Umbrales sugeridos para promedios de métricas de diseño de una aplicación en Java

Jorge Ramirez, Gustavo Gil, Carina Reyes

Universidad Nacional de Salta, Facultad de Ciencias Exactas, Centro de Investigación y Desarrollo en Informática Aplicada (CIDIA)
{ramirezj, gdgil,reyescarina}@cidia.unsa.edu.ar

Resumen. Las métricas de producto de software pueden aportar información relevante para la gestión de proyectos de software. La determinación de umbrales para esas métricas permitiría disponer de información rápida que advierta sobre potenciales problemas en los productos bajo análisis. Sin embargo, no existe amplio consenso en cuanto a estos indicadores. El Software Libre y de Código Abierto (F/OSS por sus siglas en inglés) ofrece la posibilidad de estudiar grandes cantidades de versiones a partir de la libre disponibilidad del código fuente que caracteriza a las licencias bajo las cuales se distribuyen esos productos. Este trabajo propone umbrales para métricas a partir del estudio de las distribuciones de frecuencia del promedio de la cantidad de métodos por clase y de la proporción de referencias a métodos de otras clases respecto del total de métodos definidos en la aplicación.

Palabras Clave: Métricas de Diseño, Software Libre y de Codigo Abierto. Umbrales para métricas.

1 Introducción

La caracterización de una pieza de software con miras a evaluar su potencial adopción o reutilización, requiere de disponer de elementos de juicio suficientes para la toma de decisiones. En particular, al momento de seleccionar un producto para incorporarlo a un nuevo desarrollo, es necesario contar con indicadores que orienten esta determinación.

No existe consenso general en la ingeniería de Software acerca de la vinculación entre métricas y los distintos atributos del software [1]; un ejemplo en este sentido es el estudio empírico presentado por Sjoberg y otros [2] que revela que varias métricas asociadas habitualmente a la mantenibilidad (el índice de mantenibilidad, distintas medidas de tamaño y cohesión) son inconsistentes entre sí. Fowler [3] considera que ninguna combinación de métricas puede rivalizar con la intuición humana informada; compartiendo en general ese punto de vista, no se busca postular aquí relaciones causales sino indicadores que actúen como advertencia de posibles debilidades en el diseño. Lanza y Marinescu [4], en tanto, señalaron que no existe un umbral perfecto, cuya superación signifique necesariamente un cambio cualitativo en el atributo que se pretende evaluar, pero pueden ser indicativos de utilidad en la práctica.

Desde esta perspectiva se aborda la posible determinación de umbrales de métricas de software, es decir, valores que al superarse alerten sobre posibles debilidades en el diseño o sobre riesgo potencial respecto de la mantenibilidad de código.

Para ello en este trabajo se analizan las distribuciones de frecuencias de un conjunto de métricas sobre una muestra de versiones de aplicaciones F/OSS escritas en Java pertenecientes a proyectos que han publicado 7 versiones o más, lo que refleja la evolución de estos proyectos.

El enfoque adoptado aquí se diferencia del que sigue la gran mayoría de los estudios que proponen umbrales, dado que se buscan elementos para evaluar el diseño en su conjunto mediante valores promedio en lugar de determinar valores que sirvan de referencia para módulos o clases particulares.

En concreto, se plantea detectar qué métricas referidas al diseño de una aplicación muestran una distribución de frecuencias tal que sugiera la conveniencia de adoptar determinados umbrales. El trabajo se organiza de la siguiente manera: en el apartado siguiente se ofrece una revisión de trabajos similares; a continuación, se describe la metodología utilizada en la investigación; posteriormente se analizan los datos obtenidos y se proponen umbrales para algunas métricas; a continuación, se comparan estos valores con los sugeridos en otros trabajos. Finalmente, en la sección 5 se exponen las conclusiones y los trabajos futuros.

2 Trabajos Relacionados

Se han propuesto diversos umbrales para métricas de diseño orientado a objetos definidas mediante diversos métodos. Por un lado, algunos trabajos utilizan técnicas estadísticas que intentan relacionar las mediciones obtenidas con atributos asociados a la mantenibilidad o a la propensión a errores de alguna parte de código o un elemento de diseño [5] [6] [7] [8] . Otros, en cambio, se basan en la distribución de los valores de las métricas, buscando revelar cuáles serían cifras atípicas para los indicadores que consideran [9] [10] [11] [4] [12] .

En la mayoría de los estudios las métricas que se abordan son las propuestas por Chidamber y Kemerer [13], y Brito e Abreu y Carapuça [14] entre otras de uso común.

2.1 Estudio de Umbrales en Base a Correlaciones Estadísticas

Los trabajos que se resumen a continuación apuntan a determinar valores de las métricas que pudieran advertir sobre debilidades en un módulo o una clase en concreto.

Benlarbi y otros [5] plantearon la determinación de umbrales para los valores de las métricas orientadas a objetos. En el estudio consideran a los umbrales como valores heurísticos que sirven para determinar rangos de valores deseables y no deseables para métricas de software. Los autores analizaron las métricas propuestas por Chidamber y Kemerer sobre dos sistemas de telecomunicaciones escritos en C++ en los que clasificaron a cada clase como “defectuosa” o “no defectuosa” según si se había reportado al menos una falla durante la operación del sistema.

Los autores concluyeron que el establecimiento de umbrales no aporta ventajas respecto de un modelo continuo.

Shatnawi realizó diversos estudios buscando determinar umbrales a partir de diferentes análisis estadísticos. En uno de ellos [6], emplea la curva ROC para proponer umbrales de un conjunto de métricas CK y de Lorentz. Ese método permite evaluar la pertinencia de un umbral para clasificar en dos categorías (en este caso, en defectuoso y no defectuoso). En un trabajo posterior, el mismo autor [7] empleó regresión logística (como hiciera anteriormente Benlarbi) para estudiar valores aceptables para un conjunto de métricas; los resultados por fuera de esos valores indicarían un mayor riesgo de presencia de defectos. Este autor analizó diferentes versiones del popular IDE Eclipse, realizando posteriormente una comparación con otros dos productos (Rhino y Mozilla); concluye que no pueden generalizarse las cifras postuladas en el primer caso.

El autor también compara los umbrales calculados con los propuestos por Rosenberg [9], concluyendo que éstos son menos seguros para estimar clases defectuosas.

Kaur [8] y otros obtuvieron las métricas CK y otras indicativas de posibles "bad smells" o antipatrones [15] sobre dos versiones del software JFreeChart. Emplearon regresión logística, estimando posibles umbrales en base al Nivel Aceptable de Riesgo (Value for Acceptable Risk Level, VAR) de acuerdo a la metodología elaborada por Bender [16] para analizar posibles "efectos de umbral". De este resumen se desprende que los objetivos buscados principalmente por los trabajos relevados se relacionan con la determinación de métricas que permitan caracterizar piezas individuales de software, más que caracterizar el diseño de las aplicaciones en su conjunto.

2.2 Estudio de Umbrales en Base a Distribuciones de Frecuencia de las Métricas

Rosenberg [9] utilizó histogramas para analizar el efecto de los valores de las métricas de Chidamber y Kemerer [13] sobre la calidad del software. Los valores de los umbrales se basan en el análisis de los histogramas, aunque no se analiza la posible relación entre esos valores y la probabilidad de que módulos presenten errores o deban modificarse.

Erni y Lewerentz [10] analizaron la distribución de métricas orientadas a objetos postulando como potenciales umbrales a la media ± la desviación estándar. Los autores plantean la adopción de uno de esos valores en función de los problemas que plantearía en cada caso cifras demasiado altas o demasiado bajas. Por ejemplo, si para una métrica se considera que un número alto es perjudicial, se adopta $\mu+\sigma$ como umbral (siendo μ la media y σ la desviación estándar).

Chidamber y otros [11] consideraron las distribuciones empíricas de las métricas sugiriendo que el tope para definir un valor como "alto" sería el 80º percentil (es decir, el 80% de las muestras alcanzarían valores por debajo del umbral señalado).

Lanza y Marinescu [4] también propusieron umbrales para métricas comunes partiendo de la distribución de los valores en un grupo de 45 aplicaciones escritas en Java. Clasificaron las medidas según sus frecuencias en bajo, medio, alto y muy alto.

Ferreira y otros [12] estudiaron las distribuciones de las métricas COF (Factor de Acoplamiento), cantidad de métodos públicos, cantidad de campos (atributos) públicos, LCOM (Falta de Cohesión en los métodos) y DIT (Profundidad del árbol de herencia)- Analizaron 40 aplicaciones escritas en Java, computando las métricas con

la herramienta Connecta, y las distribuciones de frecuencia se ajustaron a distribuciones teóricas con la herramienta EasyFit. Los autores dividieron la muestra según diferentes dominios, planteando umbrales distintos para cada uno de ellos.

Recientemente Filó y otros [17] presentaron un estudio de diversas métricas a partir de distribuciones de frecuencia encontradas en una muestra de 111 aplicaciones escritas en Java. A partir de ese análisis proponen umbrales para 17 métricas diferentes. El trabajo muestra similitudes con el que se expone aquí dado que se basa en una muestra públicamente accesible y emplea el análisis de distribución de frecuencia como metodología para detectar posibles umbrales; no obstante, el objetivo de ese estudio es proponer valores referidos a debilidades de clases o paquetes concretos, más que obtener indicadores referidos el diseño en su conjunto.

3 Metodología

Para este trabajo se tuvo en cuenta la infraestructura de investigación propuesta por Gaser y Scacchi [18] para los estudios empíricos sobre F/OSS. A fin de posibilitar la repetibilidad y revisión del trabajo, se utilizaron versiones públicamente accesibles que incluyan su código fuente; las herramientas empleadas siguen el mismo criterio: las mediciones se realizaron con iPlasma [19] y el tratamiento estadístico se efectuó con el entorno R [20].

Se descargó el código fuente de todas las versiones a analizar y se obtuvieron las métricas que se detallan en la tabla 1.

La selección de las métricas apunta a una visión global del software a estudiar; indagando en las características generales del código fuente. Los proyectos analizados tienen en común el haber producido numerosas versiones y que han sido utilizados de manera efectiva por muchos usuarios. Son proyectos "exitosos" en los términos de Crowston [21] y Weiss [22].

Por ello se eligieron las medidas propuestas por Lanza y Marinescu [4], quienes plantean una perspectiva de conjunto para caracterizar el diseño de una aplicación.

Tabla 1. Métricas computadas en el presente trabajo

Métrica	Descripción
CYCLO	Complejidad ciclomática, según la definición de McCabe[22]
LOC	Líneas de código, incluyendo líneas en blanco y comentarios
NOM	Cantidad de métodos y otras operaciones definidas en toda la aplicación
NOC	Cantidad de clases
NOP	Cantidad de paquetes
CALL	Cantidad de invocaciones a operaciones
FOUT	Suma de las clases referenciadas por cada una de las clases

Cada una de estas métricas arroja un valor global dependiente del tamaño del software. Es esperable que una aplicación más grande tenga un mayor número de

clases, mayor cantidad de líneas de código, etc. Lanza y Marinescu plantean entonces la caracterización a partir de las proporciones que se muestran en la tabla 2.

Tabla 2. Proporciones analizadas

Nombre	Descripción	Fórmula
Estructuración de alto nivel	Cantidad de clases por paquete	NOC/NOP
Estructuración de clases	Cantidad de operaciones en relación con le número de clases	NOM/NOC
Estructuración de operaciones	Promedio de la cantidad de líneas de código por operación	LOC/NOM
Complejidad de operación intrínseca	Cantidad de caminos independientes en relación con el tamaño en líneas de código	CYCLO/LOC
Intensidad de acoplamiento	Proporción de los métodos que invocan a cada operación en relación con la cantidad total de operaciones	CALL/NOM
Dispersión de acoplamiento	Proporción de las llamadas de cada clase a métodos de otra respecto del total de llamadas	FOUT/CALL

En este trabajo se computan las métricas mencionadas para 560 versiones de 28 proyectos diferentes escritos en Java. Si bien iPlasma no extrae WMC (métodos ponderados por clase) [13], se puede obtener a partir de la Complejidad Intrínseca, la Estructuración de Operaciones y la Estructuración de Clases. Analizamos la distribución de WMC dada de su amplia utilización.

En total, se computaron 371.934 clases que incluyen 3.907.564 métodos, cubriendo 43.842.838 líneas de código.

4 Análisis de los Datos

La tabla 3 resume las medidas de localización y dispersión [23] para las proporciones estudiadas:

Tabla 3. Medidas localización y dispersión para las métricas computadas

	1º Cuartil	Media	Mediana	3º Cuartil	Desviación estándar	Coef. de Variación	Coef. de asimetría
Complejidad Intrínseca	0,1727	0,2076	0,2053	9,2236	0,041	0,13	0,87
Estructuración de Operaciones	7,973	10,27	10,86	14,310	4,277	0,41	0,89
Estructuración de Clases	9,581	10,91	11,74	13,33	3,453	0,29	0,93
Estructuración de Alto Nivel	7,503	9,98	12,55	17,81	6,965	0,7	1,41
Intensidad de Acoplamiento	0,556	0,588	0,585	0.616	0,168	0,29	17,63
Dispersión de acoplamiento	2,534	2,992	3,024	3,399	0,82	0,27	0,13

Las distribuciones de estas proporciones muestran que la Complejidad Intrínseca, la Intensidad de Acoplamiento, la Estructuración de Clases, y la Dispersión de acoplamiento exhiben menor dispersión. De ellas, la Intensidad de acoplamiento presenta una distribución de frecuencias con una asimetría más marcada.

Se analizan a continuación las gráficas de distribución que se muestran en las figuras 1 a 4; posteriormente, se considera la variación entre deciles, a fin de observar en cuál de ellos se constata mayor distancia entre los valores.

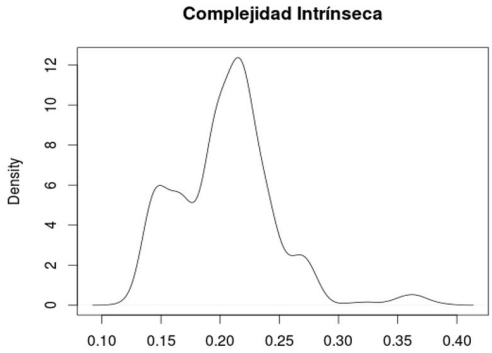

Fig. 1. Distribución de la Complejidad Intrínseca.

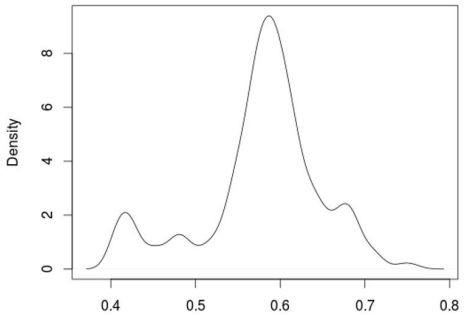

Fig. *2*. *Distribución de la Intensidad de acoplamiento*

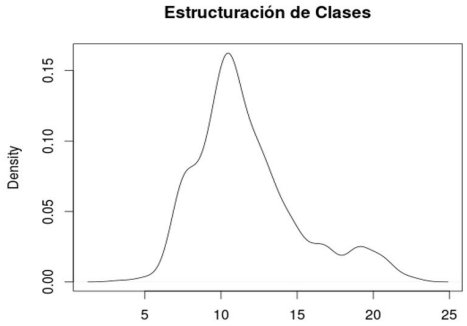

Fig. *3*. *Distribución de la Estructuración de Clases*

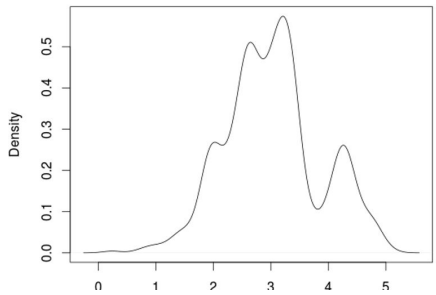

Fig. *4.* *Distribución de la Dispersión de acoplamiento*

Respecto de la Estructuración de clases, la gráfica muestra una alta concentración en torno a la moda. En tanto, la Estructuración de Operaciones exhibe varios picos, con una caída pronunciada en las frecuencias de valores mayores a 20.

Las distribuciones de frecuencias de la Estructuración de Clases y de la Intensidad de Acoplamiento muestran claramente una zona modal a partir de las cuales las frecuencias decrecen en ambos sentidos. Analizando la variación de los deciles podemos proponer umbrales superiores buscando cambios pronunciados entre los valores sucesivos de los mismos, lo que implicaría un descenso rápido de las frecuencias. La tabla 4 resume los deciles para las distribuciones de las métricas consideradas.

Tabla 4. Valores que correspoden a los diferentes deciles para las distribucionesobservadas en las proporciones analizadas.

Decil	C.I.	E.C.	E.O.	D.A.	I.A.	WMC
"0%"	0,121	3,375	3,832	0,234	0,406	7,346
"10%"	0,148	7,721	5,707	1,968	0,468	9,774
"20%"	0,166	9,001	6,876	2,372	0,544	16,233
"30%"	0,191	9,950	8,218	2,603	0,567	18,103
"40%"	0,196	10,458	9,425	2,750	0,575	20,463
"50%"	0,208	10,905	10,272	2,992	0,588	22,000
"60%"	0,216	11,730	11,301	3,172	0,594	23,772
"70%"	0,221	12,639	13,622	3,337	0,607	26,589
"80%"	0,233	14,092	14,820	3,665	0,625	36,062
"90%"	0,251	16,936	15,830	4,270	0,668	45,683
"100%"	0,385	22,773	29,321	5,084	0,759	182

En la Fig. 5 Observamos también la distribución de WMC, calculada a partir de otras métricas según se señaló más arriba.

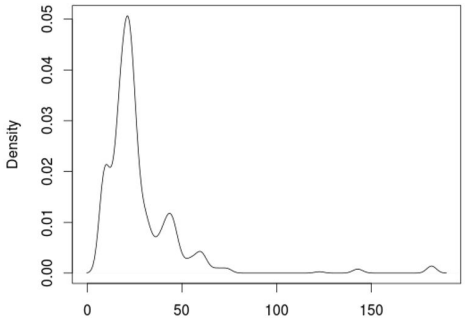

Fig. 5. *Distribución de WMC (Métodos ponderados por clase)*

Respecto de la estructuración de clases, se observa que entre el tercer y el octavo decil (60% de los casos) tienen en promedio entre 9 y 14,09 métodos por clase en promedio. Por otra parte, analizando el histograma (sesgado hacia la izquierda) se observa que la frecuencia de los valores superiores a 13 decrece significativamente, lo que convierte a ese valor en un posible umbral.

En cuanto a la Intensidad de acoplamiento, el decrecimiento más pronunciado comienza con el noveno decil , por lo que un valor superior a 0,668 podría reflejar acoplamiento excesivo.

Si contemplamos WMC, vemos que la curva de distribución de frecuencias tiene un máximo posterior a la moda; recién luego del 9º decil se verifica un descenso constante, por lo que el umbral para esta métrica sería de 45,68.

4.1 Umbrales Propuestos en Otros Trabajos

La tabla 5 muestra los valores propuestos en diversos trabajos para la métrica WMC

Tabla 5. Valores propuestos en otros trabajos como umbral para WMC.

Métricas	Shatnawi [7]	Rosenberg [9]	Lanza y Marinescu [4]	Herbold et al [24]
WMC	20	100	47	100

Como puede observarse, las cifras postuladas son muy disímiles. Cabe señalar, no obstante, que el valor planteado por Lanza y Marinescu es similar al obtenido en este trabajo sobre una muestra mucho mayor.

5 Conclusiones y Trabajos Futuros

Este trabajo adopta dos enfoques novedosos respecto de la búsqueda de umbrales útiles de métricas de diseño orientado a objetos. Por un lado, se buscan referencias para valores promedio, indicativos del diseño general de una aplicación o de un componente de software, en lugar de proponer referencias para clases particulares o módulos particulares.

Por otra parte, se retoma el análisis de distribuciones de frecuencia de los valores de las métricas, buscando topes identificables a partir de las curvas de distribución de frecuencias.

En concreto, una aplicación que presente más de 14 métodos por clase, o un promedio de WMC por encima de 46, muestra una estructuración muy poco común, con clases con demasiados métodos; de esto no se concluye que tal o cual clase se potencialmente defectuosa, sino que el diseño en su conjunto tiende a agrupar demasiadas funciones sobre las clases.

Si bien la métrica compuesta denominada Intensidad de Acoplamiento muestra un comportamiento que sugiere un tope viable, es una medida que pocas herramientas computan. No obstante, puede advertir que el diseño tiende a que las clases requieran de muchas otras para realizar sus tareas.

Teniendo en cuenta estos aspectos, sería conveniente adoptar la métrica Estructuración de Clases (promedio de la cantidad de métodos por clase) por su sencillez, o WMC, que es captada por una amplia cantidad de herramientas. Cabe señalar, sin embargo, que al basarse los umbrales propuestos en valores promedio, éstos pueden no ser relevantes para aplicaciones pequeñas, dadas las características del promedio como medida de tendencia central influenciada por valores extremos.

En futuros trabajos se prevé analizar otras métricas relacionadas con el acoplamiento, y estudiar su comportamiento con relación a las distintas etapas de la evolución de una aplicación. De esta forma, esperamos conocer diversos aspectos de la evolución del software y generar más elementos para una evaluación preliminar de una aplicación F/OSS a partir de la información que se puede obtener de su código fuente.

6 Referencias

1. Bourque, P., Wolff, S., Dupuis, R., Sellami, A., Abran, A.: Lack of Consensus on Measurement in Software Engineering: Investigation of Related Issues. In: 14th International Workshop on Software Measurement (IWSM) IWSM-Metrikon 2004, Konigs Wusterhausen. pp. 321–333. Shaker-Verlag, Magdeburg, Germany (2004).
2. Sjøberg, D.I.K., Anda, B., Mockus, A.: Questioning Software Maintenance Metrics: A Comparative Case Study. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 107–110. ACM, New York, NY, USA (2012).
3. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional (1999).
4. Lanza, M., Marinescu, R., Ducasse, S.: Object-Oriented Metrics in Practice. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005).
5. Benlarbi, S., Emam, K.E., Goel, N., Rai, S.N.: Thresholds for Object-Oriented Measures. In: ISSRE. pp. 24–39. IEEE Computer Society (2005).
6. Shatnawi, R., Li, W., Swain, J., Newman, T.: Finding software metrics threshold values using ROC curves. J. Softw. Maint. Evol. Res. Pract. 22, 1–16 (2010).
7. Shatnawi, R.: A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems. IEEE Trans. Softw. Eng. 99, 216–225 (2010).

8. Kaur, S., Singh, S., Kaur, H.: A Quantitative Investigation Of Software Metrics Threshold Values At Acceptable Risk Level. Int. J. Eng. Res. Technol. 2, (2013).
9. Rosenberg, L.: Applying and Interpreting Object Oriented Metrics. Presented at the Software Technology Conference (1998).
10. Erni, K., Lewerentz, C.: Applying Design-metrics to Object-oriented Frameworks. In: Proceedings of the 3rd International Symposium on Software Metrics: From Measurement to Empirical Results. p. 64–. IEEE Computer Society, Washington, DC, USA (1996).
11. Chidamber, S.R., Darcy, D.P., Kemerer, C.F.: Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis. IEEE Trans Softw Eng. 24, 629–639 (1998).
12. Ferreira, K.A.M., Bigonha, M.A.S., Bigonha, R.S., Mendes, L.F.O., Almeida, H.C.: Identifying Thresholds for Object-oriented Software Metrics. J Syst Softw. 85, 244–257 (2012).
13. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Trans Softw Eng. 20, 476–493 (1994).
14. Brito e Abreu, F., Carapuca, R.: Object-Oriented Software Engineering: Measuring and Controlling the Development Process. In: Proc. Int'l Conf. Software Quality (QSIC) (1994).
15. Fowler, M., Beck, K.: Bad Smells in Code. In: Refactoring: Improving the design of existing code. pp. 75–88. Addison-Wesley (1999).
16. Bender, R.: Quantitative Risk Assessment in Epidemiological Studies Investigating Threshold Effects. 41, 305–319 (1999).
17. Filó, T., Bigonha, M., Ferreira, K.A.M.: A Catalogue of Thresholds for Object-Oriented Software Metrics. In: SOFTENG 2015. pp. 48–55 (2015).
18. Gasser, L., Scacchi, W.: Towards a Global Research Infrastructure for Multidisciplinary Study of Free/Open Source Software Development. In: Russo, B., Damiani, E., Hissan, S., Lundell, B., and Succi, G. (eds.) Open Source Development, Communities and Quality. pp. 143–158. Springer, Boston (2008).
19. Marinescu, C., Marinescu, R., Mihancea, P.F., Ratiu, D., Wettel, R.: iPlasma: An Integrated Platform for Quality Assessment of Object-Oriented Design. In: ICSM (Industrial and Tool Volume). pp. 77–80 (2005).
20. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2013).
21. Crowston, K., Annabi, H., Howison, J.: Defining Open Source Software Project Success. In: in Proceedings of the 24th International Conference on Information Systems (ICIS 2003). pp. 327–340 (2003).
22. Weiss, D.: Measuring success of open source projects using web search engines. In: First International Conference on Open Source Systems (2005).
23. Devore, J.L.: Probabilidad y estadística para Ingeniería y ciencias. CENGAGE Learning (2005).
24. Herbold, S., Grabowski, J., Waack, S.: Calculation and Optimization of Thresholds for Sets of Software Metrics. Empir. Softw Engg. 16, 812–841 (2011).1. Bourque, P., Wolff, S., Dupuis, R., Sellami, A., Abran, A.: Lack of Consensus on Measurement in Software Engineering: Investigation of Related Issues. In: 14th International Workshop on Software Measurement (IWSM) IWSM-Metrikon 2004, Konigs Wusterhausen. pp. 321–333. Shaker-Verlag, Magdeburg, Germany (2004).