han recibido la formación necesaria para comprender los procesos.

Nivel 4: las métricas utilizadas en este punto no son subjetivas si no que se establecen con criterios cuantitativos formalmente definidos. Con el tiempo estos controles nos

proporcionarán mejor información sobre la calidad y estado del proyecto permitiéndonos compararlo con otros proyectos similares y así poder detectar desviaciones.

Nivel 5: en este nivel cada proceso es analizado y controlado permanentemente con la intención de mejorarlo en todo momento.

6. Método COCOMO

6.1 Introducción

El Modelo Constructivo de Costes (*Constructive Cost Model*) es un modelo matemático de base empírica utilizado para la estimación de costes de *software*. Se público por primera vez en 1981 por Barry Boehm. En la década y media siguiente las técnicas de desarrollo software cambiaron drásticamente. Estos cambios incluyen el gasto de tanto esfuerzo en diseñar y gestionar el proceso de desarrollo software como en la creación del producto software. Se conoce como COCOMO I ó COCOMO 81.

6.2 Modelos de estimación

6.2.1 COCOMO 81

COCOMO 81 asume que el modelo de desarrollo que se utiliza es en Cascada. Como ya hemos comentado anteriormente Es un modelo matemático de base empírica que permite la estimación del coste y la duración de los proyectos de software (esfuerzo y tiempo). Es empírico debido a que se basa también en los resultados obtenidos en proyectos ya realizados.

En el modelo de COCOMO existen tres tipos distintos de proyectos a elegir, según el que más nos interese para nuestra situación:

- Modo orgánico: se corresponde con proyectos sencillos, proyectos en los cuales se tiene mucha experiencia desarrollándolos y cuyo entorno es estable (dimensión del proyecto suele ser de hasta 50.000 LDC).
- Modo semiencajado (o semiacoplado). La complejidad de los proyectos es superior al anterior, también se caracteriza en que el equipo está formado a partes iguales por personas con experiencia y personas sin ella (pueden llegar a tener una dimensión de 300.000 LDC).
- Modo empotrado. Son los proyectos más complejos donde la experiencia del equipo es limitada sino nula, el proyecto puede llegar a incluir grandes innovaciones técnicas.

Para todos ellos las fórmulas del cálculo son las mismas:

$$\mathbf{E} = a \cdot (KLOC)^b \cdot m(X)$$

$$\mathbf{Tdev} = c \cdot (E)^d$$

$$\mathbf{P} = \frac{E}{Tdev}$$

Donde:

E → Es el esfuerzo medido en personas/mes
Tdev → Es el tiempo estimado en meses
P → Es el número de personas requerido para el proyecto
a, b → Son constantes con valores definidos según cada modo y cada modelo
c, d → Son constantes con valores definidos según cada modo
KLOC → Son el número de miles de líneas de código fuente que tiene el software que estamos intentado estimar

A la vez existen dos modelos que varían los valores de las constantes a y b:

Modelo básico: basado en el tamaño expresado en miles de líneas de código (KLOC), aquí m(X)) es igual a uno.

Modelo intermedio: Toma como entradas las miles de líneas de código (KLOC) y un multiplicador (m(X)) calculado a partir de 15 parámetros denominados Atributos de Coste. Los atributos de coste lo que permiten es valorar el entorno de desarrollo del proyecto para tenerlo en cuenta en la estimación.

A continuación se muestran los valores de las constantes según el modelo:

Modelo Básico

Modo	A	B	C	D
Orgánico	2.40	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

Modelo Intermedio

Modo	A	B	C	D
Orgánico	3.20	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	2.80	1.20	2.50	0.32

Figura 7: Valores constantes modelo COCOMO

6.2.2 COCOMO II

Cuando se publicó el primer modelo COCOMO 81 los programadores estaban sometidos a tareas *batch*. El giro total que se experimentó afectó a su productividad, el programador ya no tenía que esperar el parámetro de retorno que reflejaba la espera media de un programador para recibir de vuelta su trabajo, ahora no tiene sentido porque la mayoría de los

programadores tienen acceso instantáneo a los recursos computacionales a través de su estación de trabajo.

Es importante recalcar los cambios experimentados por COCOMO II con respecto a COCOMO 81 ya que reflejan cómo ha madurado la tecnología de la ingeniería de *software* durante los últimos 20 años.

Los objetivos a la hora de crear el modelo COCOMO II fueron:

- Desarrollar un modelo de estimación de tiempo y de coste del *software* de acuerdo con los ciclos de vida utilizados en los 90 y en la primera década del 2000.

- Desarrollar bases de datos con costes de *software* y herramientas de soporte para la mejora continua del modelo.

- Proporcionar un marco analítico cuantitativo y un conjunto de herramientas y técnicas para la evaluación de los efectos de la mejora tecnológica del *software* en costes y tiempo del ciclo de vida software.

- Implementar una herramienta que utilizara el modelo COCOMO II.

A continuación mostramos un cuadro con las principales características del inicial modelo COCOMO 81 y el actual COCOMO II:

	COCOMO 81	COCOMO II
ESTRUCTURA DEL MODELO	Un modelo único que asume que se ha comenzado con unos requisitos asignados para el software	Tres modelos que asumen que se progresa a lo largo de un desarrollo de tipo espiral para consolidar los requisitos y la arquitectura, y reducir el riesgo.
FORMA MATEMÁTICA DE LA ECUACIÓN DEL ESFUERZO	Esfuerzo = A (q) (SIZE)$^{\text{Exponenente}}$	Esfuerzo = A (q) (SIZE)$^{\text{Exponenente}}$
EXPONENTE	Constante fija seleccionada como una función de modo: ▪ Orgánico = 1.05 ▪ Semi-libre = 1.12 ▪ Rígido = 1.20	Variable establecida en función de una medida de cinco factores de escala: ▪ **PREC** Precedencia ▪ **FLEX** Flexibilidad de desarrollo ▪ **RESL** Resolución de Arquitectura / Riesgos ▪ **TEAM** Cohesión del equipo ▪ **PMAT** Madurez del proceso
MEDIDA	Líneas de código fuente (con extensiones para puntos de función)	Puntos objeto, Puntos de función ó líneas de código fuente.
DRIVERS DE COSTE (c_i)	15 drivers, cada uno de los cuales debe ser estimado: ▪ **RELY** Fiabilidad ▪ **DATA** Tamaño Base de datos ▪ **CPLX** Complejidad ▪ **TIME** Restricción tiempo de ejecución ▪ **STOR** Restricción de almacenamiento principal ▪ **VIRT** Volatilidad máquina virtual ▪ **TURN** Tiempo de respuesta ▪ **ACAP** Capacidad del analista ▪ **PCAP** Capacidad programador ▪ **AEXP** Experiencia aplicaciones ▪ **VEXP** Experiencia máquina virtual ▪ **LEXP** Experiencia lenguaje ▪ **TOOL** Uso de herramientas software ▪ **MODP** Uso de Técnicas modernas de programación ▪ **SCED** Planificación requerida	17 drivers, cada uno de los cuales debe ser estimado: ▪ **RELY** Fiabilidad ▪ **DATA** Tamaño Base de datos ▪ **CPLX** Complejidad ▪ **RUSE** Reutilización requerida ▪ **DOCU** Documentación ▪ **TIME** Restricción tiempo de ejecución ▪ **STOR** Restricción de almacenamiento principal ▪ **PVOL** Volatilidad plataforma ▪ **ACAP** Capacidad del analista ▪ **PCAP** Capacidad programador ▪ **AEXP** Experiencia aplicaciones ▪ **PEXP** Experiencia plataforma ▪ **LTEX** Experiencia lenguaje y herramienta ▪ **PCON** Continuidad del personal ▪ **TOOL** Uso de herramientas software ▪ **SITE** Desarrollo Multi-lugar ▪ **SCED** Planificación requerida
OTRAS DIFERENCIAS DEL MODELO	Modelo basado en: ▪ Fórmula de reutilización lineal ▪ Asunción de requisitos razonablemente estables	Tiene muchas otras mejoras que incluyen: ▪ Fórmula de reutilización No-lineal ▪ Modelo de reutilización que considera esfuerzo necesario para entender y asimilar ▪ Medidas de rotura que se usan para abordar la volatilidad de requisitos ▪ Características de autocalibración

Tabla 2: Comparativa entre COCOMO 81 y COCOMO II

De la tabla podemos resumir lo siguiente:

- COCOMO II se dirige a las siguientes tres fases del ciclo de vida en espiral: desarrollo de aplicaciones, diseño anticipado y post-arquitectura.
- Los tres modos del exponente se han reemplazado por cinco factores de escala.
- Se han añadido los siguientes drivers de coste a COCOMO II: DOCU, RUSE, PVOL, PEXP, LTEX, PCON y SITE.

- Se han eliminado los siguientes drivers de coste del modelo original: VIRT, TURN, VEXP, LEXP y MODP.
- Los valores de los drivers de coste que se han conservado del modelo original, han sido modificados considerablemente para reflejar las calibraciones actualizadas.

Para cubrir los distintos sectores del mercado de *software*, COCOMO II proporciona una familia de modelos de estimación de coste software cada vez más detallado y tiene en cuenta las necesidades de cada sector y el tipo de información disponible para sostener la estimación del coste *software*. Existen tres submodelos cada uno de los cuales ofrece mayor nivel de detalle a medida que uno avanza en la planificación del proyecto y en el proceso de diseño. Estos tres submodelos son:

- Composición de aplicaciones. Indicado para proyectos construidos con herramientas modernas de construcción de interfaces gráficos para usuario.
- Diseño anticipado. Este modelo puede utilizarse para obtener estimaciones aproximadas del coste de un proyecto antes de que esté determinada por completo su arquitectura. Utiliza un pequeño conjunto de drivers de coste nuevo y nuevas ecuaciones de estimación. Está basado en Punto de Función sin ajustar o *KSLOC* (miles de líneas de código fuente).
- El modelo Post-Arquitectura. Este es el modelo COCOMO II más detallado. Se utiliza una vez que se ha desarrollado por completo la arquitectura del proyecto. Tiene nuevos *drivers* de coste, nuevas reglas para el recuento de líneas y nuevas ecuaciones.

7. Conclusiones

En todos los aspectos de la vida utilizamos todo tipo de unidades de medida para controlar una gran diversidad de características o cantidades, todo ello para mantener un control de lo que estamos haciendo y para poder comunicar que estamos haciendo, en qué medida. Medir el *software* no es una actividad sencilla, es el resultado de una actividad básicamente intelectual.

A la hora de iniciar un proyecto de *software* no existe un uso generalizado de las métricas de productividad a nivel empresarial. Este hecho provoca que repercuta en desviaciones de tiempo, y de una manera relacionada, en un encarecimiento del producto, aspecto que el cliente no aceptará de buen grado.

Si se aplicasen las métricas de productividad indicadas en este trabajo u otras existentes en el mercado, el control de la evolución de los proyectos se simplificaría, las tareas se llevarían se finalizarían en un tiempo más acorde con lo previsto y los empleados trabajarían de una manera más cómoda y tranquila. El conocimiento de lo que se ha hecho, de lo que tus compañeros han hecho (documentación), de lo que se tiene que hacer, en qué tiempo y los recursos disponibles (librerías…), facilitan la correcta finalización del proyecto.

El modelo COCOMO ya hace décadas que se encuentra entre nosotros y evoluciona a la vez que nuestras necesidades y capacidades tecnológicas. Toda la información que necesitemos de este modelo es público y existen diferentes herramientas que utilizan este modelo como puede ser: USC-COCOMO II, CostXpert o Costar, este último de pago.

8. Bibliografia

Pressman, R.S., *Ingeniería del Software, un enfoque práctico*, Mc Graw Hill, 1998.

L.Putnam and W.Myers, "*Measures for Excellence: Reliable Software On Time, Within Budget*", Yourdon Press 1992

Dolado, J. y Fernández, "*Medición para la gestión de proyectos de software*", Ed. Ra-ma, 2000

Enlaces Internet

Enma E. Salazar, Manuel F. Salazar "Métricas de proceso y proyecto" metricas-de-proceso-y-proyecto.doc

ITU department of computer&software engineering **software process and project metrics**

http://www-2.dc.uba.ar/materias/isoft2/2006_02/clases/Software_Metrics_20061030.pdf

http://es.slideshare.net/May11IM0883/metricas-de-software-4278643

http://132.248.9.34/hevila/Gerenciatecnologicainformatica/2009/vol8/no22/5.pdf

http://www.eici.ucm.cl/Academicos/ygomez/descargas/Ing_Sw2/apuntes/cocomo_manual_espanol.pdf

http://es.slideshare.net/LuisdelaCruz2/cocomo