Evaluación de Código Mediante Múltiples Intervalos de Métricas

Carlos López[1], Yania Crespo[2], Esperanza Manso[2], Raúl Marticorena[1]
[1] Universidad de Burgos
{clopezno,rmartico}@ubu.es
[2] Universidad de Valladolid
{yania,manso}@infor.uva.es

Abstract: *Code evolution and maintenance include measurement activities to detect possible flaws and to suggest improvements. An approach to detect anomalies on code entities is based on metric values outside their thresholds. From the 90s, there are plenty of papers that propose general threshold values for a set of metrics. One of the criticisms outlined in the literature to this approach is the difficulty of using these results to other contexts. Threshold values may be influenced by multiple variables, including the nature of the problem solved: exception, graphical interface, model and test. This information can be introduced by the user in the measurement process, making it semiautomatic. To verify this hypothesis, this paper describes a case study where the measurement process is guided by the inspector/assessor, who classifies code entities to measure depending on their nature. The case study result suggests some metric thresholds for different kind of problems and relative to a specific organization.*

Resumen: *La evolución y mantenimiento del código incorporan en su proceso actividades de medición para detectar posibles defectos y proponer mejoras. Una manera de detectar anomalías sobre entidades de código se basa en comprobar que el valor de una métrica está fuera de un intervalo de valores recomendados. En este sentido, desde la década de los 90, existen multitud de trabajos que proponen de manera empírica intervalos recomendables para un conjunto de las mismas. Una de las críticas expuesta en la literatura a este planteamiento es la dificultad de transportar estos resultados a otros contextos. La determinación de los valores que comprenden el intervalo puede estar condicionada por múltiples variables, entre ellas la relacionada con la naturaleza del problema que resuelve: excepciones, interfaces gráficas, modelo, controladores y pruebas. Esta información puede ser incorporada por el usuario en el proceso de medición, transformándolo en semiautomático. Para corroborar esta hipótesis, en este trabajo se define un caso de estudio donde el proceso de medición es guiado por el inspector/evaluador, que clasifica las entidades de código a medir dependiendo su naturaleza. Como resultado del caso de estudio se proponen unos intervalos relativos a las métricas para los diferentes tipos y relativos a una organización concreta.*

Palabras Clave: *Métricas de Código, Intervalos Relativos, Experimento, Proceso de Medición, Herramientas de Medición*

1. Introducción

Desde la década de los 90 las métricas del software y el proceso de medición asociado han captado la atención de la comunidad de la ingeniería del software como medio de cuantificar y controlar la calidad del software [1-4]. La medición es una actividad que se encuentra englobada dentro de un proceso (ver Figura 1), que consiste en derivar un valor numérico para algún atributo de un producto o proceso software.

En este proceso, la detección de entidades defectuosas se basa en la identificación de medidas anómalas sobre las entidades. De manera pragmática la identificación consiste en comprobar si una medida está dentro de unos valores recomendados. De una manera general, este proceso de medición puede aplicarse sobre múltiples productos resultado del proceso de desarrollo del software, dando lugar a modelos de calidad de producto software como el propuesto en la norma ISO 9126 [5].

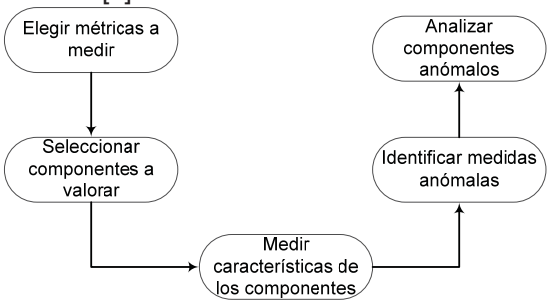

Figura 1 Proceso de medición

Desde un punto de vista más concreto el proceso se puede aplicar sobre el código ya que es un producto en constante evolución y mantenimiento [6]. La aplicación de métricas sobre él, con su posterior interpretación de valores, es una de las técnicas que puede ayudar a evaluar

la calidad del mismo. La evaluación de código mediante métricas no es nueva. Como consecuencia de ello, en la literatura existe un gran número de definiciones de métricas agrupadas por criterios diferentes dependiendo del autor. Por ejemplo, en el paradigma de la orientación a objetos algunos conjuntos de métricas bien conocidos sobre diferentes entidades de código son:

- Sobre clases: Chidamber y Kemerer [7], Lorenz y Kid [8]
- Sobre subsistemas: Robert Martin [9], Brito e Abreu [10]
- Sobre métodos: MCabe[11]
- Otros recogidos en [12]

Pero en la literatura también existen muchas críticas, resultado de su aplicación. Una de ellas es que los valores utilizados para identificar medidas sobre entidades anómalas obtenidas a través de experimentos empíricos están restringidos al contexto de medición, siendo limitada su utilización en otros contextos. Aunque somos conscientes que es muy difícil acotar por completo la heterogeneidad de los diferentes contextos, este trabajo pretende avanzar en este aspecto. Para ello se propone incorporar nuevas tareas en el proceso de medición, basadas en el conocimiento del inspector/evaluador respecto a la identificación de entidades en función de la naturaleza del problema que resuelven. Bajo esta premisa y en un entorno de la orientación a objetos, una selección de criterios de clasificación se puede basar en la utilización de algunos estereotipos sobre clasificadores de UML. Por ejemplo, los estereotipos sobre clases de análisis [13] [14]: entidad, controladores, límites. Además el criterio de clasificación límite se desglosa a su vez en: interfaces de usuario, interfaces de sistema e interfaz de dispositivo. Otros estereotipos interesantes en los clasificadores son los obtenidos como resultados de algunas tareas del proceso de desarrollo como los relacionados con excepciones y pruebas. Esto provoca un desglose de tareas en la actividad de medición obteniendo diferentes intervalos de validación para cada tipo de estereotipo considerado. En la Figura 2 se adapta el nuevo proceso de medición propuesto y se enmarca con un rectángulo el desglose de tareas.

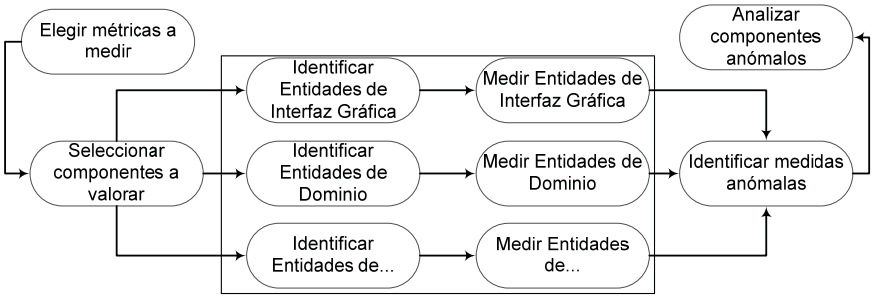

Figura 2 Proceso de medición propuesto

El objetivo del trabajo es corroborar de manera empírica la necesidad de múltiples intervalos de validación de métricas dependiendo de la naturaleza del problema que resuelve la entidad de código medida. Como resultado del caso de estudio se proponen unos intervalos de validación para un conjunto de métricas.

En lo que sigue el artículo se estructura de la manera siguiente. En la sección 2 se introducen las nociones básicas asociadas al proceso de experimentación en ingeniería del software. El grueso del trabajo se presenta en la sección 3 que describe el caso de estudio utilizando las etapas identificas previamente en el proceso de experimentación: definición, planificación, operación, análisis, validez. Por último, en la sección 4 se concluye y proponen líneas futuras de actuación.

2. Experimentación en ingeniería del software

Cuando se plantea un estudio empírico siempre hay un conjunto de elementos del mundo real del que se necesita conocer cierto comportamiento que se expresa a través de una hipótesis. Ésta se quiere aceptar o rechazar tomando como base los resultados observables y medibles. En [15] se presentan tres tipos de estrategias empíricas que pueden ayudar para realizar esta tarea: experimentos, casos de estudio y encuestas. Para este trabajo se enmarca dentro de la categoría de casos de estudio.

Los casos de estudio son estudios observacionales, esto es, investigan fenómenos en el contexto real cotidiano conforme se desarrollan. En la ingeniería del software se suelen utilizar para estudiar fenómenos a gran escala, como el desarrollo completo de procesos, monitorización de proyectos, o para evaluar métodos o herramientas. Su propósito está orientado normalmente a analizar un determinado atributo o establecer relaciones entre diferentes atributos.

Otro aspecto importante para realizar experimentación en ingeniería del software es la elección de un proceso que asista al desarrollo del caso de estudio. El proceso experimental que se ha seguido es el propuesto en [16], que consta de seis etapas principales (ver Figura 3), y sirven como plantilla base para presentar y estructurar el desarrollo del mismo.

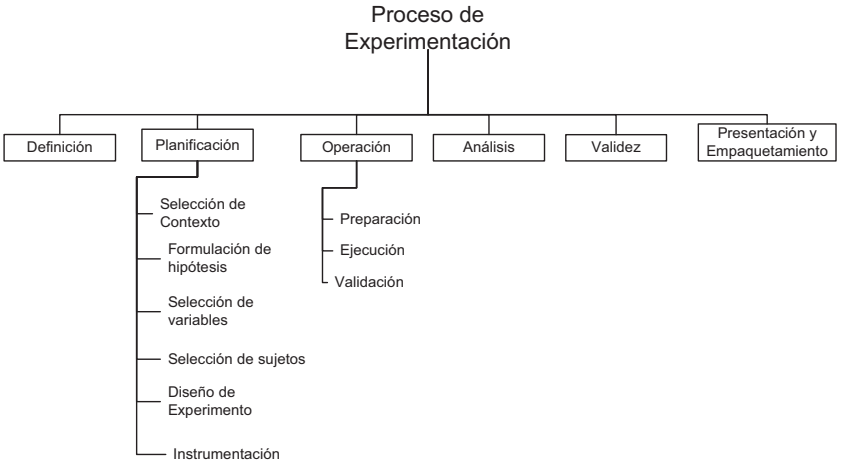

Figura 3 Visión general del proceso experimental

3. Caso de estudio

Este apartado presenta los resultados obtenidos al aplicar el de proceso experimental elegido [16]. Cada uno de los subapartados se corresponde con una etapa del proceso.

3.1 Definición

Analizar códigos fuentes mediante métricas

- Con el propósito de comprobar si sus valores recomendados varían dependiendo de la naturaleza del problema que resuelve la entidad medida. La naturaleza del problema se define a partir de algunos estereotipos de UML sobre clasificadores.
- Con respecto a los factores de calidad de código relacionados con de tamaño, documentación, estructura y complejidad.
- Desde el punto de vista de mejorar la identificación de entidades anómalas en el proceso de medición.
- En el contexto de una asignatura de proyecto fin de carrera de Ingeniería Informática

Se busca obtener evidencias empíricas que en la evaluación de sistemas heterogéneos con un solo intervalo de interpretación puede dar lugar a muchos resultados erróneos.

3.2 Planificación

Si la definición establece por qué se va a realizar el experimento, la planificación establece como se llevará a cabo. Esta etapa se divide en seis pasos que se corresponden con cada una de las siguientes secciones.

3.2.1 Selección de Contexto

En el 5º curso de Ingeniería Informática en la Universidad de Burgos (UBU) existe una asignatura de Sistemas Informáticos conocida comúnmente como proyecto final de carrera. En ella, los alumnos deben presentar un proyecto obtenido fruto de un proceso de desarrollo.

Como responsables de la evaluación de los mismos se establecieron unos criterios de evaluación donde se incluyó uno para valorar la calidad del código entregado. Para ello se utilizan métricas de código recogidas automáticamente a través de herramientas y su interpretación a través de intervalos generales y otros relativos a la Universidad de Burgos. Como criterio cuantificable se utiliza un indicador de cobertura de métricas que se obtiene contando el número de los valores de las métricas de un proyecto que están dentro de los intervalos recomendados y se divide entre número total de métricas consideradas. Este indicador sirve como criterio de conformidad equivalente al concepto *compliance* definido en los modelos de calidad de producto ISO 9126.

En la Figura 4 se muestra una evaluación de un conjunto de proyectos (filas de la tabla) utilizando dos indicadores de cobertura, uno relativo a la propia organización UBU y otro general basado en los intervalos recomendados por la herramienta. Además, se muestran los valores ordenados respecto al indicador general (columna *Cobertura Tool*). Un histórico de los datos de métricas de distintos trabajos puede obtenerse en http://pisuerga.inf.ubu.es/lsi/Asignaturas/SI/MetricSist.html#Coberturas.

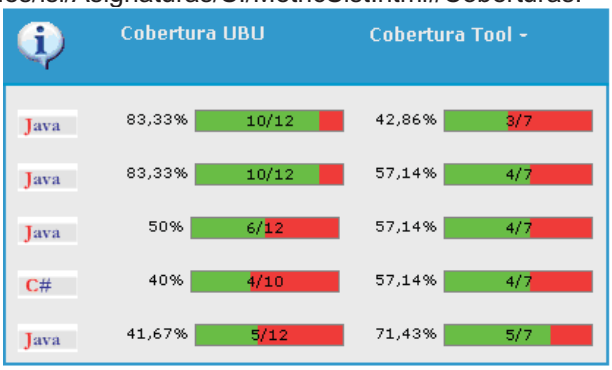

Figura 4 Evaluación de proyectos

En este proceso de evaluación, aparecieron algunos casos en los cuales se obtuvieron resultados contradictorios entre las coberturas de las métricas y los criterios subjetivos de los miembros del tribunal de evaluación.

La experiencia adquirida es que la interpretación de un valor de una métrica utilizando un sólo intervalo general o relativo a la propia organización no es suficientemente fiable y puede ser mejorada si se incorpora al proceso de medición información de la naturaleza del problema que resuelve el proyecto medido.

Hasta donde alcanza el conocimiento de los autores no existen clasificaciones de proyectos basadas en la naturaleza del problema. Pero cuando se realiza una rápida inspección de código tanto los desarrolladores como los evaluadores de una organización sí conocen la naturaleza del problema al que se enfrentan distintas entidades de código que componen el proyecto. Así, a través de las inspecciones realizadas basadas en la estructura código, se pueden distinguir las capas y subsistemas del proyecto: interfaz gráfico, modelo, controladores, pruebas. Además dentro de las capas y subsistema existen unas clases especiales llamadas excepción para el tratamiento de errores.

3.2.2 Formulación de hipótesis

H0: Todos los grupos diferentes de entidades orientadas a objeto respecto de la naturaleza del problema que resuelven (e1 excepciones, e2 interfaz gráfico, e3 modelo, e4 controladores, e5 pruebas) se comportan igual con respecto a las métricas de código.

El refinamiento de esta hipótesis se basa en la formulación de hipótesis estadísticas formuladas como hipótesis nula (H0) frente a su alternativa (H1), la cual esta implícitamente aceptada como la negación de H0 (H0 = ¬ H1).

Sobre cada una de las métricas de código consideradas (m) se refina la hipótesis basada en la media teórica (μ) de cada grupo de clases (ei)

H0: $\mu^m_{e1}=\mu^m_{e2}=\mu^m_{e3}=\mu^m_{e4}=\mu^m_{e5}$

La decisión de aceptar una u otra hipótesis en función de los resultados observados implica una serie de riesgos que medidos en términos de probabilidad son α y β. Si se elige H1 pero la real es H0 se comete un error que en términos de probabilidad se denomina α. El valor considerado para α ha sido de 0.01.

3.2.3 Selección de sujetos

El proceso de muestreo seguido ha sido aleatorio respecto a los proyectos presentados en la asignatura. Los datos se corresponden con los valores de las métricas de código de 30 proyectos presentados en la asignatura de Sistemas Informáticos de 5º de Ingeniería Informática de la Universidad de Burgos desde el curso 2003-04 (http://pisuerga.inf.ubu.es/lsi/Asignaturas/SI/HistoricoSist.html), 2 desarrollados por profesores de la misma Universidad y 2 de carácter industrial, Java Development Kit 1.5.0, JUnit 4.0. Estos cuatro últimos sirven para tener una referencia de comparación con el trabajo realizado por los alumnos y como criterio de validación externo del caso de estudio.

Tomando como referencia las líneas de código (LOC) en la Tabla 1 se recoge el tamaño de los códigos analizados dependiendo de la naturaleza del problema considerada.

Naturaleza del problema	LOC.
e_1 excepciones	8.711
e_2 interfaz gráfico	645.293
e_3 modelo	154.661
e_4 controladores	182.243
e_5 pruebas	93.730

Tabla 1 Tamaño de la muestra

3.2.4 Diseño del experimento

El diseño del experimento va a determinar cómo se estudian los datos obtenidos. Esto está muy relacionado con los tipos de análisis estadísticos que se pueden llevar a cabo. Por cada proyecto se deben identificar las entidades de cada categoría (e1 excepciones, e2 interfaz gráfico, e3 modelo, e4 controladores, e5 pruebas) y posteriormente calcular sus medias aritméticas. En la Tabla 2 se muestra un ejemplo con los resultados de la medición de una métrica de complejidad sobre un mismo proyecto y respecto a todas las categorías consideradas. La categoría “global” mostrada en la primera fila de la columna cuya cabecera es “Naturaleza” hace referencia al cálculo de la métrica sobre todo el proyecto sin hacer ningún tipo de clasificación. Se considera un diseño del caso de estudio intrasujeto, es decir todos los proyectos tienen el mismo tratamiento.

Nombre Proyecto	Naturaleza	Complej.
2005_refactorizaciones_xmi	global	2,12
2005_refactorizaciones_xmi	excepción	1
2005_refactorizaciones_xmi	interfaz	1,9
2005_refactorizaciones_xmi	modelo	1,95
2005_refactorizaciones_xmi	controlador	1,89
2005_refactorizaciones_xmi	prueba	2,59

Tabla 2 Medición de un proyecto

Como técnica de contraste de hipótesis estadística se ha elegido el Análisis de Varianza en sus variantes ANOVA de una vía o ANOVA no paramétrico de Kruskal-Wallis, que afecta simultáneamente a los valores medios o esperados (media aritmética, mediana) de k poblaciones (variables aleatorias). En el caso de de ANOVA de una vía, los valores de la muestra deben seguir una distribución normal y tener idénticas varianzas, en caso de no cumplirse se aplicará la versión de Kruskal-Wallis.

En este estudio interviene una variable aleatoria Y (métrica de código), denominada variable observable o variable respuesta, que se analiza en k situaciones experimentales (e1 excepciones, e2 interfaz gráfico, e3 modelo, e4 controladores, e5 pruebas), las cuales definen el llamado factor o vía k niveles de tratamiento (Naturaleza). Esta técnica deberá ser aplicada sobre cada una de las métricas de código consideradas. Si al aplicar el contraste de análisis de varianza el resultado de p-valor < 0,01 se rechaza la hipótesis de igualdad de un único intervalo de validación H0, aceptando la necesidad de múltiples intervalos dependiendo de naturaleza del problema que resuelven.

Para determinar la variante de análisis de varianza, una vía o Kruskal-Wallis, se evaluarán las precondiciones de ANOVA de una vía utilizando los siguientes contrastes:

- Respecto a la estructura de probabilidad normal con el contraste de Shapiro-Wilk.
- Respecto a la homocedastidad se empleará el test de homogenidad de varianzas de Barlett.

Si alguno de ellos falla, es decir se obtiene un resultado p-valor <= 0,05 se aplicará el test de Kruskal Wallis.

Si se aplica el contraste de ANOVA de una vía y se rechaza la hipótesis de igualdad nula de medias H0 se procederá a la realización de contrastes de medias dos a dos ($m_ie2 – m_ie1$, $m_ie3 – m_ie1$, $m_ie4 – m_ie1$,…, $m_ie5 – m_ie4$, donde mi es la métrica de código tratada y ej las situaciones experimentales que definen el factor guía naturaleza del problema). El resultado es un mapa de relaciones de las k situaciones consideradas. El test recomendado para llevar a cabo comparaciones múltiples es el de Tukey por proporcionar intervalos de confianza de menor longitud.
Por último, independientemente de la versión de análisis de varianza empleado y en caso de aceptar H1, se aplicarán los estadísticos percentil 25 y 75 sobre cada una de las métricas consideradas para cada uno de los tipos de problemas considerados (e1 excepciones, e2 interfaz gráfico, e3 modelo, e4 controladores, e5 pruebas). Estos valores se pueden utilizar para estimar los valores de los intervalos válidos de una métrica [4].
En el siguiente diagrama de actividad se muestra a modo guía los diferentes test aplicados y las decisiones basadas en p-valor (α) para llevar a cabo el caso de estudio. La Figura 5 muestra un diagrama de actividad con del diseño del experimento seguido.

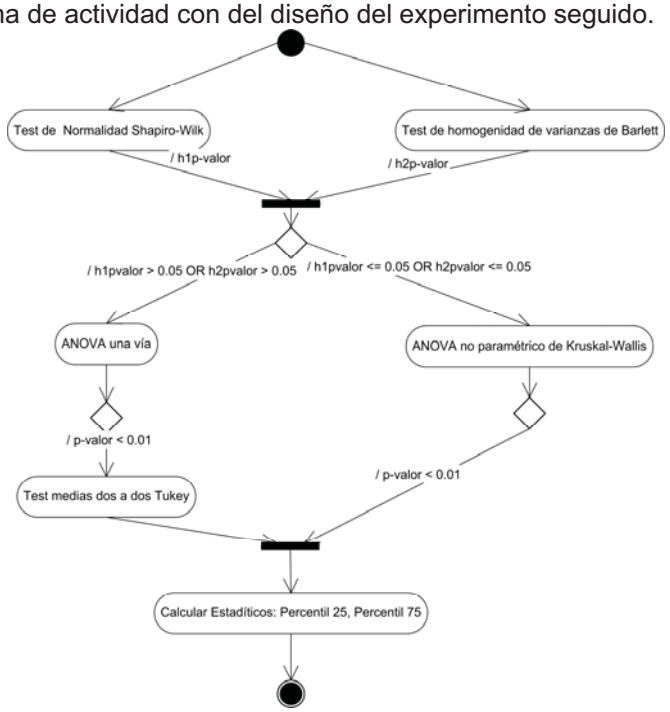

Figura 5 Diseño de del experimento

3.2.5 Instrumentación

Esta sección se ha desglosado en dos, una dedicada a herramientas para medir métricas de código y otra que muestra una relación entre métricas y propiedades de diseño.

3.2.5.1 Métricas de código y herramientas

La actividad de medición necesita de herramientas que automaticen el cálculo de valores de las métricas para una determinada entidad de código. Además los resultados deben ser exportables para poder ser analizados con otras herramientas de análisis estadísticos de datos. Para ayudar a la elección de herramientas en la Tabla 3 se muestra el resultado de la evaluación de un conjunto de herramientas respecto a las siguientes características:

C1.Lenguaje de programación sobre el que trabajan.
C2.Entrada: ficheros binarios o fuentes (binarios/fuentes/ambos).
C3.Número de métricas que calcula.
C4. Formato de exportación de resultados (html/txt/xml/xls).
C5. Indicadores gráficos o técnicas de agrupación y filtrado para analizar los resultados (Si/No)

Una de las críticas realizadas a la experimentación con métricas de código es la falta de precisión en su definición. Esto se pone de manifiesto en las herramientas en las diferentes reglas de cálculo aplicadas con la consecuente diferencia de valores obtenidos en el cálculo de la misma métrica sobre la misma entidad de código.

Herramientas	C1	C2	C3	C4	C5
Dependency Finder	java	binarios	33	html,txt,xml	No
RefactorIt	java	fuentes	25	html,txt xml	Si
JDepend	java	binarios	9	html,txt,xml	No
Eclipse Metrics - v1.3.6	java	fuentes	25	xml	No
NDepend	.NET	ambos	66	html, txt, xml, xls	Si
SourceMonitor	java, C#, C++, VB	fuentes	14	txt, xml,	Si

Tabla 3 Herramientas de medida de código

Aunque las definiciones de muchas de las métricas de código se mantienen independientes del lenguaje de programación, en la práctica muchas herramientas o componentes sólo trabajan sobre un único lenguaje de programación (ver columna C1 de la Tabla 3).

Descripción	Identificador	MinValor	MaxValor
Nombre proyecto	M0		
Líneas de código	J0		
Nº de sentencias	J1		
% de sentencia condicionales	J2		
Nº de llamadas a métodos	J3		
% Líneas de comentarios	J4	8	20
Nº de clases e interfaces	J5		
Nº de métodos por clase	J6	4	16
Media de sentencias por método	J7	6	12
Máxima complejidad	J10	2	8
Máxima profundidad de bloques	J12	3	7
Media de profundidad de bloques	J13	1	2,2
Media de Complejidad	J14	2	4

Tabla 4 Métricas de la herramienta SourceMonitor

En el contexto de medición que se plantea en este trabajo se debe permitir evaluar proyectos de diferentes lenguajes de programación y la exportación de resultados en formatos de texto

para el posterior análisis de los mismos. Bajo estas premisas la herramienta seleccionada en medición es SourceMonitor [17]. En la Tabla 4 se presentan las métricas de subsistema proporcionadas por la herramienta y los intervalos recomendados sobre algunas de ellas.

3.2.5.2 Caracterización de subsistemas mediante métricas

Independientemente de las convenciones particulares sobre las métricas de código, cada vez que se analiza o se habla del código de un sistema software, se quiere obtener una impresión del tamaño y complejidad del mismo. Algunos trabajos expresan el tamaño de un sistema en términos de líneas de código, número de clases e incluso cantidad de megabytes del código fuente.

Estos números no son nada más que valores de alguna métrica básica. Desafortunadamente después de obtener un conjunto de valores de manera aislada todavía se tiene problemas para caracterizar el sistema o entidad evaluada.

La caracterización a través de métricas tiene que servir para reflejar la bondad de los principales aspectos de diseño: tamaño, complejidad, acoplamiento, herencia, documentación, etc.

Para proporcionar una caracterización de este tipo a partir de las métricas definidas en la Tabla 4, este caso de estudio supone la relación entre características y métricas que se muestra en la Tabla 5.

DOCUMENTACIÓN	% de líneas de comentarios	J4
ESTRUCTURA	Nº sentencias por clase	J1/J5
	Nº de métodos por clase	J6
	Nº sentencias por método	J7
COMPLEJIDAD	% de sentencia condicionales	J2
	Media de profundidad de bloques	J13
	Media de complejidad	J14

Tabla 5 Caracterización de código mediante métricas

3.3 Operación

Por cada proyecto, el tiempo empleado para el cálculo de métricas e identificación de tipos de entidades ha estado comprendido entre 15 y 20 minutos.

Para identificar los diferentes tipos de las entidades de código, se ha utilizado como técnica la convención de nombres de las entidades. Además por cada tipo de entidad se han considerado dos criterios de identificación: identificación por agrupamiento (agrupación o individual) y el tipo de conocimiento necesario para la identificación (genérico o específico del proyecto).

La identificación por agrupación se basa en la identificación de capas del proyecto, se considera que todas las entidades que están en una misma capa son del mismo tipo. Por ejemplo, la identificación de entidades de tipo e2 interfaz gráfico de un proyecto, puede realizarse con la identificación de la capa de interfaz gráfica, cuya convención de nombres suele ser “interfaz”, “gui”, “form”, etc. El conocimiento necesario basado en convención de nombres para identificar entidades puede ser genérico respecto a convenciones de diseño o del lenguaje de programación, o bien por conocimiento específico de los requisitos del proyecto. Por ejemplo, si en un proyecto de cálculo de métricas aparece una capa llamada “métricas”, en una primera inspección pertenecerá a los tipos e3 modelo o e4 controladores. La Tabla 6 muestra un resumen de los criterios de identificación expuestos dependientes del tipo de entidad.

En la inspección realizada, la identificación de ciertas entidades no se ha podido etiquetar y no se han considerado en el caso de estudio. El porcentaje de etiquetado ha sido 76,6 %, siendo la unidad de porcentaje la línea de código.

Otra característica operacional del caso de estudio ha sido que no en todos los proyectos se han podido identificar entidades de los 5 tipos propuestos, el porcentaje de datos desconocido respecto al total ha sido del 27,9%.

	e_1	e_2	e_3	e_4	e_5
Agrupación	No	Si	Si	Si	Si
Conocimiento Genérico	Si	Si	Ambos	Ambos	Si
Convenciones de nombres	exception	interfaz gui forms ui report	core model	control fachada manager handler	test debug dummy

Tabla 6 Identificación de tipo de entidades

3.4 Análisis

A partir del caso de estudio llevado a cabo y aplicando los contrastes de hipótesis estadísticos especificados en la sección 3.2.4, se concluye de manera general en rechazar la hipótesis de partida H0. Por tanto, se acepta como cierta la hipótesis H1, que indica que los grupos diferentes de entidades orientadas a objeto respecto de la naturaleza del problema que resuelven (e1 excepciones, e2 interfaz gráfico, e3 modelo, e4 controladores, e5 pruebas) se comportan de manera distinta respecto a métricas de código. En la Tabla 7 se muestra un resumen de los contrastes de hipótesis llevados a cabo (filas) y de sus resultados al ser aplicados sobre los datos de medición de cada de una de las métricas consideradas (columnas). El valor de cada celda se corresponde con la probabilidad de error α.Por otro lado, excepto el conjunto de datos formado por los valores de la métrica de porcentaje de líneas de comentarios (J4), el resto de conjunto de datos no cumplen alguna de las dos precondiciones para aplicar la variante ANOVA una vía, normalidad y homecedasticidad.

Respecto al conjunto de datos correspondientes a la métrica porcentaje de líneas comentarios (J4), en la Figura 6 se han calculado los intervalos de confianza de Tukey comparando dos a dos los valores de la media de J4 respecto a tipos de entidades considerados. El análisis de la salida lleva a las siguientes conclusiones:

- El porcentaje de líneas de comentario es mayor en e1 excepciones que en el resto de tipos considerados.
- El porcentaje de líneas de comentario respecto a e2 interfaz gráfico es mayor que el de e5 pruebas, y es menor que e3 modelo y e4 controladores.
- El porcentaje de líneas de comentario respecto a e3 modelo es mayor que el de e4 controladores y e5 pruebas.
- El porcentaje de líneas de comentario respecto a e4 controladores es mayor que el de e5 pruebas.

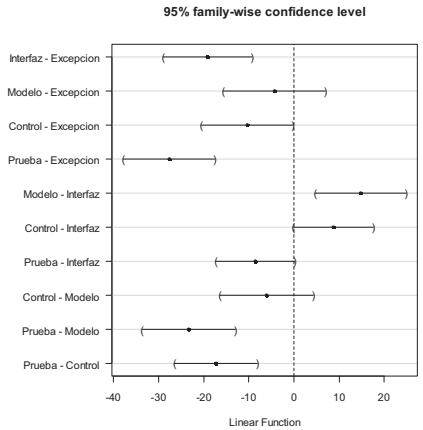

Figura 6 Intervalos de confianza de Tukey para métrica de porcentaje de documentación J4

Con estas evidencias empíricas este apartado concluye proponiendo unos intervalos válidos de métricas para cada uno de los tipos de entidades considerados (ver Tabla 8). La obtención de los extremos de los intervalos se adquiere calculando los estadísticos primer cuartil Q1 y tercer cuartil Q3 para cada tipo considerado.

Se ha utilizado R, como herramienta software para realizar los contrastes de hipótesis propuestos [18].

4. Conclusiones y líneas de trabajo futuras

En este trabajo se propone una variación en el proceso de medición incorporando conocimiento del inspector/evaluador sobre la clasificación de entidades del programa dependiendo de su naturaleza (e1 excepciones, e2 interfaz gráfico, e3 modelo, e4 controladores, e5 pruebas). A partir de dicha clasificación se han propuestos por cada una de sus categorías un intervalo de validación para un conjunto de métricas.
Se ha realizado un caso de estudio para corroborar la hipótesis de que la naturaleza del problema que resuelve la entidad de código medida influye en los valores de sus medidas. En este sentido, somos conscientes que la restricción multilenguaje marcada por el contexto específico de trabajo en el que se ha llevado a cabo el caso de estudio, ha determinado en exceso el conjunto de métricas seleccionadas y la subjetiva relación con características de diseño: documentación, estructura y complejidad. Es por ello que para mejorar la validez del experimento se podrían realizar réplicas aplicando otros conjuntos de métricas.
El caso de estudio aquí definido necesita de múltiples habilidades personales y técnicas en el uso de herramientas. Una de las líneas futuras de trabajo deseable en las herramientas de medida es la integración nuevas funcionalidades que reduzcan esta complejidad. Así se podría incorporar funcionalidad para clasificar las entidades según su naturaleza, mantenimiento de histórico para calcular intervalos de medidas, y poder realizar una evaluación multi intervalo en función de su naturaleza.

	DOC	ESTRUCTURA			COMPLEJIDAD		
	J4	J1 / J5	J6	J7	J2	J13	J14
Normalidad Shapiro Wilk	0,4281	2,69 e-09	1,152 e-08	0,0007163	2,96 e-06	0.7094	2,459e-06
Igualdad de varianzas Barlett	0,1348	0,000439	0,000433	1,162 e-05	1,40 e-11	0.000216	3,955e-06
ANOVA una vía	5,3 e-12	-	-	-	-	-	-
ANOVA Kruskal Wallis	-	0,000305	0,000101	4,947e-06	0,02264	4,934e-05	0,0001728

Tabla 7 Resultado de los contrastes estadísticos para cada una de las métricas

		DOC	ESTRUCTURA			COMPLEJIDAD		
		J4	J1 / J5	J6	J7	J2	J13	J14
e_1 excepciones	**Q1**	36.900	4.75000	1.750	1.0000	0.000	0.8450	1.0000
	Q3	56.900	25.06250	4.0000	5.785	29.500	2.2150	2.670
e_2 interfaz gráfico	**Q1**	23.450	44.16648	3.960	5.3250	8.875	1.8350	1.8750
	Q3	35.050	66.94631	7.4050	9.360	13.750	2.6450	2.815
e_3 modelo	**Q1**	41.900	33.11111	6.520	2.6750	7.050	1.4050	1.3950
	Q3	46.000	62.18954	9.2600	4.090	14.000	1.9100	2.020
e_4 controladores	**Q1**	32.050	36.19753	3.075	5.6450	10.300	1.8000	1.9700
	Q3	44.750	110.15374	11.1600	9.765	16.800	2.3550	2.935
e_5 pruebas	**Q1**	10.725	48.62500	5.545	4.5675	1.375	1.4775	1.0825
	Q3	26.875	96.30572	9.2775	9.130	8.575	1.8075	1.960

Tabla 8 Intervalos de valores recomendables de las métricas según la naturaleza de la entidad

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación TIN2008-05675 cuyo título de proyecto es Detección de oportunidades de refactorización para la corrección de defectos de diseño y la evolución de sistemas mediante la generación y ejecución de planes de refactorización.

Referencias

1. Pressman, R.S., Ingeniería del software : un enfoque práctico 6ª ed. 2005, McGraw-Hill Interamericana.
2. Sommerville, I., Ingeniería del software. 7ª ed. 2005, Pearson Educación.
3. IEEE, C.S., Guide to the Software Engineering Body of Knowledge: 2004 Edition - SWEBOK. 2005.
4. Fenton, N.E. and S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach. 2nd ed. 1998: Course Technology. 656.
5. ISO/IEC, Software engineering -- Product quality Part 1: Quality model. 2001.
6. Marín, B., N. Condori-Fernández, and O. Pastor, Calidad en modelos conceptuales. Un análisis multdimensional de modelos cuantitativos basados en ISO 9126., in Revista de Procesos y MÉtricas. 2007.
7. Chidamber, S.R. and C.F. Kemerer, A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 1994. 20: p. 476-493.
8. Lorenz, M. and J. Kidd, Object-oriented software metrics: a practical guide, ed. I. Prentice-Hall. 1994, Upper Saddle River, NJ, USA.
9. Martin, R., OO Design Quality Metrics. An Analysis of Dependencies. 1994.
10. Brito e Abreu, F. and R. Carapuça, Candidate metrics for object-oriented software within a taxonomy framework. Journal of Systems and Software, 1994. 26(1): p. 10.
11. McCabe, T., A Complexity Measure. IEEE Transactions on Software Engineering, 1976. 2: p. 308-320.
12. Piattini Velthuis, M.G., Calidad en el desarrollo y mantenimiento del software / Mario G. Piattini Velthuis, Félix O. García Rubio, ed. F.O. García Rubio. 2002, Paracuellos de Jarama (Madrid) :: Ra-Ma. 310 p. ; 24 cm.
13. Jacobson, I., G. Booch, and J. Rumbaugh, El proceso Unificado de Desarrollo del Software. 2000: Addison Wesley. 435 p. ;.
14. Arlow, J. and I. Neustadt, Uml 2 And The Unified Process: Practical Object-oriented Analysis And Design. 2005: Addison-Wesley Object Technology Series. 569.
15. Dolado, J.J. and L. Fernández, Medición para la Gestión en la Ingeniería del Software. 2000: Ra-Ma.
16. Wohlin, C., et al., Experimentation in Software Engineering: An Introduction. 2000: Kluwer Academic Publishers.
17. Campwood and Software, SourceMonitor. 2007.
18. OPEN-SOURCE, R. 1997 - 2003.