Departamento de Informática

Una Ontología de la Medición del Software

F. García[1], F.Ruiz[1], M. F. Bertoa[2], C.Calero[1], M.Genero[1], L. Olsina[3], M. Martín[3],
C. Quer[4], N. Tondori[5], S. Abrahao[5], A. Vallecillo[2], M. Piattini[1]

[1]Universidad de Castilla-La Mancha, España.
[2]Universidad de Málaga, España.
[3]Universidad Nacional de La Pampa, Argentina.
[4]Universidad Politécnica de Cataluña, España.
[5]Universidad Politécnica de Valencia, España.

Informe Técnico UCLM DIAB-04-02-2
Febrero 2004

Universidad de
Castilla-La Mancha

Dep. Legal:
I.S.B.N.: 84-688-5427-1

“ Medir lo que es medible, y hacer medible lo que no lo es”

Galileo Galilei

ÍNDICE

1. Introducción

La medición de software es una disciplina relativamente joven, y para la cual no existe aún un consenso general sobre la definición exacta de los conceptos y terminología que maneja. A pesar de contar con diversos estándares internacionales que tratan de normalizar estos temas, se han detectado ciertas lagunas e inconsistencias en los términos que dichos estándares definen, como son por ejemplo los conceptos de medida, métrica, medición, indicador, etc. La situación no es mucho mejor en los contextos académicos e industriales, en donde las distintas propuestas de modelos de medición de software tampoco han logrado consensuar una terminología coherente y ampliamente aceptada entre toda la comunidad científica.

Distintos grupos de investigación latino-americanos dedicados a temas de la medición del software se han dado cuenta de la necesidad de contar con una terminología bien definida desde donde poder abordar su investigación. Eso ha dado lugar a la aparición de diversas propuestas de meta-modelos y ontologías de la medición, realizadas de forma individual por cada grupo [Genero et al, 2003; Olsina et al., 2002; Burgués y Franch, 2003].

En Noviembre de 2002 surgió la idea de tratar de consensuar un glosario de términos y una ontología de la medición del software entre los grupos de diferentes universidades que trabajaban en estos temas, tanto de España como de Argentina. A raíz de esa decisión se celebraron tres

reuniones de trabajo (en Málaga, en marzo de 2003; en Ciudad Real, en julio de 2003; y en Alicante, en noviembre de 2003).

Este documento presenta las conclusiones obtenidas tras dichas reuniones, en forma de un glosario de términos en español con los principales conceptos que intervienen en la medición del software, y un diagrama UML que muestra las relaciones entre tales conceptos.

La propuesta aquí presentada está basada en cuatro conceptos fundamentales:

- la *forma de medir* (que se corresponde básicamente con el concepto de *measurement* de ISO-15939, y que va a cubrir: el *método de medición* para *métricas directas*, la *función de cálculo* utilizada en *métricas indirectas*, y el *modelo de análisis* utilizado en los *indicadores*);
- la acción de medir (denominada *medición*);
- el resultado de la medición (denominada *medida*); y
- el concepto de *métrica* (definido como "una *forma de medir* + una *escala*")

Es importante señalar que el objetivo principal del proyecto no se ha cumplido del todo, pues no se ha conseguido alcanzar un consenso completo entre todos los participantes sobre algunos de los aspectos del glosario. La principal causa de esta falta de consenso se debe, sobre todo, a las diferentes visiones y enfoques de cada uno de los grupos. Sin embargo, en la última reunión se decidió al menos elaborar el presente

documento con lo producido hasta el momento, y publicarlo como informe técnico que recoja el resultado del proyecto. El objetivo es poder disponer de un documento concreto que poder referenciar en trabajos futuros, y a partir de aquí citarlo como propuesta si se está de acuerdo con él, o bien para poder discrepar de alguna de sus ideas.

Este documento está estructurado en 4 secciones, de las cuales la primera es esta introducción, la sección 2 define los términos que conforman el glosario, la sección 3 contiene el diagrama UML con las relaciones entre dichos términos, y finalmente la sección 4 presenta un ejemplo que pretende aclarar y justificar algunas de las decisiones tomadas.

2. Términos del Glosario

2.1. Categoría de Entidad

Definición

Una colección de *entidades* caracterizadas por satisfacer un cierto predicado común.

Relaciones

- Una *categoría de entidad* puede "incluir a" una o varias *categorías de entidad*, y puede "estar incluida en" una o varias *categorías de entidad*.
- Una *categoría de entidad* tiene uno o varios atributos.
- Una *categoría de entidad* puede tener definidos varios *modelos de calidad*.

Ejemplos

- "Programas", "Programas en C", "Componentes software", "Componentes COTS", "Componentes software para comunicaciones", etc.
- Procesos, productos, servicios, proyectos, o recursos son ejemplos de categorías de entidad.

Notas

- Pueden definirse jerarquías entre categorías de entidad.

2.2. Entidad

Definición

Un objeto que va a ser caracterizado mediante una *medición* de sus *atributos* [ISO-15939]

Relaciones

- Una *entidad* puede pertenecer a una o más *categorías de entidad*.
- Una *medición* se realiza sobre los *atributos* de una *entidad*

Ejemplos

- El programa "holaMundo.c".

Notas

- Una entidad puede ser un proceso, un producto, un servicio, un proyecto, o un recurso concreto.
- Una entidad puede ser física –tangible– (p.e. un ordenador) o abstracta (p.e. un programa en C).

2.3. Atributo

Definición

Una propiedad mensurable, física o abstracta, que comparten todas las *entidades* de una *categoría de entidad*.

Relaciones

- Un atributo solo puede pertenecer a una categoría de entidad.
- Una medición se realiza sobre los atributos de una entidad
- Un atributo tiene definida cero, una o varias métricas.
- Un atributo está relacionado con uno o más conceptos medibles.

Ejemplos

- El *atributo* "tamaño de código fuente", como *atributo* de la *categoría de entidad* "programas en C" va a ser diferente del *atributo* "tamaño de código fuente" de la *categoría de entidad* "programa en Ada".
- El "tamaño" de un "módulo en C" no es el mismo atributo (aunque tenga el mismo nombre) que el "tamaño" de un "diagrama de clases UML".
- Tamaño de código fuente, precio.

Notas

- ISO 9126 habla de atributos *internos* y externos. En esta ontología esta clasificación no se considera. La naturaleza de los atributos viene establecida por el tipo de métrica (directa, indirecta o indicador) definida para llevar a cabo sus mediciones.
- Dos atributos de dos categorías diferentes no son el mismo atributo aunque tengan el mismo nombre (ver lista de ejemplos).

2.4. Forma de Medir

Definición

Conjunto de operaciones cuyo objeto es determinar el valor de una *medida*. Una *forma de medir* puede ser un *método de medición*, *función de cálculo* o *modelo de análisis*.

Relaciones

- Una *forma de medir* es ejecutada en cada *medición*, dependiendo de la *métrica* que calcula.

Ejemplos

- Véase los ejemplos de *método de medición*, *función de cálculo* o *modelo de análisis*, ya que la *forma de medir* es una generalización de ellos.

2.5. Métrica

Definición

Una *forma de medir* (*método de medición*, *función de cálculo* o *modelo de análisis*) y una *escala,* definidas para realizar *mediciones* de uno o varios *atributos* .

Relaciones

- Una métrica está definida para uno o más *atributos*
- Dos *métricas* pueden relacionarse mediante una función de transformación. El tipo de dicha función de transformación va a depender del tipo de *escala* de ambas métricas.
- Una métrica puede expresarse en una *unidad* (sólo para métricas cuya escala sea de tipo intervalo o ratio)

Ejemplos

- La métrica "líneas de código" puede ser definida para realizar mediciones del "tamaño" de un "módulo en C" y para realizar mediciones del "tamaño" de un "programa en Ada".
- Véase el ejemplo descrito en la sección 4.

Notas

- ISO 9126 e ISO 14598-1:1999 hablan de *métricas* internas o externas, que en esta ontología no se consideran porque no se distingue entre atributos internos y externos.

2.6. Métrica Directa

Definición

Una *métrica* de la cual se pueden realizar *mediciones* sin depender de ninguna otra *métrica* y cuya *forma de medir* es un *método de medición*.

Relaciones

- La forma de medir una métrica directa es un método de medición.
- Una métrica directa puede ser utilizada en funciones de cálculo.

Ejemplos

- LCF (líneas de código fuente escritas).
- HPD (horas-programador diarias).
- CHP (coste por hora-programador, en unidades monetarias).

Notas

- Las métricas directas se corresponden, aunque no de manera exacta, con lo que ISO 15939 denomina "base measure".

- [ISO 14598-1:1999] define “direct measure” (y no “direct metric”).

2.7. Métrica Indirecta

Definición

Una *métrica* cuya *forma de medir* es una *función de cálculo,* es decir, las *mediciones* de dicha *métrica* utilizan las *medidas* obtenidas en *mediciones* de otras *métricas directas* o *indirectas*.

Relaciones

- La forma de medir una métrica indirecta es una función de cálculo.
- Una métrica indirecta puede usarse en una función de cálculo.

Ejemplos

- HPT (horas-programador totales).
- LCFH (líneas de código fuente por hora de programador).
- CTP (coste total actual del proyecto, en unidades monetarias).
- CLCF (coste por línea de código fuente).

Notas

- En ISO 15939 sólo se permite derivar métricas indirectas a partir de “2” o más “base measures” (aquí métricas directas)". En esta propuesta se considera que la forma de medir una métrica indirecta puede usar una o más métricas directas o indirectas.
- [ISO 14598-1:1999] define “indirect measure”.

2.8. Indicador

Definición

Una *métrica* cuya *forma de med*ir es un *modelo de análisis*, es decir, las *mediciones* de dicha *métrica* utilizan las *medidas* obtenidas en las *mediciones* de otras *métricas (directas, indirectas o indicadores)* junto con *criterios de decisión*.

Relaciones

- Un indicador satisface necesidades de información.
- Un indicador es definido por un modelo de análisis.

Ejemplos

- PROD (productividad de los programadores).
- CAR (carestía del proyecto).

2.9. Medición (Acción de Medir)

Definición

La acción que permite obtener el valor de una *medida* para un *atributo* de una *entidad*, usando una *forma de medir*.

Relaciones

- Cada *medición* produce una *medida*.
- Una *medición* usa una *métrica*, la cual debe estar definida para el atributo objeto de la medición.
- Una *medición* es llevada a cabo usando una *forma de medir*. Esta forma de medir es la que define la *métrica* usada en la *medición*.

- Una *medición* se realiza para un *atributo* de una *entidad.* El atributo ha de estar definido para la categoría a la que pertenece dicha entidad.

Ejemplos

- Acción consistente en usar la forma de medir "contar el número de líneas de código" para obtener la medida del atributo "tamaño" de la entidad "módulo nominas.c".

Notas

- En un proceso de medición, al menos siempre interviene una métrica directa y un indicador (después). Lo que puede no haber es alguna métrica indirecta.

2.10. Medida

Definición

Resultado de una *medición*.

Relaciones

- Una *medida* es el resultado de una *medición*

Ejemplos

- 35.000 líneas de código, 200 páginas, 50 clases.
- 5 meses desde el comienzo al fin del proyecto.
- 0,5 fallos por cada 1.000 líneas de código.

Notas

- La medida sólo aparece cuando se realiza una medición.

2.11. Método de Medición

Definición

La *forma de medir* una *métrica directa*. Secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar *mediciones* de un *atributo* respecto de una *escala* específica.

Relaciones

- Un método *de medición* define una o más *métricas directas*.
- Un método *de medición* puede usar *Instrumentos de Medición*.

Ejemplos

- Contar líneas de código.
- Anotar cada día las horas dedicadas por los programadores al proyecto.
- Valorar el grado de dificultad de un problema.

Notas

- El tipo de *método de medición* va a depender de la naturaleza de las operaciones utilizadas para realizar *mediciones* del *atributo*: podemos distinguir entre *métodos* subjetivos y objetivos. Subjetivo: cuando el *método de medición* supone un juicio realizado por un ser humano. Objetivo: cuando el *método de medición* está basado en métodos numéricos.

2.12. Instrumento de Medición

Definición

Instrumento que asiste o es útil a un *método de medición*.

Relaciones

- Un instrumento *de medición* asiste a uno o más *métodos de medición*.

Ejemplos

- Un reloj es un *instrumento de medición* que asiste al *método de medición* contar el paso del tiempo.
- Una herramienta CASE que sirva para contar líneas de código es un *instrumento de medición* que asiste al *método de medición* contar líneas de código.

Notas

- Un instrumento software de medición es un caso particular de un *instrumento de medición.*

2.13. Necesidad de Información

Definición

Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).

Relaciones

- Una *necesidad de información* se asocia a un *concepto medible.*
- Una *necesidad de información* se satisface con uno o más *indicadores*.

Ejemplos

- Conocer el nivel de productividad de los programadores del proyecto en comparación con lo habitual en otros proyectos en la organización (ver ejemplo desarrollado).
- Determinar si los recursos del proyecto son adecuados para satisfacer sus objetivos.
- Evaluar el rendimiento de la actividad de codificación.
- Evaluar si un producto software satisface las expectativas del cliente.

2.14. Concepto Medible

Definición

Relación abstracta entre *atributos* y *necesidades de información*

Relaciones

- Un concepto medible se asocia a una o varias necesidades de información.
- Un concepto medible puede incluir otros conceptos medibles.
- Un concepto medible relaciona uno o más atributos.
- Un concepto medible está incluido en uno o más *modelos de calidad*.

Ejemplos

- Ratio de productividad de un equipo de desarrollo frente a un grado de productividad objetivo.
- Adecuación de la tecnología.

Notas

- Un *concepto medible* es el encargado de indicar cómo satisfacer las necesidades de información. Para ello indica los atributos que hay que medir para satisfacer una o más necesidades de información [ISO 15939]

2.15. Modelo (de calidad)

Definición

Un marco conceptual que especifica una serie de *conceptos medibles* y sus relaciones, para una determinada *categoría de entidad*.

Relaciones

- Un modelo (de calidad) está definido para una determinada categoría de entidad.
- Un modelo (de calidad) evalúa uno o varios *conceptos medibles*.

Ejemplos

- Modelo de calidad para productos software de ISO 9126.
- Factores de calidad de McCall [McCall 1977].

2.16. Unidad

Definición

Una cantidad particular, definida y adoptada por convención, con la que poder comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular [ISO-15939]

Relaciones

- Una unidad sirve para expresar una o varias métricas cuyo tipo de escala sea intervalo o ratio.

Ejemplos

- Kilómetros, metros, millas.
- Líneas de código, Páginas, Persona-mes,
- Número de módulos, Número de clases,...
- Dólares, Pesetas, Horas, días, Meses, Años,...

2.17. Escala

Definición

Un conjunto de valores con propiedades definidas [ISO 14598-1]

Relaciones

- Toda escala es de un cierto "*Tipo de Escala*".

Ejemplos

- Los valores que puede tomar la métrica "lenguaje de Programación usado en un proyecto": Pascal, C, Java (Nominal).
- El nivel de madurez CMM: 1, 2, 3, 4, 5 (Ordinal).
- El tamaño de un código software expresado en líneas de código: Conjunto de los números naturales (Ratio).
- La temperatura expresada en grados centígrados o grados Fahrenheit (Intervalo).

Notas

- Las escalas están definidas por un homomorfismo y los tipos de escala están definidos por las transformaciones admisibles [Zuse, 1997].
- ISO 9126 define la escala como un conjunto de valores con unas propiedades definidas. Ejemplos de tipos de escalas son: escala nominal, que corresponde a un conjunto de categorías, escala ordinal que corresponde a un conjunto ordenado de puntos, escala intervalo que corresponde a una escala ordenada con puntos equidistantes, escala ratio que no sólo tiene puntos equidistantes sino que también tiene un cero absoluto. Las métricas en escalas nominal u ordinal producen datos cualitativos. Las métricas en escala intervalo o ratio producen datos cuantitativos [ISO 9126]

2.18. Tipo de Escala

Definición

Indica la naturaleza de la relación entre los valores de la escala [ISO 15939]

Relaciones

- A un *Tipo de Escala* pertenecen una o más *Escalas*.

Ejemplos

- Nominal, Ordinal, Intervalo, Ratio y Absoluta

Notas

- Cada *tipo de escala* determina las transformaciones admisibles, el tipo de operaciones matemáticas y los análisis estadísticos que

pueden aplicarse sobre el conjunto de valores de la escala. [Zuse, 1997]

– No existe definición de tipo de escala como tal en la ISO 9126 aunque sí comenta los tipos de escala posibles cuando define escala.

2.19. Función de Cálculo

Definición

La *forma de medir* una *métrica indirecta*. Algoritmo o cálculo realizado para combinar dos o más métricas directas y/o indirectas.

Relaciones

– Una función de cálculo usa cero o más métricas directas.
– Una función de cálculo usa cero o más métricas indirectas.
– Una función de cálculo utiliza al menos una métrica (sea directa o indirecta).
– Una función de cálculo define una o más métricas indirectas.

Ejemplos

– Véanse los ejemplos descritos en la sección 4.
– LCFH = LCF / HPT [métrica indirecta definida en base a 2 métricas directas].
– $\mathsf{HPT} = \sum_{\text{días}} \mathsf{HPD}$ [métrica indirecta definida en base a sólo 1 métrica directa].
– CTP = CHP * HPT [métrica indirecta definida en base a 2 métricas, una directa y otra indirecta].

2.20. Criterio de Decisión

Definición

Valores umbrales, objetivos, o patrones, usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado.

Relaciones

- Un *criterio de decisión* es utilizado en uno o más *modelos de análisis*.

Ejemplos

- Véase los ejemplo descritos en la sección 4.
- LCFH/LCFHvm < 0’70 => PROD=’muy baja’.
- 0’70 ≤ LCFH/LCFHvm < 0’90 => PROD=’baja’.
- 0’90 ≤ LCFH/LCFHvm < 1’10 => PROD=’normal’.
- 1’10 ≤ LCFH/LCFHvm < 1’30 => PROD=’alta’.
- 1’30 ≤ LCFH/LCFHvm => PROD=’muy alta’.

Notas

- La ISO 15939 habla del nivel de “bondad” del valor de una métrica.

2.21. Modelo de Análisis

Definición

La *forma de medir* un *indicador*. Algoritmo o cálculo realizado para combinar una o más *métricas* (directas, indirectas o indicadores) con *criterios de decisión* asociados.

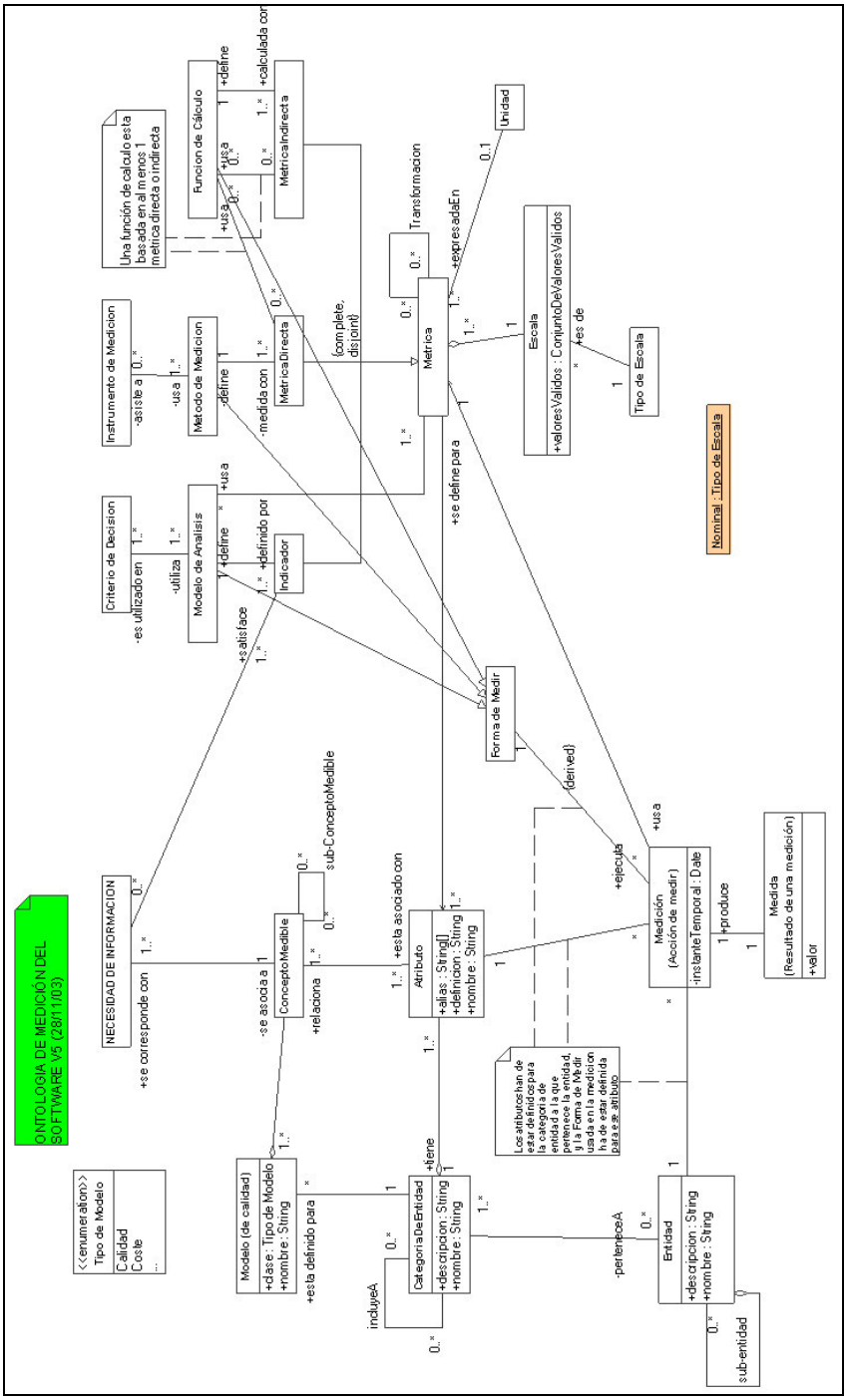
Relaciones

- Un modelo de análisis define uno o más indicadores.
- Un modelo de análisis utiliza uno o más criterios de decisión.
- Un modelo de análisis usa una o más métricas.

Ejemplos

- Véanse los ejemplos descritos en la sección 4.
- Modelo de Análisis para obtener la métrica PROD. Utiliza los valores de las métricas LCF, HPT, LCFH y CTP para establecer un valor cualitativo de la productividad de los programadores en este proyecto. El modelo se basa en extraer de una base histórica de proyectos de la organización los valores medios de LCF, HPT, LCFH (LCFHvm) y CTP del subconjunto de proyectos similares (aquellos que tienen LCF entre el 80% y el 120%)

Figura 1. Representación en UML de la Ontología de la Medición



3. Diagrama Ontológico

En la Figura 1 se muestra el diagrama UML que representa los conceptos descritos en el glosario de la sección anterior, y las relaciones entre ellos.

4. Ejemplo de Métricas (Directas, Indirectas e Indicadores)

Supongamos una organización que lleva a cabo un proyecto de desarrollo de un software X. En un determinado momento el responsable del proyecto necesita saber si la productividad es adecuada, es decir, la necesidad de información es conocer el nivel de productividad de los programadores del proyecto en comparación con lo habitual en otros proyectos en la organización.

Las métricas a utilizar podrían ser:

Directas:

- LCF (líneas de código fuente escritas). El método de medición es contar las líneas utilizando como instrumento una herramienta CASE.
- HPD (horas-programador diarias). El método de medición es que el responsable del proyecto anota cada día las horas dedicadas por los programadores al proyecto.
- CHP (coste por hora-programador, en unidades monetarias). El método de medición es consultar el plan del proyecto, donde se

tuvo que indicar este valor, previa consulta a un responsable de personal.

Indirectas:

- HPT (horas-programador totales). La función de cálculo es un sumatorio de las HPD de cada día: $HPT = \sum_{días} HPD$ [métrica indirecta definida en base a sólo 1 métrica directa].
- LCFH (líneas de código fuente por hora de programador). La función de cálculo es una simple división: LCFH = LCF / HPT [métrica indirecta definida en base a 2 métricas directas].
- CTP (coste total actual del proyecto, en unidades monetarias). La función de cálculo establece que el CTP es el producto del coste unitario de cada hora por el total de horas empleadas: CTP = CHP * HPT [métrica indirecta definida en base a 2 métricas, una directa y otra indirecta].
- CLCF (coste por línea de código fuente). CLCF = LCF/CTP.

Indicadores:

- PROD (productividad de los programadores). El modelo de análisis utiliza los valores de las métricas LCF, HPT, LCFH y CTP para establecer un valor cualitativo de la productividad de los programadores en este proyecto. El modelo se basa en extraer de una base histórica de proyectos de la organización los valores medios de LCF, HPT, LCFH (LCFHvm) y CTP del subconjunto de proyectos similares (aquellos que tienen LCF entre el 80% y el 120%). Los criterios de decisión establecidos son:
  - LCFH/LCFHvm < 0’70 => PROD=’muy baja’.

- 0’70 ≤ LCFH/LCFHvm < 0’90 => PROD=’baja’.
- 0’90 ≤ LCFH/LCFHvm < 1’10 => PROD=’normal’.
- 1’10 ≤ LCFH/LCFHvm < 1’30 => PROD=’alta’.
- 1’30 ≤ LCFH/LCFHvm => PROD=’muy alta’.

- CAR (carestía del proyecto). El modelo de análisis es parecido al anterior. Utiliza los valores de las métricas CLCF y PROD para establecer un valor cualitativo del tipo de carestía del proyecto. De la base histórica de proyectos se obtienen los valores medios de LCF, CLCF, y PROD del subconjunto de proyectos similares. Los criterios de decisión son bidimensionales. La primera dimensión compara el valor de CLCF del proyecto con el valor medio histórico (CLCFvm) para determinar si el coste de cada línea es más alto o bajo de lo normal. La segunda dimensión simplemente compara PROD con PRODvm para ayudar a decidir si la razón de ese coste alto (o bajo) por línea es o no debida a la baja (o alta) productividad. De esta manera, los valores de CAR serán cualitativos del tipo: ‘muy alta por baja productividad’, ‘normal a pesar de productividad baja’, etc.

5. Referencias

[Burgués y Franch, 2003] X. Burgués, X. Franch. "Modelado de la Calidad del Software Mediante una Estructura jerárquica". Actas de las VIII Jornadas de Ingeniería del Software y Bases de Datos, Alicante, Noviembre 2003.

[DRAE, 2001] Diccionario de la Real Academia de la Lengua Española. Vigésimo primera edición. 2001.

[Fenton, 1997] Fenton, N. and Pfleeger, S. L. 1997. Software Metrics: A rigorous approach, 2 ed. Chapman & Hall, London.

[Genero, et al, 2003] M. Genero, F. Ruiz, M. Piattini, F. García y C. Calero. "Towards an Ontology for Software Measurement". 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'2003). Proceedings, ISBN: 1-891706-12-8, pp. 307-317.

[IEEE 610.12:1990] IEEE Std 610.12-1990, IEEE standard glossary of software engineering terminology

[ISO-14598] ISO/IEC 14598-1 "Information Technology—Software Product Evaluation—Part 1: General overview". May 1999.

[ISO-15939] ISO/IEC 15939. Software Engineering – Software Measurement Process. 2002.

[ISO-9126] ISO/IEC 9126-1:2001 "Software Engineering—Product Quality—Part 1: Quality model". June 2001.

[IVM, 1993] International Vocabulary of Basic and General Terms in Metrology, 1993.

[McCall 1977] J.A. McCall, P.K. Richards y G.F. Walters. "Factors in software quality, volume III: Preliminary handbook on software quality for an acquisition manager". Informe técnico RADC-TR-77-369, vol. III, Hanscom AFB, MA 01731, 1977

[McGarry et al., 2002] J. McGarry et al. "Practical Software Measurement. Objective Information for Decision Makers". Addison-Wesley, 2002. (ISBN 0-201-71516-3)

[Olsina et al., 2002] L. Olsina et al. "Un Marco Conceptual para la Definición y Explotación de Métricas de Calidad". Actas de las VII Jornadas de Ingeniería del Software y Bases de Datos, El Escorial, Madrid, Noviembre 2002. .

[SE-CMM] Software Engineering Institute (SEI). The Capability Maturity Model: Guidelines for Improving the Software Process, (1995). In http://www.sei.cmu.edu/cmm/cmm.html

[Zuse, 1997] Zuse, H. 1998. A framework of Software Measurement. Walter de Gruyter, Berlin.

Agradecimientos

Este informe técnico forma parte del proyecto MAS, financiado por la Dirección General de Investigación del Ministerio de Ciencia y Tecnología (TIC 2003-02737-C02-02).