Umbrales relativos. Caso de estudio para incorporar métricas en la detección de *Bad Smells*

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

Abstract: *To detect flaws, bad smells, etc, we often use quantitative methods: metrics or measures. It is common in practice to use thresholds setting the correctness of the measures. Most of the current tools use absolute values. Nevertheless, there is a certain concern about threshold applications on obtained values. Current work tries to accomplish case studies about thresholds on several products and different versions. By other side, product domain and size could also affect the results. We tackle if it is correct to use absolute vs. relative thresholds, seeing that effects could have in metric collection and bad smell detection.*

Resumen: *Para la detección de defectos, errores, etc. se utilizan en muchas ocasiones métodos cuantitativos: métricas. El empleo de umbrales (thresholds) para determinar la corrección de los valores obtenidos está ampliamente difundido. Sin embargo, existe una cierta discusión sobre la aplicación de dichos umbrales a los valores obtenidos. Este trabajo pretende realizar varios casos de estudio sobre la posible aplicación de umbrales sobre distintos productos. El dominio y tamaño del producto medido pueden afectar sobre las consideraciones a tomar. Este trabajo trata de establecer si es correcto utilizar umbrales absolutos frente a relativos, así como el efecto que puede tener sobre soluciones de recuperación de métricas y detección* de *bad smells sobre el código.*

Palabras Clave: *Métricas de Código, Umbrales, Proceso de Medición, Herramientas de Medición, Bad Smells,*

1. Contexto inicial

En anteriores trabajos [3,11], se ha planteado el uso de *frameworks* con el fin de dar un soporte completo al proceso de *refactoring* [4]: extracción de información del códigos fuente, recolección de métricas para la detección de *bad smells*, soporte de refactorizaciones y recuperación del código transformado.

Sin embargo en dicho proceso quedan puntos por cubrir. En particular, la relación entre métricas y *bad smell* definida en base a umbrales absolutos, en anteriores soluciones [3]. Aunque dichos umbrales podían ser personalizados, ésta era una labor para el gestor del producto. Quedan varias cuestiones que se plantearon a la hora de utilizar dicho enfoque sobre el código fuente: ¿qué valores utilizar?, ¿son adecuados dichos valores?, ¿son correctos para el producto concreto que utilizamos?

Este trabajo pretende contestar estas preguntas, utilizando un caso de estudio con varios productos de tamaño medio y centrándose también en la evolución de alguno de ellos. Las conclusiones obtenidas deberían dejar claro la necesidad de utilizar valores relativos.

El resto del artículo está organizado de la siguiente forma, en la sección 2 se muestra un conjunto de trabajos relacionados, la sección 3 desarrolla el caso de estudio centrándose en varios productos y sus versiones. En la sección 4 se establece la aplicación de los resultados para la detección de *bad smells*. Por último, en la sección 5 se muestran unas conclusiones de la solución propuesta.

2. Trabajos relacionados

En la mayoría de entornos que incluyen la recolección de métricas se empieza a incluir la posibilidad de fijar límites (umbrales) a los valores recogidos por las métricas. Generalmente, es el propio programador, normalmente bajo las guías de desarrollo de su empresa, el que establece estos valores. Pero estos filtros se establecen para todos los productos y los resultados obtenidos no sugieren siempre defectos catalogados.

Existen trabajos en la línea de detección de defectos del diseño. En los trabajos de Marinescu [9,10] se propone lo que denomina estrategias de diseño (*design strategies*). Dichas

estrategias se definen en base a métricas y aplicadas sobre la información de un metamodelo. El metamodelo recoge información del código, pero está pensado básicamente para consultas (todas las operaciones se traducen a sentencias SQL sobre las tablas) y no para completar el proceso de refactorización del código. En dicho trabajo, se plantea la utilización de filtros absolutos y relativos, pero no se menciona ningún estudio para tomar en cuenta la idoneidad de unos u otros.

En los trabajos de Mäntyla [7,8] se catalogan los defectos denominados *bad smell*, intentando ligarlos a un juego de métricas. Esta última asignación se sugiere, pero no se profundiza en exceso sobre la problemática de los umbrales, seleccionando en su mayoría umbrales absolutos. Mäntyla también muestra a través de validaciones con expertos el problema que detectar bad smell está sujeto a decisiones de la intuición humana.

Por otro lado, en [12], Tourwé y Mens proponen la detección de oportunidades de refactorización utilizando consultas en un entorno de meta-programación lógica. Ellos definen consultas para llevar acciones de corrección en el sistema. Como continuación de este trabajo, en [1], Muñoz usa un conjunto de preguntas lógicas basadas en métricas orientadas a objetos para detectar estos *bad smells* con umbrales absolutos.

Los umbrales también se han tratado por French en [5]. En el trabajo se propone un sistema basado en métodos estadísticos para determinar umbrales de diferentes productos, éstos no se aplican ni en la detección de *bad smells* ni para comprobar los efectos sobre varias versiones del mismo producto.

Partiendo de estos trabajos previos se pretende resolver ciertas cuestiones:

- Comprobar la corrección de la utilización de umbrales absolutos frente a relativos para una futura detección de defectos código o *bad smells*.
- Influencia del tipo y tamaño del producto software: frameworks vs. bibliotecas.
- Adecuación de la solución en diferentes versiones del mismo producto.

3. Caso de estudio

Partiendo de un conjunto de productos se inicia la recolección y estudio de sus métricas obtenidas para establecer qué hipótesis son correctas respecto al uso de umbrales.

3.1 Primera fase: Comparación entre productos

Se ha realizado un estudio comparativo entre seis productos. Se han tomado diferentes productos, en su mayoría con versiones estables utilizadas durante un periodo mayor de un año, y de tamaños medios a grandes, obviando el uso de los denominados "toys" o casos de pequeño tamaño, casi sin funcionalidad.

Los productos elegidos son:

- jfreechart-1.0.0.pre2 (629 clases)
- jhotdraw-6.0b1 (496 clases)
- struts-1.2.8 (273 clases)
- jcoverage-1.0.5 (90 clases)
- easymock-1.0.5 (47 clases)
- junit-3.8.1 (46 clases)

Todos estos productos están escritos en un lenguaje orientado a objetos como Java, puesto que los resultados extraídos se quieren aplicar sobre trabajos previos en dicho paradigma. En el estudio, se ha utilizado Eclipse y su plugin Metrics-1.3.6 como herramienta de recolección de métricas. Esto condiciona a la utilización de productos escritos en Java, aunque a juicio de los autores, se cree que el método es generalizable a otros lenguajes orientados a objetos.
Las métricas seleccionadas han sido tomadas sólo sobre clases, seleccionando métricas que muestren valores de: tamaño, complejidad, cohesión, herencia y especialización [2,6].

- NOF número de atributos
- NOM número de métodos
- WMC complejidad ciclomática
- LCOM ausencia de cohesión
- DIT profundidad en el árbol de herencia
- NSC número de hijos
- SIX coeficiente de especialización
- NORM número de métodos redefinidos

Para cada uno de estas métricas se han recogido los valores y calculado: media, media acotada (eliminando el 15% de valores extremos), desviación estándar, cuartil Q1, mediana, cuartil Q3, mínimo y máximo.

3.2 Conclusiones parciales

De los resultados obtenidos, ver Fig.1, se pueden deducir las siguientes conclusiones:

- Las distribuciones no son simétricas, las diferencias obtenidas entre las medias y medianas, así como la proximidad de la mediana al cuartil Q3 lo corroboran. En la mayoría de casos nos encontramos con distribuciones con asimetría positiva (cola de la distribución a la derecha).
- Las diferencias entre valores mínimos y máximos es muy grande, y además en cada producto es muy distinto. Esto sugiere datos muy dispersos, y con intervalos muy diferentes entre los productos.
- El tamaño del producto (número de clases) está ligeramente correlacionado con ciertas métricas lo que sugiere que el tamaño podría influir en los umbrales. Esto es más acusado en métricas de tamaño como NOF, NOM y WMC, con alta correlación entre ellas. Por el contrario métricas como LCOM, DIT sufren pocas variaciones entre los distintos productos.

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean JFreeChart 1.0.0-pre2	2,40	10,08	22,98	0,21	2,55	0,36	0,16	0,69
Bounded mean (15%)	1,41	7,45	15,87	0,17	2,47	0,04	0,08	0,46
Q3	3,00	11,00	25,00	0,50	3,00	0,00	0,14	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	3,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	5,05	15,01	38,82	0,32	1,14	1,48	0,37	1,23
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	48,00	166,00	490,00	1,00	7,00	16,00	3,20	9,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit-3.8,1	2,17	8,13	15,70	0,21	2,70	0,28	0,18	0,35
Bounded mean (15%)	1,50	6,53	12,33	0,18	2,58	0,15	0,09	0,28
Q3	2,00	9,75	15,75	0,50	3,75	0,00	0,12	1,00
Q2 Median	1,00	4,50	8,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	4,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	3,59	10,35	20,42	0,33	1,84	0,72	0,45	0,60
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	18,00	62,00	106,00	0,91	6,00	3,00	2,00	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Jcoverage-1.0.5	1,49	4,35	9,56	0,24	1,78	0,39	0,81	0,28
Bounded mean (15%)	1,23	3,70	8,17	0,20	1,62	0,19	0,10	0,21
Q3	2,00	5,00	14,00	0,50	2,00	0,00	0,00	0,00
Q2 Median	1,00	3,00	5,00	0,00	1,00	0,00	0,00	0,00
Q1	0,00	2,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	1,87	4,46	9,59	0,34	1,05	0,96	0,37	0,52
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	7,00	25,00	46,00	1,00	5,00	4,00	1,67	2,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean easymock-2.0	1,41	5,83	12,54	0,15	1,24	0,09	0,12	0,33
Bounded mean (15%)	1,24	4,13	8,32	0,11	1,08	0,00	0,02	0,16
Q3	2,00	5,00	13,50	0,33	1,00	0,00	0,00	0,00
Q2 Median	1,00	3,00	3,50	0,00	1,00	0,00	0,00	0,00
Q1	1,00	3,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	1,34	7,51	19,25	0,24	0,67	0,46	0,41	0,73
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	6,00	38,00	105,00	0,85	4,00	3,00	2,00	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean struts-1.2.8	2,91	8,60	18,84	0,28	2,59	0,46	0,51	0,96
Bounded mean (15%)	2,09	6,66	13,21	0,25	2,45	0,24	0,33	0,67
Q3	4,00	11,00	22,00	0,67	4,00	1,00	0,60	1,00
Q2 Median	2,00	4,00	8,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,56	11,02	29,13	0,36	1,48	1,13	0,95	2,04
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	40,00	82,00	260,00	0,98	7,00	10,00	5,00	28,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean JHotDraw60b1	1,40	9,51	13,36	0,16	2,84	0,57	0,31	0,73
Bounded mean (15%)	1,09	7,72	10,31	0,11	2,68	0,07	0,16	0,38
Q3	2,00	11,00	14,00	0,00	4,00	0,00	0,32	1,00
Q2 Median	1,00	7,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	4,00	5,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	1,86	10,40	16,76	0,30	1,49	3,84	0,74	1,70
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	19,00	90,00	158,00	1,50	9,00	71,00	8,00	19,00

Figura 1 Resultados globales

Como ejemplo se muestra en la figura 2 un diagrama de cajas (eliminando mínimos y máximos), centrándonos en los cuartiles Q1 y Q3 como límites de las cajas. Los productos se ordenan de izquierda a derecha, de mayor a menor número de clases. Como se puede observar gráficamente las diferencias de distribución entre los productos son apreciables.

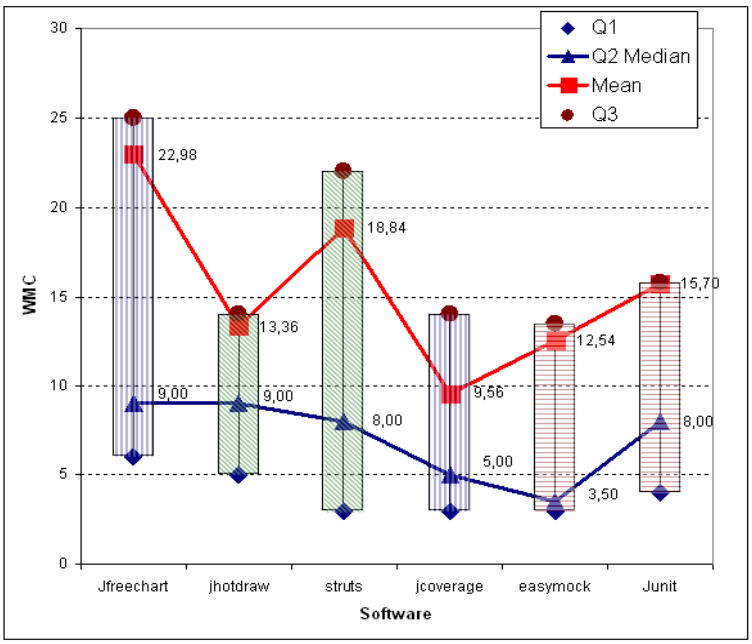

Figura 2 Distribución de la métrica WMC

En trabajos previos, establecimos la posibilidad de utilizar umbrales de métricas con el objetivo de detectar defectos (en diseño no en funcionalidad). Como primera aproximación, se establecieron esos umbrales en base a valores absolutos. Los resultados de este estudio empírico indican que los valores deberían estar ajustados a los productos concretos.
Otro factor que podría tener influencia en los resultados es la clase de producto (observar el tipo de relleno en de las cajas en la figura 2). Productos similares como: frameworks de pruebas (junit y easymock), frameworks de desarrollo (struts y jhotdraw) y bibliotecas (jcoverage y jfreechart), presentan grandes diferencias entre los valores mínimo y máximo.

A la vista de estos resultados, aparece una nueva hipótesis: *la ausencia de umbrales absolutos genera un cierta degeneración de las valores de las métricas (cuando los productos incrementan su tamaño las métricas están descontroladas)*. Para comprobar esta hipótesis, se presenta un nuevo caso de estudio con diferentes versiones de algunos productos.

3.3 Segunda fase: Evolución de versiones

Se han tomado diferentes versiones de tres de estos productos: jfreechart, jhotdraw y junit. Se muestran las versiones y número de clases de las versiones. Las versiones muestran la evolución del producto en un periodo medio de tiempo:

- jfreechart-1.0.1 (691 clases, 2006-01-27)
- jfreechart-1.0.0-pre2 (629 clases, 2005-03-10)
- jfreechart-0.9.21 (570 clases, 2004-09-10)
- jfreechart-0.9.7 (492 clases, 2003-04-17)
- jfreechart-0.9.4 (326 clases, 2002-10-18)

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-1.0.1	2,22	9,94	22,42	0,19	2,53	0,33	0,16	0,69
Bounded mean (15%)	1,27	7,27	15,25	0,15	2,46	0,03	0,09	0,48
Q3	2,00	11,00	23,00	0,40	3,00	0,00	0,17	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	4,00	7,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	4,86	15,18	39,39	0,31	1,12	1,41	0,35	1,18
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	46,00	173,00	513,00	1,00	7,00	14,00	3,33	8,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreeChart-1.0.0-pre2	2,40	10,08	22,98	0,21	2,55	0,36	0,16	0,69
Bounded mean (15%)	1,41	7,45	15,87	0,17	2,47	0,04	0,08	0,46
Q3	3,00	11,00	25,00	0,50	3,00	0,00	0,14	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	3,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	5,05	15,01	38,82	0,32	1,14	1,48	0,37	1,23
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	48,00	166,00	490,00	1,00	7,00	16,00	3,20	9,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-0.9.21	2,38	9,99	22,47	0,21	2,52	0,36	0,16	0,66
Bounded mean (15%)	1,41	7,33	15,44	0,17	2,45	0,05	0,08	0,44
Q3	2,00	12,75	26,00	0,50	3,00	0,00	0,16	1,00
Q2 Median	1,00	5,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	3,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	4,93	15,20	38,66	0,32	1,12	1,47	0,37	1,20
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	47,00	155,00	473,00	0,96	7,00	16,00	3,00	8,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-0.9.7	2,14	7,03	15,63	0,20	3,21	0,31	0,17	0,49
Bounded mean (15%)	1,22	5,06	11,08	0,15	3,07	0,04	0,07	0,28
Q3	2,00	9,00	19,00	0,50	4,00	0,00	0,09	1,00
Q2 Median	0,00	3,00	6,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	1,00	3,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	4,55	10,65	24,45	0,32	1,97	1,34	0,44	1,02
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	39,00	87,00	203,00	1,00	7,00	15,00	3,00	7,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jfreechart-0.9.4	2,69	7,77	18,00	0,26	3,02	0,40	0,27	0,61
Bounded mean (15%)	1,71	6,13	13,51	0,22	2,85	0,08	0,11	0,32
Q3	3,00	11,00	24,00	0,62	4,00	0,00	0,16	1,00
Q2 Median	1,00	4,00	8,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	1,00	3,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,87	9,70	25,11	0,35	1,95	1,49	0,70	1,34
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	39,00	60,00	195,00	1,00	7,00	16,00	6,00	8,00

Figura 3 Resultados JFreeChart

En el caso de jhotdraw las versiones, número de clases y fechas son:

- jhotdraw-6.0b1 (497 clases, 2004-02-01)
- jhotdraw-5.4b2 (478 clases, 2004-01-31)
- jhotdraw-5.3 (208 clases, 2002-01-20)
- jhotdraw-5.2 (149 clases, 2001-02-18)

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean JHotDraw60b1	1,40	9,51	13,36	0,16	2,84	0,57	0,31	0,73
Bounded mean (15%)	1,09	7,72	10,31	0,11	2,68	0,07	0,16	0,38
Q3	2,00	11,00	14,00	0,00	4,00	0,00	0,32	1,00
Q2 Median	1,00	7,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	0,00	4,00	5,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	1,86	10,40	16,76	0,30	1,49	3,84	0,74	1,70
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	19,00	90,00	158,00	1,50	9,00	71,00	8,00	19,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jhotdraw54b1	1,41	9,67	13,89	0,16	2,90	0,58	0,32	0,73
Bounded mean (15%)	1,12	7,85	10,80	0,11	2,75	0,08	0,17	0,39
Q3	2,00	11,00	15,00	0,00	4,00	0,00	0,33	1,00
Q2 Median	1,00	7,00	9,00	0,00	3,00	0,00	0,00	0,00
Q1	1,00	4,00	6,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	1,81	10,33	16,88	0,30	1,48	3,89	0,75	1,71
Minimum	0,00	0,00	0,00	0,00	1,00	0,00	0,00	0,00
Maximum	16,00	88,00	148,00	1,50	9,00	71,00	8,00	19,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jhotdraw53	1,83	9,12	15,51	0,27	2,65	0,86	0,51	1,21
Bounded mean (15%)	1,46	7,07	11,60	0,23	2,43	0,20	0,41	0,97
Q3	3,00	10,00	18,00	0,63	3,00	0,00	0,75	2,00
Q2 Median	1,50	6,50	12,50	0,00	1,00	0,00	0,00	0,00
Q1	0,00	3,00	4,75	0,00	2,00	0,00	0,00	0,00
Standard Deviation	2,38	10,87	20,54	0,35	1,66	3,53	0,66	1,66
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	17,00	72,00	146,00	1,50	8,00	40,00	3,00	12,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean jhotdraw52	1,83	8,30	13,53	0,26	2,81	0,68	0,56	1,28
Bounded mean (15%)	1,52	6,37	10,25	0,23	2,60	0,24	0,48	1,06
Q3	3,00	10,00	15,25	0,60	3,00	0,00	1,00	2,00
Q2 Median	1,00	5,00	8,00	0,00	2,00	0,00	0,28	1,00
Q1	0,00	3,00	4,00	0,00	2,00	0,00	0,00	0,00
Standard Deviation	2,19	9,82	16,84	0,33	1,70	1,91	0,66	1,69
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	14,00	61,00	108,00	1,50	8,00	12,00	3,11	12,00

Figura 4 Resultados JHotDraw

En el caso de junit las versiones y número de clases son:

- junit-3.8.1 (47 clases)
- junit-3.2 (32 clases)
- junit-2.1 (19 clases)

	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit 3.8.1	2,17	8,13	15,70	0,21	2,70	0,28	0,18	0,35
Bounded mean (15%)	1,50	6,53	12,33	0,18	2,58	0,15	0,09	0,28
Q3	2,00	9,75	15,75	0,50	3,75	0,00	0,12	1,00
Q2 Median	1,00	4,50	8,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	4,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	3,59	10,35	20,42	0,33	1,84	0,72	0,45	0,60
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	18,00	62,00	106,00	0,91	6,00	3,00	2,00	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit 3.2	2,72	7,94	14,75	0,25	2,56	0,19	0,13	0,34
Bounded mean (15%)	1,68	5,96	11,29	0,22	2,43	0,04	0,09	0,25
Q3	3,00	11,00	18,50	0,50	3,50	0,00	0,19	1,00
Q2 Median	1,00	3,50	5,50	0,00	2,00	0,00	0,00	0,00
Q1	0,00	2,00	2,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,85	11,65	21,05	0,34	1,93	0,64	0,24	0,65
Minimum	0,00	0,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	20,00	60,00	103,00	0,92	6,00	3,00	0,75	3,00
	NOF	NOM	WMC	LCOM	DIT	NSC	SIX	NORM
Mean Junit 2.1	2,16	8,11	14,05	0,22	2,53	0,32	0,31	0,58
Bounded mean (15%)	1,35	7,18	12,41	0,19	2,41	0,18	0,17	0,47
Q3	2,00	8,50	18,00	0,50	3,00	0,00	0,26	1,00
Q2 Median	1,00	4,00	6,00	0,00	2,00	0,00	0,00	0,00
Q1	0,00	4,00	4,00	0,00	1,00	0,00	0,00	0,00
Standard Deviation	4,06	8,52	15,30	0,31	1,61	0,82	0,70	0,84
Minimum	0,00	1,00	1,00	0,00	1,00	0,00	0,00	0,00
Maximum	18,00	31,00	55,00	0,89	6,00	3,00	3,00	3,00

Figura 5 Resultados JUnit

Para cada uno de estos productos se han recogido las métricas anteriormente mencionadas. Se calculan de nuevo: media, media acotada, desviación estándar, cuartil Q1, mediana, cuartil Q3, mínimo y máximo.

Como se puede observar (ver Figuras 3-5), los resultados para cada producto se mantienen entre versiones, con pequeñas variaciones. Esto sugiere que una vez que el producto es estable, pese a que su tamaño aumenta (llegando en casi todos los casos estudiados a duplicarse el número de clases), los umbrales establecidos pueden ser relativamente correctos.

3.4 Conclusiones del estudio

De los resultados obtenidos se pueden deducir las siguientes conclusiones:

- Los umbrales se deben establecer de manera relativa al producto.
- Estos umbrales se pueden mantener entre distintas versiones del mismo producto.
- El tipo de producto (*framework* / biblioteca) no determina el cómo deben ser estos umbrales.

A partir de estas conclusiones se plantean nuevos problemas. Ante productos nuevos sería difícil decidirse por unos umbrales iniciales. Como primera estimación se podrían tomar valores de productos similares (mismo dominio, funcionalidad similar, tamaño estimado). En el caso de disponer de varias versiones del producto, el tomar los valores de dichas versiones parece una buena medida inicial. En ambos casos posiblemente sea necesario realizar algún ajuste.

4. Aplicación de umbrales relativos

En anteriores trabajos, se ha mostrado la utilidad del uso de métricas como indicador de la presencia de lo que se denominan “malos olores” o síntomas (*bad smells*). Este vocabulario, es propio de refactoring, aunque se puede generalizar a defectos del software. Se ha planteado el uso de umbrales para sugerir la presencia de dichos defectos. Por otro lado, se definió un framework que soportase todos estos conceptos. Sin embargo la definición de los umbrales es algo que quedó abierto [5]. En la Figura 6, se muestra un diagrama de cajas de dos distribuciones de datos: la distribución ideal y la distribución de la métrica WMC en JFreeChart 1.0.1. Se consideran valores altos y bajos a aquellos valores por encima y por debajo del bigote más alto y del bigote más bajo respectivamente. En un proceso ideal los valores fuera de los bigotes (*outlayer*) se deberían eliminar aplicando detección y corrección de *bad smells*.

Como se ha concluido anteriormente, el uso de los mismos umbrales para diferentes productos no parece adecuado y el ajuste de los mismos debería ser un proceso asistido. A continuación se muestra una revisión de uno de los resultados obtenidos en trabajos previos, aplicando todo lo obtenido hasta ahora.

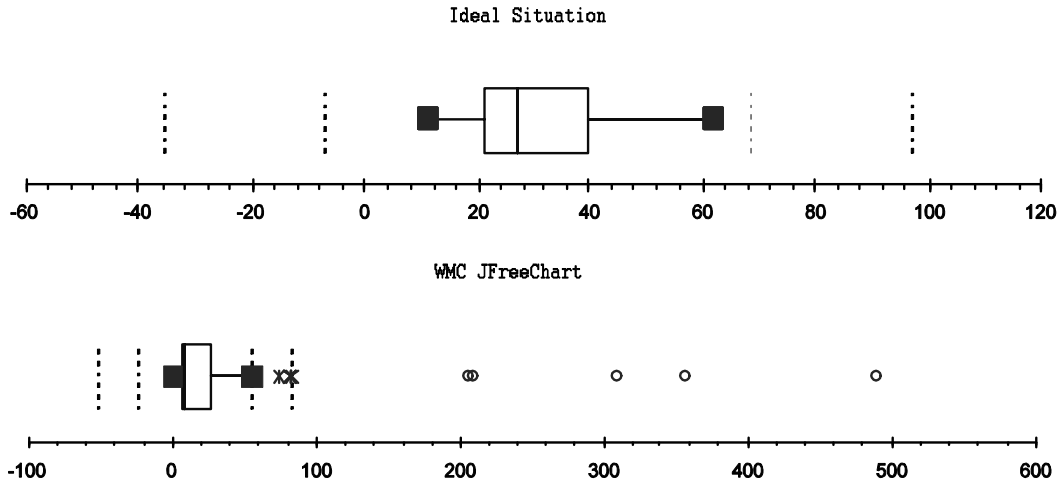

Figura 6 Diagrama de Box Plot

4.1 Detección de un *bad smell*: *Lazy Class*

Revisando de nuevo la definición [4], se trata de “clases que no están haciendo mucho por sí mismas y que podrían ser eliminadas”. Los criterios que se establecieron eran un bajo número de métodos y atributos, con complejidad baja y bajo valor de DIT. Además se puede añadir algún criterio adicional como pequeño número de hijos.

El problema de este tipo de planteamientos es: ¿qué se considera bajo en este sistema? Como muy bien se apuntó en [8,10], ciertos sistemas son particulares y puede que las conclusiones generales extraídas no sean correctas. Si se toman como umbrales los tres cuantiles Q1 de NOF, NOM y WMC.
Como se observa existen ciertas discrepancias en los filtros (NOF<=Q1 AND NOM<=Q1 AND WMC<=Q1) a utilizar. Por ejemplo, ¿qué ocurre si aplicamos los valores recogidos en junit a jfreechart como filtro? En este caso para jfreechart se obtienen como sospechosas 63 clases mientras que utilizando los valores propios de jfreechart el número de clases es de 97. La diferencia de número de clases recogida muestra que no se pueden aplicar los mismos criterios sobre uno u otro producto.

4.2 Detección de un *bad smell*: *Large Classes*

Aparentemente, este *bad smell* parece uno de los más sencillos de detectar. Sin embargo, como se ha podido extraer anteriormente, cada sistema puede tener sus particularidades. En concreto, si se establece el mismo umbral para todos, cabe la posibilidad de tomar un valor muy grande que aplicado a muchos productos no devuelva ningún resultado. Puede ocurrir lo contrario y es que al elegir un valor pequeño se tomen demasiados elementos en otros conjuntos.

Utilizando el conocimiento de cómo se distribuyen los datos, se buscan valores entre los que se denominan extremos. En concreto, si se fijan como valor umbral el cuantil Q3 para cada producto y combinando las métricas NOF/NOM/WMC.

Observando los valores, se intuye el problema que supondría aplicar los valores de filtrado de JFreeChart a jcoverage o viceversa. En estos casos nos quedamos con los elementos que están por encima del 75% de elementos en la población. Más concretamente, con aquellos cuyos valores de métricas coinciden en el extremo derecho de sus distribuciones (NOF>=Q3 AND NOM>=Q3 AND WMC>=Q3). Idealmente se debería poder ajustar este valor para localizar aquellos elementos que se denominan *outlayers* de la distribución. Si repetimos el proceso de aplicar, por ejemplo el filtro de junit a jfreechart nos encontramos con 148 clases sospechosas. Por el contrario, aplicando el filtro de jfreechart sólo tenemos 108.

Los resultados demuestran la gran diferencia de aplicar unos u otros umbrales. Aunque obviamente la corrección final venga sujeta a una determinación manual de cuantas clases no son falsos positivos, sí que es evidente la diferencia de escalas.

5. Conclusiones y trabajo futuro

El presente trabajo, ha establecido a partir de un caso de estudio, la idoneidad del uso de umbrales relativos, aún considerando que se pueden seguir utilizando y combinando con valores absolutos. El uso de esta última solución no deja de ser recomendable, de hecho se sigue habilitando dicha solución con los perfiles de métricas, aunque como se ha visto cada producto establece normalmente sus propios límites.

La escasa diferencia de resultados entre distintos tipos de producto indica que no se puede usar esto como discriminante. Es el tamaño del producto el que limita en muchos casos los valores de las métricas. Por otro lado, otras métricas son menos sensibles a estos efectos.

En este trabajo no se ha pretendido obtener con precisión umbrales, ni añadir nuevos métodos de obtención de los mismos, sino comprobar empíricamente la necesidad de uso y el soporte a la definición particular de los mismos para cada tipo de producto. El proceso de detección se debería repetir hasta alcanzar una distribución estable intentando reducir el número de

outlayers. Estos resultados podrían ayudar a asistir el mantenimiento sistemático del software, hasta obtener una versión estable.
Finalmente, se ha extendido la solución ya propuesta para incorporar la declaración de *bad smells*, junto con el *framework* de métricas. Esto permite que la herramienta final sea capaz de asistir al usuario a la hora de establecer los criterios de detección de defectos de una forma más objetiva.

Obviamente, existe un gran número de líneas abiertas:

- Se deben validar los resultados, aumentando el número de productos bajo estudio.
- Se debe comprobar si los resultados están influenciados por el lenguaje de programación.
- Aunque no se ha mencionado, la formación y cultura de los programadores puede influir en que el producto establezca sus límites.

Aclaraciones y agradecimientos

Una versión inglesa de este trabajo fue publicada y presentada previamente por los mismos autores en 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, disponible en http://www.inf.usi.ch/faculty/lanza/QAOOSE2006/QAOOSE2006Proc.pdf

La traducción del trabajo ha sido financiada por el Ministerio Español de Ciencia e Innovación a través del proyecto de investigación ROADMAP (TIN2008-05675).

Referencias

1. Francisca Muñoz Bravo. A Logic Meta-Programming Framework for Supporting the Refactoring Process. PhD thesis, Vrije Universiteit Brussel, Belgium, 2003.
2. Shyam R. Chimdaber and Chris F. Kemerer. A metrics suite for object oriented design. IEEE Transactions On Software Engineering, 20:476–493, 1994.
3. Yania Crespo, Carlos L´opez, Raul 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, jul 2005.
4. Martin Fowler. Refactoring. Improving the Design of Existing Code. Number 0-201-48567-2. Addison-Wesley, 2000.
5. V.A. French. Establishing software metric thresholds. 9th International Workshop on Software Measurement, 1999.
6. Mark Lorenz and Jeff Kidd. Object-oriented software metrics: a practical guide. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
7. Mika Mäntylä. Bad Smells in Software - a Taxonomy and an Empirical Study. PhD thesis, Helsinki University of Technology, 2003.
8. Mika Mäntylä, Jari Vanhanen, and Casper Lassenius. Bad smells - humans as code critics. In 20th IEEE International Conference on Software Maintenance, 2004.
9. Radu Marinescu. Detecting design flaws via metrics in object-oriented systems. In Proceedings of the TOOLS, USA 39, Santa Barbara, USA, 2001.
10. Radu Marinescu. Measurement and Quality in Object-Oriented Design. PhD thesis, Faculty of Automatics and Computer Science, october, 2002.
11. Raul Marticorena. Analysis and definition of a language independent refactoring catalog. In 17th Conference on Advanced Information Systems Engineering (CAiSE 05). Doctoral Consortium, Porto, Portugal., page 8, jun 2005.http://gnomo.fe.up.pt/ caise/.
12. Tom Tourwè and Tom Mens. Identifying Refactoring Opportunities Using Logic Meta Programming. In Proc. 7th European Conf. on Software Maintenance and Reengineering, pages 91 – 100, Benvento, Italy, 2003. IEEE Computer Society.