Un caso de estudio sobre la identificación de valores umbrales para medidas de código

Carlos López[1], Esperanza Manso[2], y Yania Crespo[2]

[1] Universidad de Burgos, EPS Edificio C, C/Francisco Vitoria s/n, Burgos, España, clopezno@ubu.es,
[2] Universidad de Valladolid, Campus Miguel Delibes, Valladolid, España, manso@infor.uva.es, yania@infor.uva.es

Resumen La identificación de medidas anómalas, basándose en valores umbrales de diferentes métricas, es uno de los recursos más utilizados para detectar posibles problemas en elementos software. En el caso particular del código, esta identificación puede afectar sobre todo al mantenimiento del mismo. En cuanto se refiere a valores umbrales de medidas de código, hemos constatado que para construir dichos valores umbrales, no se tiene en cuenta la naturaleza del problema que resuelve la entidad. En este contexto definimos la naturaleza del problema como aquella que se expresa con estereotipos estándar de clasificadores UML. Nuestra hipótesis de partida es que los valores umbrales de medidas están condicionados por la naturaleza del problema que resuelve dicha entidad de código. El trabajo presenta un caso de estudio con un doble objetivo, por un lado, definir una clasificación de entidades de código que se pueda automatizar, y por otro lado, estudiar la relación entre la clasificación y las medidas de las entidades. Los objetos para obtener la clasificación son proyectos de código abierto y los sujetos son un experto y dos estudiantes. Los resultados obtenidos han sido una clasificación de entidades y la constatación empírica de la existencia de relación entre la clasificación y las medidas.

Palabras clave: mantenimiento del software, métricas orientadas a objetos, medición de código, experimentación en ingeniería del software

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 para cuantificar y controlar la calidad del software [1–3]. Según Sommerville en [4], la medición es una actividad que forma parte de un proceso (ver Fig.1), que consiste en asociar valores numéricos a atributos de productos o procesos de software.

Pero en la literatura también existen muchas críticas sobre el uso que se hace de las métricas del software [5]. Una de ellas es que los valores umbrales utilizados para identificar medidas anómalas, obtenidas a través de experimentos

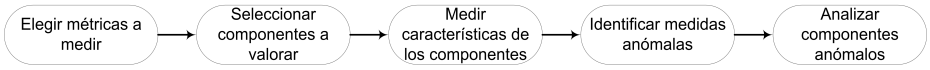

Figura 1: Proceso de medición definido por Sommerville

empíricos, están restringidos al contexto de medición, siendo limitada su utilización en otros contextos. Incluso valores umbrales recomendados extraídos de históricos de medidas de un mismo contexto, no son aplicables igualmente a todas las entidades de dicho contexto. Esto se debe a que las entidades pueden ser muy diferentes, de acuerdo a la naturaleza del problema que resuelven. Nos referiremos a éstas como entidades de diferentes categorías. En este sentido sería de interés estudiar la influencia que tiene la naturaleza del problema en las mediciones de las entidades de código, por ejemplo, como se modifican las características (media aritmética, varianza, etc) de las medidas cuando las entidades se clasifican por su naturaleza.

Ciertos estereotipos de UML pueden ayudarnos a clasificar las entidades de código según la "naturaleza del problema", y las medidas de las entidades dependerán de los objetivos iniciales del proceso de medición. Nosotros sospechamos que incorporar al proceso de medición la naturaleza de la entidad a medir, puede proporcionar resultados más precisos y útiles. Por ejemplo, las mediciones de una colección de entidades puede proporcionar un resultado "significativo", cuando no están clasificadas, y "significativo" en unas clases y "no significativos" en otras, cuando están clasificadas.

En el diseño de un sistema software orientado a objetos la información de los estereotipos de los de clasificadores muchas veces no está disponible explícitamente en los modelos o en el código, aunque los diseñadores y programadores la hayan tenido en cuenta implícitamente en sus soluciones. Por eso, es necesario definir alguna técnica que ayude a obtener esta información desde el código.

En lo que sigue el artículo se estructura de la manera siguiente. En la sección 2 se proporciona una motivación y una propuesta de mejora en el proceso de medición. En las secciones 3, 4, 5, 6 y 7 se expone el caso de estudio según el proceso experimental expuesto en [6] donde se define, se planifica, se ejecuta, se analiza y se valida, respectivamente. Por último, en la sección 8 se concluye y se proponen líneas futuras de actuación.

2. Motivación y propuesta de mejora

El objetivo del proceso de medición expuesto en la Figura 1 es la identificación *correcta* de componentes anómalos. Este trabajo propone una mejora basada en refinar la tarea de identificación de entidades anómalas del proceso de medición, concretamente en entidades de código. La propuesta será validada mediante un caso de estudio cuya realización ha seguido el proceso experimental propuesto en [6]: definición, planificación, operación, análisis, validez y presentación.

El código es un producto en constante evolución, y su evaluación mediante métricas no es nueva. De hecho, 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].
- Sobre métodos: McCabe [10].

Nuestra propuesta consiste en modificar el proceso de medición de la Figura 1 , incorporando una clasificación de la entidad a medir, previa a su medición, con el propósito de mejorar la evaluación de su calidad. Una vez conocida la categoría de la entidad, los valores umbrales se obtienen particularizados para cada categoría. La Figura 2 muestra nuestra propuesta del proceso de medición, al incorporar las nuevas tareas, enmarcadas con un rectángulo.

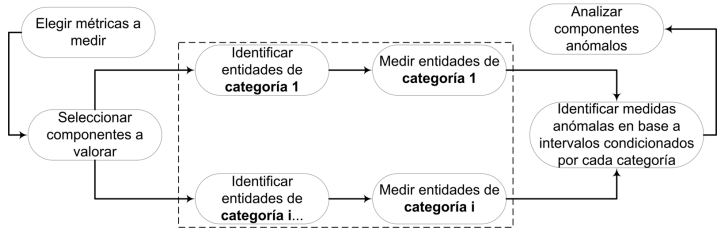

Figura 2: Proceso de medición propuesto

Bajo esta premisa y en un entorno de orientación a objetos, la selección de categorías se ha basado en la utilización de algunos estereotipos sobre clasificadores de UML. Por ejemplo, los estereotipos sobre clases de análisis [11, 12], entidad (*entity*), controladores (*control*) y límites (*boundary*). Además el criterio de clasificación límite se desglosa a su vez en: interfaz de usuario, interfaz 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, pruebas y utilidades. Obtener los estereotipos manualmente de los códigos fuentes es una tarea ingente y exigiría gran cantidad de recursos, por la cantidad de entidades de código en un sistema real. Por esto en [13] se presentó un plugin de Eclipse que asiste la clasificación automática de entidades de código a partir de un algoritmo basado en convención de nombres de las entidades.

En resumen, en esta propuesta la naturaleza de la entidad de código, y por tanto las diferentes categorías, se extraerá de los siguientes estereotipos UML: *e1 exception, e2 interface, e3 entity, e4 control, e5 test, e6 utility.*

3. Definición del caso de estudio: objetivos, hipótesis y variables

Nuestro estudio tiene dos objetivos:

Objetivo 1

- Analizar entidades de código fuente
- Con el propósito de obtener una clasificación
- Con respecto a la naturaleza de entidades basada en estereotipos UML
- Desde el punto de vista de los investigadores
- En el contexto de un conjunto de entorno de la Universidad de Burgos.

Objetivo 2

- Analizar entidades de código fuente
- Con el propósito de estudiar la relación de la naturaleza de dichas entidades
- Con respecto a las métricas de tamaño, documentación, acoplamiento, cohesión, herencia y complejidad
- Desde el punto de vista de los investigadores
- En el contexto de un conjunto de aplicaciones de código abierto, en concreto, plugins de Eclipse.

Los objetivos de este estudio dan lugar a la siguiente hipótesis:

Las métricas de entidades de código se comportan igual, independientemente de la naturaleza de la entidad

Las variables dependientes son las medidas de entidades de código de la Tabla 1 (M_i). La variable independiente es la clasificación de dichas entidades, basada en la siguiente escala nominal: *e1 exception, e2 interface, e3 entity, e4 control, e5 test, e6 utility.*

4. Planificación

Esta sección contiene información relativa a los sujetos y objetos del estudio y cuestiones relacionadas con la instrumentación para llevar a cabo el estudio.

4.1. Selección del contexto y sujetos

El caso de estudio se llevó a cabo como trabajo en una asignatura de 5º de Ingeniería Informática. En el trabajo participaron dos estudiantes para la recogida de mediciones y un tutor especializado en el área de mantenimiento del software.

En el contexto global de la asignatura, los tutores proponen varios trabajos distintos para grupos de dos estudiantes, uno de ellos es este caso de estudio. Posteriormente los estudiantes de la asignatura eligen uno o varios. La asignación final del trabajo al grupo único de alumnos sigue la siguiente regla: en caso de existir más de un grupo interesado en el mismo proyecto, el profesor elige el de mejor expediente académico.

4.2. Objetos del estudio

Para seleccionar los proyectos que son los objetos de este estudio, hemos considerado únicamente plugins de la herramienta Eclipse, obtenidos a través del repositorio de software de código abierto *SourceForge* (`http://sourceforge.net/`). La selección de proyectos se hizo siguiendo los criterios siguientes:

- Porcentaje de actividad, medido a través de la información de actividad de modificaciones continuadas y la actividad reciente. El criterio establecido ha sido porcentaje de actividad $\geqslant 85$
- Popularidad, medida a través del número de descargas por parte de los usuarios. El criterio establecido ha sido Nº de Descargas $\geqslant 7000$.
- Estado del desarrollo de la aplicación, medido utilizando la siguiente escala ordinal: 1 Planificación, 2 PreBeta, 3 Alpha, 4 Beta, 5 Producción/Estable, 6 Madurez, 7 Inactive. El criterio establecido ha sido estado $\geqslant 3$.
- Lenguaje de programación utilizado en su implementación, aplicando el criterio de que sea el mismo lenguaje para todos, Java en este caso.

4.3. Diseño del caso de estudio

Todos los sujetos implicados trabajaron con todos los objetos utilizados, con el fin de obtener la clasificación de los mismos, para ello utilizaron una convención de nombres (Tabla 3). De esta forma se cubre el primer objetivo del estudio. Los alumnos proponen cadenas de texto que ayudan a identificar entidades de cada clase y el tutor valida dichas cadenas con inspecciones sobre el código de entidades que contenían las cadenas en sus nombres. Este proceso de validación es iterativo hasta conseguir clasificar más del 50 % de las entidades con el conjunto de nombres proporcionado. Una vez obtenidas las cadenas de cada categoría considerada se procedió a automatizar el proceso de clasificación.

4.4. Instrumentación

Este experimento requiere dos tipos de herramientas, unas para calcular métricas de código y otras que asistan en el proceso de categorización de las entidades de código.

En el contexto de medición que se plantea en este trabajo se necesita una herramienta que calcule un amplio conjunto de métricas orientadas a objetos y permita la exportación de resultados para su análisis posterior. Bajo estas premisas, la herramienta seleccionada para obtener las medidas ha sido RefactorIt [14]. En la Tabla 1 se presentan las métricas proporcionadas por la herramienta y los valores umbrales que recomienda RefactorIt para algunas de ellas (columnas MinValor y MaxValor). Además en la columna llamada ámbito se presenta a qué tipo de entidad de código está asociada la métrica: paquete (P), clase (C), método (M) o todas las categorías anteriores juntas (T).

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

Descripción	Identificador	MinValor	MaxValor	Ámbito	Característica
Comment Lines of Code	CLOC			T	TAM
Cyclomatic Complexity	V(G)	1	10	M	COM
Density of Comments	DC	0.2	0.4	T	DOC
Executable Statements	EXEC	0	20	T	TAM
Non-Comment Lines of Code	NLOC			T	TAM
Number of Parameters	NP	0	4	M	TAM
Total Lines of Code	LOC	5	1000	T	TAM
Abstractness	A	0.0	0.5	P	ABS
Afferent Coupling	Ca	0	500	P	ACO
Depth in Tree	DIT	0	5	C	HER
Efferent Coupling	Ce	0	20	P	ACO
Instability	I	0.7	1.0	P	ACO
Number of Abstract Types	NOTa	0	20	P	ABS
Number of Children	NOC	0	10	C	HER
Number of Concrete Types	NOTc	0	80	P	ABS
Number of Exported Types	NOTe	3	50	P	ACO
Number of Fields	NOF	0	1	C	TAM
Number of Types	NOT	0	80	P	TAM
Response for Class	RFC	0	50	C	COM
Weighted Methods per Class	WMC	1	50	C	COM
Number of Attributes	NOA	0	5	C	TAM
Cyclic Dependencies	CYC	0	1	P	PDA
Dependency Inversion Principle	DIP	0.3	1.0	C	PDA
Direct Cyclic Dependencies	DCYC	0	1	P	PDA
Distance from the Main Sequence	D	0.0	0.1	P	PDA
Encapsulation Principle	EP	0	0.6	P	PDA
Lack of Cohesion of Methods	LCOM	0.0	0.2	C	COH
Limited Size Principle	LSP	0	10	P	PDA
Modularization Quality	MQ	0	1000	P	PDA
Number of Tramps	NT	0	1	M	O

Tabla 1: Conjunto de métricas definido en RefactorIt

quiere obtener información 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. Desde nuestra perspectiva, la caracterización de las entidades por medio de métricas es de interés en cuanto que éstas reflejen la bondad de ciertos aspectos del diseño como son: tamaño (TAM), documentación (DOC), acoplamiento (ACO), herencia (HER), complejidad estructural (COM), abstracción (ABS), cohesión (COH) o principios de diseño (PDA). Se ha incorporado una columna en la Tabla 1, llamada *característica*, que recoge esta clasificación subjetiva. En cualquier caso, estas

métricas son consideradas de interés por su relación con diferentes aspectos de la calidad de los productos software [15–18].

5. Operación

Los alumnos han sido formados en el uso de la herramienta RefactorIt por el mismo profesor que les guiará en el caso de estudio. Además seleccionaron los proyectos a medir bajo los criterios propuestos por el tutor. En cada fila de la Tabla 2 se presentan las características de los proyectos elegidos en este caso de estudio. Por cada uno se presenta información relativa a su tamaño, expresada como número de entidades de cada categoría, e información externa obtenida por el repositorio software de código abierto. En la columna *sin clasificar* se indica el número de entidades que no han podido ser clasificadas en ninguna de las categorías. La última fila y la última columna muestran el total de líneas de código, por cada tipo de entidad y cada proyecto, respectivamente.

	Número de entidades							Información Sourceforge			Tamaño
Plugins Eclipse	e_1	e_2	e_3	e_4	e_5	e_6	Sin clasificar	%Actividad	Nº Descargas	Estado	LOC
esFtp	1	90	38	68	8	86	52	94,19	45878	4	5156
AVR	12	376	125	306	362	789	310	98,68	1464780	5,6	55004
Jedit	25	1671	1657	1548	12	1719	681	99,62	5371954	5,6	156656
EclEmma	1	257	288	78	186	35	150	99.93	1488061	5	13294
AzSMRC	45	511	655	272	9	759	758	99,00	59327	4,5	67760
EclipseME	39	575	951	535	281	1215	446	97,63	731177	5	80012
ELBE	26	1788	1561	1216	138	380	288	85,95	30172	7	76188
OpenReports	17	0	568	1074	2	159	383	96,80	235708	4,5	27338
EclipseCorba	7	145	148	232	143	205	334	94,86	23062	3	25051
LabelDecorator	7	0	116	34	52	177	74	85,00	7004	5	5256
LOC Totales	**909**	**111818**	**85911**	**83876**	**14238**	**14797**	**200166**				

Tabla 2: Información de los objetos (proyectos) elegidos

Para obtener la clasificación de entidades, *objetivo 1* de este estudio, inicialmente se ha partido de un algoritmo de clasificación que se basa en criterios de convención de nombres de las entidades. Todas las entidades de código tratadas tienen un nombre *simple* y un nombre cualificado. El nombre cualificado es el nombre *simple* precedido con los nombres *simples* de todos los espacios de nombre donde esta contenido el elemento. En la Tabla 3 se recoge la convención de nombres utilizada, de manera que una entidad se clasifica en una de esas categorías, si contiene en su nombre cualificado alguna de las cadenas referenciadas en la tabla. En el caso de conflicto, porque el nombre contenga cadenas que correspondan a más de una categoría, prevalece el criterio del nombre simple de la

entidad. Aunque en la Tabla 3 se presentan los nombres en inglés se aplica una traducción de los mismos términos en distintos idiomas. En [13] se presentó un plugin de Eclipse que ejecuta el algoritmo utilizado en este estudio.

Categorías	e_1	e_2	e_3	e_4	e_5	e_6

Convención de nombres	`exception`	`interface`
		`gui`
		`forms`
		`ui`
		`report`
		`swing`
		`visual`
		`view`
		`awt`
			`core`
			`model`
			`entity`
				`control`
				`facade`
				`manager`
				`handler`
				`action`
				`callback`
				`maker`
				`provider`
					`test`
					`debug`
					`dummy`
						`util`
						`properties`
						`log`
						`preference`
						`template`
						`options`

Tabla 3: Convención de nombres en inglés para clasificar entidades

El tutor del experimento validó la Tabla 3, tal y como se describe en la Sec. 4.3 . Los alumnos realizaron después las tareas siguientes:

1. Realizaron la medición de las entidades con la herramienta RefactorIt
2. Exportaron los datos de las entidades y sus medidas a hojas de cálculo
3. Aplicaron las técnicas de agrupamiento y convención de nombres para obtener las clasificaciones de las entidades a partir de los datos de la 3

Algunas entidades no se pudieron clasificar utilizando la técnica de convención de nombres ya mencionada, por ello se excluyeron del estudio. El porcentaje de clasificación ha sido 60,8 %, siendo la unidad de porcentaje la línea de código.

La gran cantidad de entidades medidas y clasificadas en la recolección de datos de este caso de estudio (27.256) impide incluirlas en este documento. El conjunto de datos completo generado para su posterior análisis puede ser obtenido en `http://pisuerga.inf.ubu.es/clopez/JISBD2011/`.

6. Análisis de resultados

Algunas de las entidades seleccionadas (39,2 %) no se pudieron clasificar por diferentes razones: porque la entidad pertenecía a varias categorías o porque la técnica utilizada para clasificar no es exhaustiva, y hay entidades que no se pueden clasificar. El porcentaje de entidades clasificadas ha sido del 60,8 %, tomando como unidad la línea de código, y son estas entidades las consideradas en el estudio que sigue.

La hipótesis que vamos a estudiar es la siguiente:

H_0: La métrica M_i se comporta igual en las entidades de código, independientemente de su naturaleza.
H_1: La métrica M_i se comporta de manera diferente, dependiendo de la naturaleza de las entidades de código.

Donde M_i es una de las métricas de la Tabla 1, y la naturaleza de la entidad tiene 6 categorías distintas (*e1 exception, e2 interface, e3 entity, e4 control, e5 test, e6 utility*), que definen los niveles del tratamiento. Un test de ANOVA (ANalysis Of VAriance) es el adecuado para contrastar este tipo de hipótesis, siempre y cuando se cumplan ciertas condiciones [19]: Normalidad de M_i y homocedasticidad. Si alguna de las condiciones no se cumple, usaremos el test similar, no paramétrico, de Kruskal-Wallis [20], en adelante K-W. El contraste de Shapiro-Wilk estudia la normalidad de las observaciones de la variable dependiente, la métrica M_i. Con respecto a la homocedastidad emplearemos el test de homogeneidad de varianzas de Barlett si la muestra contiene menos de 50 elementos o el test de Kolmogorov-Smirnov si la muestra es mayor.

Si el resultado del test, paramétrico o no, es significativo se rechazará la hipótesis nula, aceptando que la naturaleza de la entidad condiciona los valores de la métrica, y por ello sus valores umbrales. En este caso tendrá sentido proporcionar valores umbrales de la métrica para cada categoría considerada, valores umbrales que estimaremos con los percentiles 25 y 75 [1] para cada uno de los estereotipos considerados (*e1 exception, e2 interface, e3 entity, e4 control, e5 test, e6 utility*). Estos umbrales son de propósito general, y se podían haber obtenido a partir de límites de confianza, usando la desigualdad de Tchebychev o la distribución Normal, dependiendo del tipo de distribución de M_i. Los contrastes de hipótesis mencionados se han realizado utilizando el software R [21].

6.1. Resultados observados

Puesto que los contrastes de normalidad dieron significativos, no podemos aceptar que la distribución sea Normal. Es por ello que utilizaremos el contraste de K-W. Para cada una de las métricas de código (M_i) consideradas se enuncia la hipótesis siguiente:

$$H_0 : \mu_{e1}^{M_i} = \mu_{e2}^{M_i} = \mu_{e3}^{M_i} = \mu_{e4}^{M_i} = \mu_{e5}^{M_i} = \mu_{e6}^{M_i}$$

En la Tabla 4 se presenta un resumen de los resultados obtenidos al aplicar el test no paramétrico K-W para cada una de las métricas presentadas, cuando las entidades de código están clasificadas por estereotipos.

Para facilitar el análisis, la Tabla 4 representa dos criterios de clasificación respecto a las métricas: por un lado el ámbito de aplicación de la entidad que mide (columna Ámbito) y por otro lado, la relación subjetiva con ciertas características del sistema relacionadas con su calidad. Las filas de la tabla están ordenadas por el segundo criterio. Las tres últimas columnas recogen las tres categorías del criterio de ámbito. Respecto al significado del contenido literal de las celdas - - - que aparece en la Tabla 4 , indica que la métrica no es aplicable en ese ámbito.

En la Tabla 4 se muestran los p-valores del test de K-W. Puede verse que 34 de los 39 resultados han sido significativos al nivel 0,05. Por ello en la mayoría de las métricas estudiadas podemos aceptar que los valores umbrales debemos obtenerlos teniendo en cuenta la naturaleza de la entidad medida.

Si se realiza este análisis utilizando el criterio de ámbito, se observa que en el ámbito de paquete 3 métricas son no significativas (cumplen H_0) frente a 16 significativas (cumplen H_1), en el ámbito de clase y método la hipótesis H_1 se afianza, 18 cumplen H_1 frente a 2 H_0. Desde el criterio de característica se afianza H_1 en las métricas de complejidad (cero no significativos y tres significativos) y acoplamiento (uno no significativo y tres significativo), respecto a la abstracción, tamaño y herencia la aceptación de H_1 es más moderada. Respecto a la categoría de principios de diseño la aceptación de H_1 es más débil (tres no significativos y cuatro significativos).

Identificador	Característica	Resultados *test K-W*		
		Paquete	Clase	Método
NOT	TAM	0.02238	---	---
NP	TAM	---	---	< 2.2e-16
NOA	TAM	---	< 2.2e-16	---
NLOC	TAM	0.01340	< 2.2e-16	< 2.2e-16
EXEC	TAM	0.005242	< 2.2e-16	< 2.2e-16
CLOC	TAM	0.02767	< 2.2e-16	0.0129
MQ	PDA	**0.1427**	---	---
LSP	PDA	**0.0775**	---	---
EP	PDA	0.006352	---	---
D	PDA	0.0004983	---	---
DCYC	PDA	0.03573	---	---
DIP	PDA	---	**0.2769**	---
CYC	PDA	0.0002174	---	---
NT	O	---	---	< 2.2e-16
NOC	HER	---	**0.2813**	---
DIT	HER	---	< 2.2e-16	---
DC	DOC	0.01089	< 2.2e-16	0.0367
WMC	COM	---	< 2.2e-16	---
RFC	COM	---	2.68E-09	---
V(G)	COM	---	---	< 2.2e-16
LCOM	COH	---	< 2.2e-16	---
NOTe	ACO	**0.1909**	---	---
I	ACO	0.0001181	---	---
Ce	ACO	0.007674	---	---
Ca	ACO	0,003002	---	---
NOTc	ABS	0.001988	---	---
NOTa	ABS	0.03124	---	---
A	ABS	0.0005088	---	---

Tabla 4: p-valores observados en el test K-W, subrayados los no significativos al nivel 0.05

6.2. Valores umbrales basados en percentiles

Como resultado final del caso de estudio las Tablas 5, 6 y 7 recogen los valores umbrales para aquellas métricas cuyos resultados son favorables a la hipótesis H_1.

Los valores umbrales propuestos se han obtenido seleccionando los estadísticos percentil 25 (Q1) y percentil 75 (Q3) de cada métrica [1]. Basándonos en estos valores umbrales propuestos, se expone un escenario posible para utilizar esta información con el fin de identificar medidas anómalas. En la Tabla 6 se ha resaltado la columna de la métrica de complejidad estructural *WMC (Weighted Methods per Class*) [7], donde se indican los valores umbrales para cada uno de los estereotipos considerados *e1 [1.75 - 4], e2[4 - 16], e3 [5 - 25], e4 [3 - 16.25], e5 [2 - 8.25], e6 [4 - 22]*, mientras que en la Tabla 1 aparece *[1, 50]* como valores umbrales recomendados por la herramienta RefactorIt, para esa métrica. A partir de esta información podemos destacar dos cosas: por un lado, el intervalo propuesto por la herramienta es menos preciso que los proporcionados por nosotros, de hecho los incluye. Por otro lado, la precisión (amplitud) de los valores umbrales depende de las categorías consideradas; de hecho en las clases excepción (e1) y test (e5) la precisión es mejor que en las restantes. Análogamente podemos realizar este análisis sobre el resto de mediciones de las tablas.

		EXEC	Ca	Ce	I	A	D	NOTc	CYC	EP
Exception	**Q1**	---	---	---	---	---	---	---	---	---
e_1	**Q3**	---	---	---	---	---	---	---	---	---
Interfaz	**Q1**	15.00	0.0	5.00	0.60	0.00	0.00	4	0	0.071
e_2	**Q3**	134.00	7.0	18.00	1.000	0.07	0.36	18	3.0	0.80
Entity	**Q1**	11.75	2.0	3.00	0.25	0.00	0.10	2	0	0.52
e_3	**Q3**	194.50	35.5	14.00	0.68	0.26	0.50	12	4.5	1
Control	**Q1**	7.50	0.0	2.00	0.31	0.00	0.00	1	0	0.00
e_4	**Q3**	93.00	8.5	11.00	1.00	0.29	0.25	11	1.5	1
Test	**Q1**	2.00	0.0	2.00	0.93	0.000	0.00	1	0	0.00
e_5	**Q3**	28.00	1.0	7.75	1.0	0.39	0.34	7	0	0.19
Utility	**Q1**	12.75	0.0	2.00	0.45	0.00	0.00	2	0	0.000
e_6	**Q3**	127.25	6.0	14.00	1.00	0.17	0.40	13	1.0	0.81

Tabla 5: Métricas de paquetes: Valores umbrales recomendados según la naturaleza del problema

7. Validación de los resultados

Los resultados observados son aplicables a una población de entidades similares a las seleccionadas en el estudio. Los resultados de este estudio se confirmaron con una réplica [22], que se llevó a cabo con un conjunto de métricas de código distintas, en el cual, la naturaleza de las entidades de código aparece relacionada con las métricas de dichas entidades. En esta réplica la identificación de entidades estaba más controlada, puesto que el inspector había formado a los desarrolladores de las aplicaciones que formaban parte del caso de estudio.

La validez de los resultados esta amenazada por: la subjetividad de la clasificación, que es difícilmente evitable, la no supervisión de los sujetos y la elimi-

		DC	LOC	NCLOC	EXEC	CLOC	WMC	DIT	RFC	LCOM	NOA
Exception	**Q1**	0.00	10.25	6.25	0	0	**1.75**	3	1.75	0	0
e_1	**Q3**	0.14	29.25	15.00	1.75	3	**4.00**	4	3.25	0.00	2
Interfaz	**Q1**	0.00	35.00	24.00	2	0	**4.00**	1	2.00	0	1
e_2	**Q3**	0.20	179.25	133.25	14.00	25	**16.00**	2	19.00	0.95	7
Entity	**Q1**	0.00	40.00	24.75	2	0.75	**5.00**	1	2.00	0	1
e_3	**Q3**	0.27	169.50	119.25	17.00	32	**25.00**	2	21.00	0.96	6
Control	**Q1**	0.00	18.00	14.00	1	0	**3.00**	1	2.00	0	0
e_4	**Q3**	0.21	115.25	79.00	12.00	13	**16.25**	2	17.00	0.75	3
Test	**Q1**	0.00	19.00	12.25	0	0	**2.00**	1	2.00	0	0
e_5	**Q3**	0.28	105.00	71.25	8.00	19	**8.25**	2	9.00	0.81	2
Utility	**Q1**	0.02	33.00	22.00	2	1	**4.00**	1	3.00	0	1
e_6	**Q3**	0.31	202.75	130.75	23.00	41	**22.00**	2	23.00	0.92	6

Tabla 6: Métricas de clases: Valores umbrales recomendados según la naturaleza del problema

		LOC	NCLOC	EXEC	NP	V(G)	NT
Exception	**Q1**	1	1	0	0	1	0
e_1	**Q3**	8	7.75	1	2	2	0.75
Interfaz	**Q1**	1	1	0	0	1	0
e_2	**Q3**	14	12	2	1	2	0
Entity	**Q1**	1	1	0	0	1	0
e_3	**Q3**	9	8	2	1	2	0
Control	**Q1**	1	1	0	0	1	0
e_4	**Q3**	11	9	2	1	3	0
Test	**Q1**	1	1	0	0	1	0
e_5	**Q3**	10	8	1	1	1	0
Utility	**Q1**	1	1	0	0	1	0
e_6	**Q3**	14	11	3	1	3	0

Tabla 7: Métricas de métodos: Valores umbrales recomendados según la naturaleza del problema

nación de una parte de los objetos, ya que no se pudieron clasificar. La última de estas amenazas puede solventarse en el futuro perfeccionando la clasificación, y la supervisión del estudio puede resolver la otra amenaza considerada.

8. Conclusiones y Líneas de Trabajo Futuro

Este trabajo tiene dos objetivos:

1. Proponer un nuevo proceso de medición, que incorpore el conocimiento del inspector/evaluador para clasificar las entidades a medir según su naturaleza.
2. Estudiar la relación entre una colección de métricas de entidades y la clasificación obtenida anteriormente.

De estos objetivos se derivan las hipótesis del estudio: Las métricas de entidades de código se comportan igual, independientemente de la naturaleza de la

entidad. La definición de la clasificación de entidades de código según su naturaleza en seis categorías: e1 exception ... cubre el primer objetivo.

Para estudiar el segundo objetivo hemos hecho una propuesta de cálculo de valores umbrales, basados en percentiles, para cada métrica, condicionados por la naturaleza de las entidades. Dichos umbrales son de propósito general, pero ya ponen de relieve la diferencia de resultados obtenidos cuando se considera la naturaleza de las entidades con respecto a los obtenidos sin considerar la naturaleza de las entidades. En conclusión, los resultados observados son favorables a considerar la naturaleza de las entidades cuando se trabaja con valores umbrales de métricas para detectar medidas anómalas.

La utilización de los resultados de este caso de estudio en una empresa, que desarrolle aplicaciones diversas de plugins de Eclipse, requiere la creación de un histórico de medidas, clasificadas según la naturaleza de las entidades. A partir del histórico se podrán construir, bien valores umbrales genéricos con los percentiles, o valores umbrales específicos dependiendo de los objetivos a conseguir. El equipo de desarrollo sería el más adecuado para clasificar las entidades, generando la tabla de convención de nombres. Por ello el proceso de este caso de estudio se puede reutilizar en otros contextos, donde las decisiones se apoyen en valores umbrales de métricas.

Aunque ya se ha realizado una réplica, creemos que se necesitan más. Las nuevas réplicas se pueden definir con otros conjuntos de proyectos seleccionados a partir de factores externos similares: porcentaje de actividad, popularidad, estado de desarrollo y área funcional. Con ello se pretende refinar y comparar los valores umbrales propuestos y probar si son dependientes de estos factores [23].

Se necesita validar mediante experimentación el proceso de medición. Por esto queremos estudiar como mejora la detección de entidades anómalas este nuevo proceso de medición, con respecto al proceso que no utiliza la clasificación de las entidades.

Por último, queremos validar nuestra propuesta incorporando nuevas métricas de las entidades.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el *Ministerio de Ciencia e Innovación*, en el proyecto TIN2008-05675.

Agradecimiento a los alumnos egresados Hector Alonso e Iñaki de Eguía por el gran empeño y entusiasmo que pusieron en realizar el caso de estudio, para obtener la clasificación de los entidades.

Referencias

1. Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* Course Technology, 2nd edition, 1998.
2. Computer Society IEEE. *Guide to the Software Engineering Body of Knowledge: 2004 Edition - SWEBOK.* 2005.

3. Roger S. Pressman. *Ingeniería del software : un enfoque práctico*. McGraw-Hill, 6ª edition, 2005.
4. Ian Sommerville. *Ingeniería del software*. Pearson Educación, 7ª edition, 2005.
5. Radu Marinescu. *Measurement and Quality in Object-Oriented Design*. PhD thesis, University of Timisoara, October 2002.
6. Per Runeson and Martin Host. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14:131–164, 2009.
7. Shyam R. Chidamber and Chris F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20:476–493, 1994. Journal.
8. Mark Lorenz and Jeff Kidd. *Object-oriented software metrics: a practical guide*. 1994.
9. Robert Martin. Oo design quality metrics. an analysis of dependencies. 1994. Journal.
10. Tomas McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2:308–320, 1976. Journal.
11. J. Arlow and I. Neustadt. *Uml 2 And The Unified Process: Practical Object-oriented Analysis And Design*. Addison-Wesley Object Technology Series, 2005.
12. Ivar Jacobson, Grady Booch, and James Rumbaugh. *El Proceso Unificado de Desarrollo del Software*. Addison Wesley, 2000.
13. Carlos López, Esperanza Manso, and Yania Crespo. The identification of anomalous code measures with conditioned interval metrics. In *13th TOOLS Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2010)*, Málaga (Spain), 2010.
14. Aqris-Software. Refactorit http://www.aqris.com/display/a/refactorit, 2001.
15. Lionel C. Briand, Jürgen Wüst, and Hakim Lounis. Replicated case studies for investigating quality factors in object-oriented designs. *Empirical Software Engineering*, 6(1):11–58, 2001.
16. R. G. Dromey. Cornering the chimera. IEEE Software:33–43, 1996.
17. Daniel L. Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55(3):243–276, 2005.
18. Esperanza Manso. *Estudio empírico para la validación de indicadores de la reusabilidad de diagramas de clases UML*. PhD thesis, Valladolid, 2009.
19. B.J. Winer, D.R. Brown, and K.M. Michels. *Statistical principles in experimental design*. Mc Graw Hill, 3ª ed edition, 1991.
20. S. Siegel and N.J. Castellan. *Non-parametric statistics for the behavioural sciences*. Mc Graw Hill, 2nd edition, 1988.
21. Open-Source. R, 1997- 2009.
22. Carlos López, Yania Crespo, Esperanza Manso, and Raúl Marticorena. Evaluación de código mediante múltiples intervalos de métricas. *Revista de Procesos y MÉtricas*, 6(1):19–30, Julio 2009 2009.
23. Yania Crespo, Carlos López, Raúl Marticorena, and Esperanza Manso. Language independent metrics support towards refactoring inference. In *9th ECOOP Workshop on QAOOSE 05 (Quantitative Approaches in Object-Oriented Software Engineering). Glasgow, UK. ISBN: 2-89522-065-4*, pages 18–29, July 2005.