Universidad Latinoamericana de Ciencia y Tecnología

Facultad de Ingeniería

Escuela de Ingeniería Informática

Trabajo final para optar al grado de Licenciatura en Informática con Énfasis en Gestión de Recursos Tecnológicos

Tema:

Análisis de los tipos de medición de software.

Sustentante: Edith Cattia Baldiviezo Ochoa

Cédula: 8-087-269

Tutor: Lic. Miguel Pérez Montero

III cuatrimestre del 2009

Resumen ejecutivo

El ser humano en su afán de tener dominio y control de su ambiente se las ha ingeniado para realizar mediciones; por eso, en nuestros días, existen una infinidad de métricas comenzando en lo cotidiano de nuestro vivir, en la música, en el tiempo, etc. y las métricas de software no podían ser la excepción, de ellas hablaremos en este artículo.

Por medio de una clasificación de las métricas de software y la descripción de las más representativas, se proporcionan elementos de valor que son un aporte al conocimiento sobre este tema, y que pueden ser igualmente beneficiosas si somos desarrolladores de software, gerentes, etc.

La tecnología de la informática desde sus inicios hasta la fecha ha tenido un desarrollo constante que apunta hacia la calidad. En la actualidad, la calidad ya no es una utopía, es un requisito para poder complacer a los clientes cada vez más exigentes, por medio de ella las empresas logran mantenerse en el mercado y las métricas son herramientas de gran ayuda.

Abstract

The Human in their desire to have dominion and control of their environment has managed to make measurements, which is why, nowadays, there are an infinite number of metrics starting in the everyday of our lives, in music, in time, etc. and software metrics could not be the exception, of which we will discuss in this article.

Using a classification of software metrics and descriptions of the most representative, are provided items of value that they add to knowledge about this subject, and can be equally beneficial if we are software developers, managers, etc.

The information technology since the beginnings until our times has been a steady development aimed towards quality. At present, the quality is no longer a utopia, is required to please customers increasingly demanding, through her companies manage to stay on the market and the metrics are very helpful tools.

Frases descriptoras

Métricas de software – Métricas de proceso – Métricas de producto – Métricas de proyecto – Medidas de software

Índice de contenido

Introducción

Comenzaremos hablando del significado de medir. En una de las definiciones dadas en el Diccionario de la Lengua Española, medir significa: “comparar una cantidad con su respectiva unidad, con el fin de averiguar cuántas veces la segunda está contenida en la primera”.

Basados en esta definición y dando una mirada retrospectiva del accionar del ser humano a través de su evolución hasta nuestros días vemos que la medición representa un papel muy importante en su quehacer cotidiano, desde algo tan sencillo como medir los ingredientes para una receta, como la medición del avance de un proyecto de una expedición al espacio.

En todas las ingenierías es muy importante el uso de la medición al igual que en la Ingeniería de software, esto lo vemos reflejado en una encuesta realizada por el SEI (*Software Engineering Institute*) es una institución operada por Carnegie Mellon University para el Departamento de Defensa de USA. Esta encuesta se llevó a cabo el año 2006, en 84 países, y sus objetivos de estudio fueron:

- El grado en que los profesionales utilizan la medición de software de medición cuando realizan su trabajo.
- La percepción del valor de medición.
- Enfoques que son utilizados para guiar, cómo la medición es definida y usada.
- Los tipos de medición más usados por los profesionales de software.

Una de sus preguntas relacionadas con la medición fue: ¿cree que la medición ayuda en algún grado? Obtuvieron 1868 respuestas de las cuales el 92% estuvo de acuerdo; el 2%, en desacuerdo; el 4% no estuvo seguro y el 2% no se pronunció ni en acuerdo, ni en desacuerdo.

Este es un porcentaje muy elevado de aceptación de la medición de software como un apoyo en la consecución de los objetivos de calidad en el software.

Con este artículo se pretenden mostrar los tipos de medición más utilizados o de renombre en el mercado del software.

Conceptos básicos de medición de software:

La medición de software es una disciplina joven, aún más joven que la Ingeniería del software, y ello ha influido notablemente en que esta última no haya alcanzado aún el grado de madurez que tienen las otras ingenierías (Piattini, Garcia, Garzás y Genero, 2008).

Con base en ello podemos deducir que, debido a que se trata de una disciplina relativamente joven, existen ciertos vacíos en lo que se refiere a los conceptos que se manejan porque todavía no se han normalizado los estándares internacionales.

En un informe técnico realizado por la UCLM (Universidad de Castilla-La Mancha, 2006) optaron por organizar la Ontología de la Medición de Software (SMO) de la siguiente forma:

a) **Caracterización y objetivos de la medición software**, que incluye los conceptos necesarios para establecer el ámbito y los objetivos del proceso de medición software. El principal objetivo del proceso de medición software es satisfacer ciertas *necesidades de información* identificando las *entidades* (que pertenecen a *categoría de entidad*) y los *atributos* de estas *entidades* (que son el enfoque del proceso de medición). *Atributos* y *necesidades de información* están relacionados por medio de *conceptos medibles* (que pertenecen a *modelo de calidad*).

b) **Medidas software**, que trata de establecer y aclarar los elementos claves en la definición de una *medida software*. Una *medida* relaciona un *método de medición* definido y una *escala* de medición (que pertenece a *tipo de escala*). La mayoría de las *medidas* pueden ser expresadas en una *unidad de medición*, y pueden ser definidas para más de un *atributo* (medidas nominales son ejemplos de medidas

que no pueden ser expresadas en unidades de medición). Las *medidas* pueden ser de 3 tipos: *medidas base, medidas derivadas e indicadores.*

c) **Formas de Medir**. Esta subontología introduce el concepto de *forma de medir* para generalizar los diferentes "métodos" usados por los tres tipos de *medidas* para obtener sus respectivos *resultados de medición*. Una *medida base* aplica un *método de medición*. Una *medida derivada* usa una *función de cálculo* (que se apoya en otra medida base y/o derivada). Por fin, un *indicador* utiliza un *modelo de análisis* (basado en un *criterio de decisión*) para obtener un *resultado de medición* que satisfaga a una *necesidad de información*.

d) **Medición**. Establece la terminología relacionada con el acto de medir el software. Una *medición* (que es una acción) es un conjunto de operaciones cuyo objetivo es determinar el valor de un *resultado de medición* para un determinado *atributo* de una *entidad*, utilizando una *forma de medir*. Los *resultados de la medición* se obtienen a través de la acción de llevar a cabo una *medición*.

Sobre el significado de ontología, este término se utiliza para indicar un marco estructural en el que una teoría se encuentra incluida (SALVAT, 2004).

Para aplicar una métrica de software es necesario que se implementen niveles de madurez en la organización de acuerdo con modelos de evaluación y mejora tales como el ISO 15504 ó CMMI. Se debe establecer una base cuantitativa que de menor a mayor grado de madurez debe estar enfocada sobre:

- Medición del proceso, basado en el estudio y control de la capacidad de los procesos, así como en la gestión de los cambios en el proceso.
- Medición del proyecto, basado en la gestión de proyectos.
- Medición del producto, centrado en su calidad y aspectos técnicos.

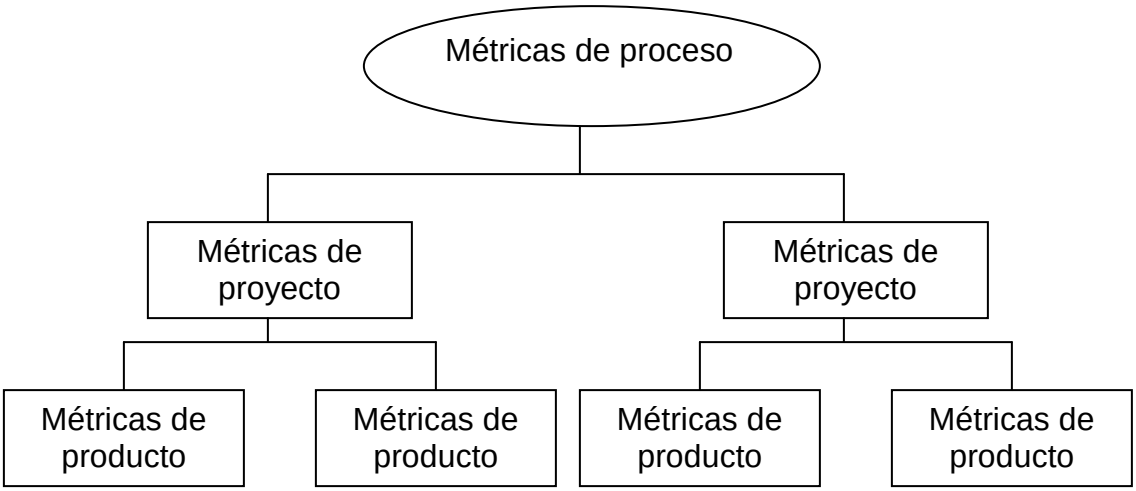

Figura 1. Relación entre las métricas del proceso, proyecto y producto.

Fuente: (Piattini et al., 2008).

1) Medición del proceso

Como se puede observar en la Figura 1, el proceso de software constituye la base a partir de la cual se realiza el trabajo dentro de una organización (Piattini et al., 2008).

Para entender el significado del proceso, agarramos la definición encontrada en el Diccionario (SALVAT, 2004) “Conjunto de fases sucesivas de un fenómeno natural o de una operación artificial”.

Un proceso de software es cualquier actividad relacionada con el desarrollo de software (Riguzzi, 1996).

Pressman (2006) contesta las siguientes preguntas con respecto al proceso de software:

¿Qué es un proceso de software?

Cuando se trabaja par construir un producto o sistema es importante seguir una serie de pasos predecibles: una especia de mapa de carreteras que ayude a crear un resultado de alta calidad y a tiempo. El mapa de carreteras que debe seguirse se llama proceso de software.

¿Quién lo hace?

Los ingenieros de software y sus jefes adaptan el proceso a sus necesidades y después lo siguen. Además, la gente que ha solicitado el software tiene una función qué desempeñar en el proceso de definirlo, construirlo y probarlo.

¿Por qué es importante?

Porque ofrece estabilidad, control y organización a una actividad que puede volverse caótica si no se controla. Sin embargo, un enfoque de ingeniería del software moderno debe ser “ágil”. Debe requerir sólo aquellas actividades, controles y documentaciones apropiados par el equipo del proyecto y el producto que ha de producirse.

¿Cuáles son los pasos?

El proceso que se adopte dependerá del software que se está construyendo.

¿Cuál es el producto obtenido?

Los productos obtenidos son los programas, documentos y datos. Desde el punto de vista del ingeniero de software.

¿Cómo puedo estar seguro de que lo he hecho correctamente?

Existen muchos mecanismos de evaluación del proceso de software que pueden determinar su nivel de madurez. No obstante, la calidad, el tiempo requerido, la viabilidad a largo plazo del producto que se construye son los mejores indicadores de la eficacia del proceso que se utiliza.

Con las definiciones anteriores podemos rescatar que el **proceso de software** comprende todas las actividades que se realizan para obtener un producto de software.

El proceso de software viene a ser la estrategia que utilizan las empresas para producir software, pueden aplicar procesos que ya han sido depurados por otras empresas, hasta derivados de otras industrias; es primordial tener métricas del proceso porque si está mal planteado o interpretado repercutirá en los proyectos y los productos.

Según Pressman (2006) las **métricas del proceso** se recopilan en el curso de todos los proyectos y durante largos periodos. Su objetivo es proporcionar un conjunto de indicadores de proceso que conduzcan a la mejora de los procesos de software de largo plazo. Proporcionan una amplia visión del proceso y un conocimiento detallado acerca del proyecto.

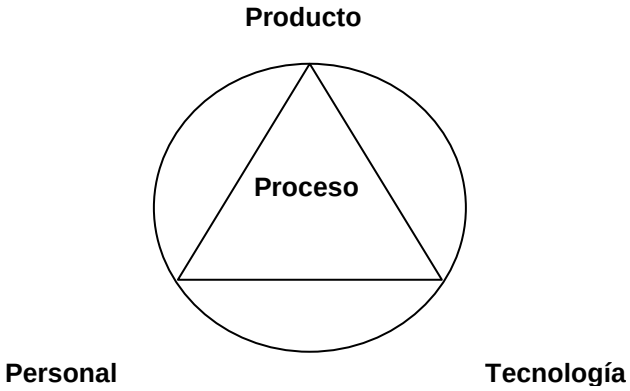

Figura 2. El proceso y los tres factores que influyen en la calidad de software.

Fuente: (Pressman, 2006).

El proceso se encuentra en el centro de un triángulo que conecta a tres factores que tienen mucha influencia sobre la calidad del software y desempeño organizacional:

Personal, la destreza y motivación del personal, que es uno de los factores que más influencia tiene en la calidad y desempeño.

Producto, la complejidad del producto tiene un impacto sustancial en la calidad y desempeño del equipo.

Tecnología, comprende los métodos y herramientas de ingeniería de software que también tienen una alta influencia.

El proceso de software se mide indirectamente utilizando un conjunto de métricas que se basan en los resultados que se derivan del proceso como por ejemplo: resultados de los errores descubiertos antes de liberar el software, los defectos que detectan y reportan los usuarios finales, los productos de trabajo entregados (productividad), el esfuerzo humano gastado, el tiempo de la planificación consumido. Pressman (2006).

Piattini et al., 2008 señala otro enfoque de medición de proceso es a nivel conceptual como, por ejemplo, el modelo conceptual SPEM (Software Process Engineerin Metamodel).

Este modelo conceptual se basa en que un proceso de desarrollo de software es el resultado de la colaboración entre entidades abstractas y activas, denominadas “roles de proceso”, que realizan operaciones denominadas “actividades” sobre entidades tangibles denominadas “productos de trabajo”. A la hora de establecer las métricas de los modelos de procesos se consideran dos niveles de alcance:

Nivel de modelo, métricas que se aplican para medir la complejidad estructural del modelo de procesos en conjunto.

Nivel de los elementos fundamentales del modelo, actividad, rol del proceso y producto de trabajo.

El nivel de modelo se puede representar de la siguiente manera:

Métrica	Definición
NA(MP)	Número de actividades del modelo de procesos
NPT(MP)	Número de productos de trabajo del modelo de procesos
NRP(MP)	Número de roles que intervienen en el proceso
NDPTIn(MP)	Número de dependencias de entrada de los productos en las actividades del modelo
NDPTOut(MP)	Número de dependencias de salida de los productos en las actividades del modelo
NDPT(MP)	Número de dependencias de productos de trabajo NDPT(MP) = NDPTIn(MP) + NDPTOut(MP)
NDPA(MP)	Número total de dependencias de precedencia entre actividades.
NCA(MP)	Nivel de conectividad entre actividades NCA(MP) = NA (MP) / NDPA(MP)
RDPTIn(MP)	Proporción de dependencias de entrada de Productos de Trabajo RDPTIn(MP) = NDPTIn(MP) / NDPT(MP)
RDPTOut(MP)	Proporción de dependencias de salida de productos de trabajo RDPTOut = NDPTOut(MP) / NDPT(MP)
RPTA(MP)	Proporción de productos de trabajo y actividades RPTA(MP) = NPT(MP) / NA(MC)

RRPA(MP)	Proporción de roles de proceso y actividades RRPA(MP) = NRP(MP) / NA(MP)

Tabla 1. Métricas a nivel de modelo de procesos.

Fuente: (Piattini et al., 2008).

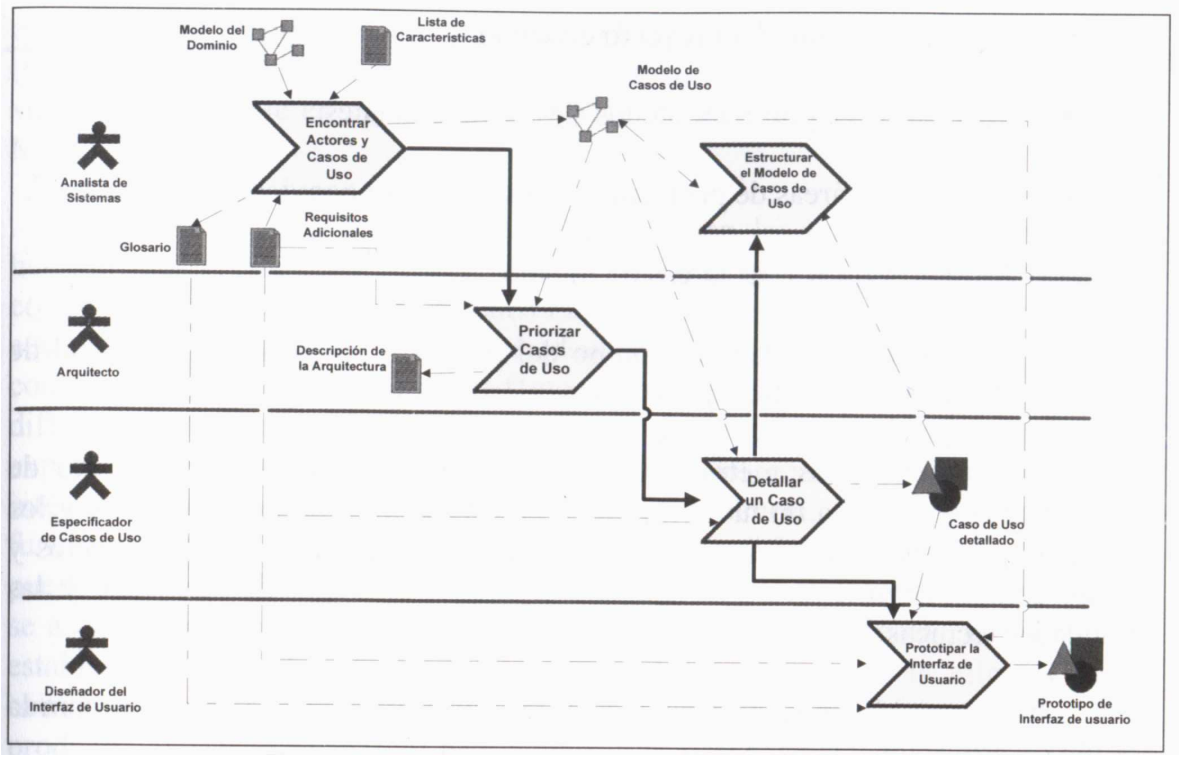

Figura 3. Ejemplo de un modelo de procesos.

Fuente: (Piattini et al., 2008).

Métrica	Valor	Métrica	Valor
NA	5	NDPA	4
NPT	8	NCA	1,25
NRP	4	RDPTIn	0,68
NDPTIn	13	RDPTOut	0,32
NDPTOut	6	RPTA	1,6

NDPT	19	RRPA	0,8

Tabla 2. Valores de las métricas aplicadas al diagrama de la figura 3.

Fuente: (Piattini et al., 2008).

El nivel de los elementos fundamentales del modelo se puede representar utilizando diagramas de UML para representar las distintas vistas del proceso de software, como se muestra en la Figura 3.

Los valores obtenidos en la Tabla 1, Métricas a nivel de modelo de procesos, se pueden consultar en la Tabla 2 que, a su vez, es el resultado de la aplicación de los diagramas de la figura 3.

Para demostrar la utilidad práctica de una métrica se deberá realizar una validación empírica realizando pruebas en diferentes proyectos de forma que se pueda comprobar la relación existente entre las métricas señaladas como: NA, NPT, NDPTIn, NDPTOut, NDPT, NDPA y NCA.

2) Medición del proyecto

Antes de hablar de formas para medir un proyecto comenzaremos con la definición de Proyecto.

Según Lewis (2002), un proyecto es un trabajo de múltiples tareas que tiene rendimiento, costo, tiempo y los requerimientos de alcance que se realizan solamente un vez.

Ya más enfocado a lo que es la gestión de proyectos, Presman (2002) hace referencia a un documento sobre proyectos de software realizado por John Rell en el que define las diez señales que indican que un proyecto de sistemas de información está en peligro:

1. La gente del software no comprende las necesidades de los clientes.
2. El ámbito del producto está definido pobremente.
3. Los cambios están mal realizados.
4. La tecnología elegida cambia.
5. Las necesidades del negocio cambian o están mal definidas.
6. Las fechas de entrega no son realistas.
7. Los usuarios se resisten.
8. Se pierden los patrocinadores o nunca se obtuvieron adecuadamente.
9. El equipo del proyecto carece del personal con las habilidades apropiadas.
10. Los gestores y los desarrolladores evitan buenas prácticas y sabias lecciones.

Se llega a la conclusión de que para gestionar un proyecto de software con éxito, debemos comprender qué puede ir mal, para evitar esos problemas y cómo hacerlo bien.

La medición del proyecto y sus recursos asociados constituye el elemento principal sobre el que se basa el estudio de las métricas del proceso de software. Cuando se mide el proyecto el objetivo fundamental que se pretende es el de reducir el coste total y el tiempo de desarrollo del mismo (Piattini et al., 2008).

Las métricas del proceso de software se utilizan para propósitos estratégicos. Las medidas del proyecto de software son tácticas. Esto es, las métricas de proyectos y los indicadores derivados de ellos los utilizan un gestor de proyectos y un equipo de software para adaptar el flujo del trabajo del proyecto y las actividades técnicas. Pressman (2002).

Estrategia

En un proceso regulable, conjunto de las reglas que aseguran una decisión óptima en cada momento.

Táctica

Método o sistema para ejecutar o conseguir algo.

Las métricas de proceso, entonces, son las que nos sirven en cada uno de los procesos de la ejecución de los sistemas de software proporcionándonos indicadores. La medición de proyecto nos sirve para tomar las decisiones que sean necesarias para que el proyecto pueda realizarse; claro está que para esto hace uso de los indicadores proporcionados por las métricas del proceso y, a su vez, crea indicadores pero a nivel de proyecto.

Pressman (2002) menciona el uso de las métricas en las siguientes etapas del proyecto:

Métricas de estimación: la primera aplicación de métricas de proyectos en la mayoría de los proyectos de software ocurre durante la estimación. Generalmente, se hace uso de las métricas recopiladas de proyectos anteriores que se usan como base para la medición del esfuerzo y del tiempo para el actual trabajo del software. A medida que se avanza en el desarrollo del proyecto se compara el esfuerzo y el tiempo consumido contra las estimaciones iniciales.

Métricas técnicas: A medida que comienza el trabajo técnico, otras métricas de proyectos van adquiriendo significado. Se miden los índices de producción representados mediante páginas de documentación, las horas de revisión los puntos de función y las líneas fuente entregadas. También, se sigue la pista de los errores detectados durante todas las tareas de ingeniería del software. Todo lo anterior para evaluar la calidad del diseño y para proporcionar indicadores que influirán en el enfoque tomado para la generación y prueba del código.

Otros autores como Putman y Myers(2003) establecen los siguientes aspectos por medir para la gestión de proyectos:

Cantidad de funcionalidad, se obtiene a través de las métricas de tamaño (LOC, puntos de función, etc.).

Productividad, relación entre funcionalidad producida en el tiempo y el esfuerzo dedicado.

Tiempo/calendario, duración del Proyecto (generalmente en meses de calendario).

Esfuerzo, cantidad de trabajo en personas/mes.

Fiabilidad, expresada en ratio de defectos (o su métrica recíproca MTTD – Tiempo promedio entre defectos).

Para la estimación del tamaño de software, se destaca la métrica de “Punto de Función”, para la estimación de costos de un proyecto se destacan los modelos COCOMO (Constructive COst Model) y sus posteriores mejoras COCOMO II o también el modelo SLIM.

Los métodos de estimación que señalaremos a continuación son los que representan a los más utilizados en la práctica por los jefes de los proyectos.

Existen dos grandes grupos de métodos de estimación: métodos heurísticos y métodos parámetricos.

Los **métodos heurísticos** se basan en la práctica, en la experiencia de profesionales para encontrar soluciones a los problemas frecuentes.

Entre los métodos heurísticos más representativos están:

- Método basado en la experiencia o juicio experto.
- Método por analogía. Utiliza la experiencia adquirida en proyectos anteriores para comparar con proyectos similares el proyecto por desarrollar.
- Método ascendente, separa al proyecto en componentes y estima por separado, para, luego combinar los resultados para obtener la estimación completa.

- Método descendente, estima de arriba abajo, primero los módulos principales luego submódulos y, finalmente, funciones individuales.
- Método algorítimico, se basa en patrones de datos de proyectos anteriores que son transformados a fórmulas matemáticas que son probadas y validadas en ensayos experimentales mediante pruebas rigurosas usando datos históricos e investigaciones. Luego, estas fórmulas se utilizan para derivar estimaciones del software.

Los métodos paramétricos permiten realizar aproximaciones al inicio del ciclo de vida del sistema. Los más utilizados son:

- COCOMO (Constructive COst Model) II, basado en el COCOMO propuesto por Barry Boehm, en 1981. El COCOMO II es una mejora que toma el tamaño del software y un conjunto de factores como entrada y da como resultado la estimación del esfuerzo en personas/mes.
- SLIM (Software LIfecycle Management) propuesto por Lawrence Putnam en 1978 quien estudia una base de datos (QSM) de 750 sistemas procedentes de la Air Force Electronic System Division, del Rome Air Development Center y otros sistemas de procedencias diversas. Basado en la cantidad de trabajo que se encuentra en cualquier producto se puede ver como el producto del esfuerzo realizado en un periodo de tiempo.
- Métodos basados en el cálculo de los puntos de función. Métodos que sirven para estimar el tamaño funcional de un producto de software basado en los requisitos del usuario.

3) Medición del producto

Comenzaré definiendo lo producto y artefacto de software:

Producto

Un producto de software es un producto diseñado para un usuario. (Lewis, 1994)

Artefacto de software

(Software artefact) Cualquier cosa que resulte del proceso de desarrollo de software; por ejemplo: documentos de requisitos, especificaciones, diseños, software, etc. (ITI, 2009).

Los procesos de software son los que generan los productos de software. Comprenden todos los artefactos de software, desde la documentación hasta programas o sistemas completos, en general todas las salidas del proceso de software en cualquier etapa del ciclo de vida del software.

El objetivo de la medición de software es evaluar la calidad de los entregables que, en este caso, son los productos de este.

Piattini et al. (2008) confeccionan una tabla en la que muestra una sinopsis de las métricas relevantes y su clasificación:

Denominación	**Métricas**	
Clásicas	A nivel de código	Tamaño (LOC, CLOC, NCLOC, DCD) Longitud del programa (LT, SIZE1) Métricas Halstead Ciencia Software Complejidad (complejidad ciclomática) McCabe, Fan-In, Fan-Out, Complejidad de un módulo)
Sistemas OO	A nivel de diseño	Métricas MOOSE Métricas MOOD Métricas de Lorenz y Kidd

	A nivel conceptual	Métricas para casos de uso Métricas diagrama de clases Métricas diagramas de transición de estados
Bases de Datos	A nivel conceptual	Métricas para modelos ER
	A nivel lógico	Métricas bases datos relacionales Métricas almacenes de datos
Métricas para Web	Clasificación de varias propuestas de métricas, según el modelo de calidad WQM	

Tabla 3. Clasificación de las métricas de producto.

Fuente: Piattini et al. (2008).

3.1 Métricas clásicas

Pretenden medir la longitud, tamaño, complejidad del código cuya unidad de medida son las líneas de código.

Una de las métricas más representativas para esta clasificación es la de Líneas de Código (LOC, Lines of Code). Consiste simplemente en contar las líneas de código.

Esta métrica presenta algunos problemas al definir una línea de código, que varía de acuerdo con las necesidades del que está utilizando está métrica. Debe establecer de antemano si en el conteo total incluirá las líneas en blanco, los comentarios, las declaraciones de datos, las líneas de código que contiene instrucciones separadas.

Por ejemplo, si lo que se quiere es medir el nivel de comentarios en el código se pueden tomar las líneas de comentario.

Además, existen otras métricas que se miden de acuerdo con las sentencias de código, conteo de puntos y coma, *tokens*, grafos o caminos linealmente independientes de un programa, etc.

Cabe destacar que las métricas clásicas están muy ligadas a las líneas de código y de acuerdo con la herramienta de software que se utilice para el desarrollo de software y, en esencia, las métricas nos sirven para comparar entre diferentes productos y no puede comparar los LOC de un producto desarrollado, por ejemplo, en Pascal y otro en ensamblador.

3.2 Métricas para sistemas OO (orientados a objetos)

Al tratarse de software desarrollado según el paradigma de orientación a objetos difiere del desarrollo tradicional por lo cual surgieron otras métricas específicas para sistemas OO aunque también se pueden medir con métricas clásicas.

Una de las métricas de esta clasificación más difundidas es la Métrica MOOSE que está sustentada por numerosos trabajos empíricos como la propensión a errores o mantenibilidad de clases.

La métrica de MOOSE está compuesta por otras seis métricas:

1. Métodos ponderados por clase
2. Profundidad del árbol de herencia de una clase
3. Número de hijos
4. Acoplamiento entre objetos
5. Respuesta de una clase
6. Falta de cohesión en los métodos

En general, estás métricas son muy propios de la programación orientada a objetos y que pretende medir su eficiencia.

La métrica de MOOD es otra forma de medir los sistemas de OO y su principal objetivo es medir los mecanismos del paradigma OO como ser polimorfismo, herencia, encapsulación y paso de mensajes. En resumen, asegurarse de que se esté aprovechando al máximo los mecanismos del paradigma al máximo para contribuir con la calidad de software.

Las métricas de Lorenz y Kidd también llamadas “métricas de diseño” porque están orientadas a medir las características estáticas del diseño de un producto de software. Los autores las clasifican en: métricas de tamaño, métricas de herencia y métricas de características internas de las clases.

3.3 Métricas para bases de datos

Hoy en día se maneja grandes cantidades de datos almacenados de los cuales depende la supervivencia de muchas empresas porque les ayuda en la toma de decisiones. Hoy en día la información es poder; por lo tanto, es necesario asegurar la calidad de los datos y de la información, la calidad de una base de datos que depende, a su vez, de la calidad del SGBD (Sistema Gestor de BD) ya sea relacional, OO, objeto-relacional, multidimensional, XML, etc.

Métricas conceptuales para base de datos

El objetivo de las métricas conceptuales de las bases de datos es permitir al analista realizar comparaciones cuantitativas entre diferentes alternativas de diseño y permitir la selección objetiva entre los diferentes modelos conceptuales; también, permite evaluar la calidad del modelo conceptual durante la etapa de modelado.

Las métricas de Moody consisten en la definición de 25 métricas para evaluar la calidad de los modelos de datos.

Entre los factores de calidad de modelos ER se encuentran:

- Compleción

- Integridad
- Flexibilidad
- Comprensibilidad
- Corrección
- Simplicidad
- Integración
- Implementabilidad

Cada factor cuenta con una serie de preguntas cuya respuesta debe ser cuantitativa. Por ejemplo, en el factor de simplicidad se tienen las siguientes métricas:

N° de entidades

N° de entidades y relaciones

N° de constructores

También, tenemos la Métrica de Piattini et al. (2008), que tiene como objetivo central de medición la mantenibilidad de los modelos ER.

Métricas para modelos lógicos de bases de datos

El único criterio o métrica que se toma para los modelos lógicos de base de datos es la teoría de la normalización.

Entre las métricas para los modelos lógicos se destaca la propuesta por Calero, Ruíz y Piattini (2005), que tiene como objetivo evaluar la mantenibilidad de los modelos relacionales.

Métrica que propone Calero:

- Número de atributos de una tabla.
- Número de claves ajenas de una tabla.

- Profundidad del árbol referencial de una tabla.
- Ratio de claves ajenas de una tabla.
- Número de tablas del esquema.
- Cohesión del esquema.
- Ratio de normalidad.
- Número de atributos del esquema.
- Número de claves ajenas del esquema.
- Profundidad del árbol referencial del esquema.
- Ratio de claves ajenas del esquema.

Métricas para almacenes de datos (data warehouse)

Los almacenes de datos proporcionan información histórica de la empresa; por lo tanto, se debe garantizar la calidad de los almacenes de datos empezando desde el desarrollo para no dejar pasar ningún atributo que haga falta en las tablas y que después sea necesario incluir cuando ya se ha almacenado información por algún tiempo, lo que provoca que no se tendrá información completa.

Existen las siguientes métricas para almacenes de datos:

- Métricas a nivel de tabla
- Métricas a nivel de estrella
- Métricas a nivel de esquema

Estas métricas se han validado empíricamente mediante diversos experimentos en los que se han obtenido las métricas NFT (Número de tablas de hechos), NT (Número de tablas) y NFK (Número de claves ajenas) por lo que parecen ser buenos indicadores de la complejidad de los almacenes de datos (Piattini et al., 2008).

3.4 Métricas para sistemas web

Internet se ha convertido en el nuevo canal de comunicación, por lo tanto la tecnología web en los sistemas de información ha tomado un papel protagónico

pero se ha visto la necesidad de establecer estándares mínimos de calidad por lo que se han venido desarrollando propuestas para mejorar la calidad de los sitios web representados en metodologías, guías, métricas, modelos de estimación y marcos de calidad.

Calero et al. (2005) realizaron el modelo WQM (Web Quality Model) para clasificar las distintas métricas existentes.

3.4.1 WQM (Web Quality Model)

Consiste en un modelo tridimensional cuyas dimensiones son las siguientes: componentes del sitio web, características de calidad, procesos del ciclo de vida.

1) Los componentes del sitio web: contenido, navegación y presentación.

2) Las características de calidad: funcionabilidad, fiabilidad, usabilidad, eficiencia, portabilidad y mantenibilidad (combinación del estándar ISO/IEC 9126 con el modelo QUINT2).

3) Los procesos del ciclo de vida: no organizacionales (Desarrollo, explotación y mantenimiento) y organizacionales (Esfuerzo, reutilización).

Las distintas métricas de tecnología web existentes en el mercado se pueden clasificar en las dimensiones del modelo, de manera que se puede detectar que partes del cubo tienen carencia de métricas. El cubo nos sirve para ubicar en que dimensión se encuentra la métrica por utilizar.

Conclusiones

Lo que se puede rescatar de este trabajo es que existen innumerables métricas de software en el mercado y no podría ser de otra manera ya que a través del tiempo y la evolución de la tecnología se han ido desarrollando métricas de acuerdo con las necesidades del software en cuestión es por eso que no se puede elegir una métrica de software al azar porque existen “métricas de métricas”, si se hace un mal uso de estas o por no adaptarse a las necesidades, los resultados arrojados pueden ser de poco valor.

Por lo tanto, es importante que antes de inclinarse por cierta métrica, se debe tener en cuenta lo siguiente:

- Tener claro lo que se quiere medir.
- Conocer el grado de madurez de la empresa, este aspecto es necesario para aplicar ciertas métricas. Por ejemplo si se trata de una empresa desarrolladora de software que apenas está sacando su primer producto de software, se tienen muy pocos indicadores para aplicarle una métrica de Proceso.
- Definir si las métricas se van a utilizar a nivel de proceso, proyecto o producto.
- Determinar en qué fase del ciclo de vida del desarrollo de sistemas nos encontramos.
- Que tipo de paradigma de programación se utilizará para la construcción de software (procedimientos, funcional, lógico, orientado a objetos).
- Si se trata de una aplicación de escritorio o web.
- Analizar si la métrica se adapta a lo que queremos medir de acuerdo con los puntos señalados anteriormente.

Gracias a las métricas se obtiene una mayor información que posibilita tomar decisiones que permiten mejorar el nivel de calidad de software; objetivo que, desde los inicios de la informática, se ha perseguido. En nuestros días es difícil

imaginar el mundo actual sin la tecnología informática; así paulatinamente se ha ido creando una dependencia de la tecnología para el buen funcionamiento de los negocios, por lo que es vital la calidad en los sistemas informáticos para que las empresas puedan continuar compitiendo en el mercado.

El aplicar las métricas cumpliendo objetivos claros es una necesidad para mejorar el proceso en su conjunto y de ninguna manera se deben utilizar los resultados solamente para medir el rendimiento de las personas, de una sección, departamento, etc., sino para encontrar las debilidades y ver la manera de corregirlas.

Bibliografía

Ferreira M., García F., Ruíz F., Bertoa M., Calero C., Vallecillo A., Piattini M. y Mora B. (2006). Medición del Software Ontología y Metamodelo. Informe Técnico, Universidad de Castilla, La Mancha, España.

Calero C., Ruíz J. y Piattini M., (2005). *Classifying web metrics using the web quality model*. Artículo publicado online, Emerald.

Instituto Tecnológico de Informática. (2009). *Glosario de Términos*. Recuperado el 20 de octubre de 2009, de http://squac.iti.upv.es/glosario-calidad/task,showpart/part,A/catid,30/

La Enciclopedia, (2004). Colombia: Salvat Editores.

Lewis, G. (1994). *What is Software Engineering?* . Artículo no publicado, DataPro (4015). Feb 1994. pp. 1-10.

Lewis, J. (2002). *Fundamentals of project management : developing core competencies to help outperform the competition*. Estados Unidos : Amacon.

Piattini, M., Garcia, F., Garzás J., y Genero M., (2008). *Medición y estimación del software: Técnicas y métodos para mejorar la calidad y la productividad*. México: Alfaomega.

Pressman, R. (2002). *Ingeniería del software : un enfoque práctico*. España : McGraw Hill.

Pressman, R. (2006). Ingeniería del software: un enfoque práctico. México: McGraw Hill Interamericana.

Riguzzi F., (1996). A Survey of Software Metrics. Artículo no publicado, Università degli Studi di Bologna, Italia.

Software Engineering Institute. (2009). *2006 State of Software Measurement Practice Survey*. Recuperado el 2 de octubre de 2009, de http://www.sei.cmu.edu/library/abstracts/presentations/stateofsurvey.cfm