Métricas de Código fuente y Evolución de F/OSS: un estudio exploratorio

Jorge Ramirez, Carina Reyes, Gustavo Gil

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

Resumen: La disponibilidad pública del código fuente en el Software Libre y de Código Abierto (F/OSS) constituye una abundante fuente de información que permite estudiar la evolución de gran cantidad de software desde el punto de vista del tamaño y de los cambios en el diseño a lo largo del tiempo. Este trabajo explora la evolución de diversas métricas de diseño a partir del código fuente de las versiones publicadas por 15 proyectos de F/OSS, con miras a detectar posibles tendencias que posibiliten, a futuro, caracterizar la evolución del diseño en un proyecto de software y su relación con la mantenibilidad y adaptabilidad de las aplicaciones producidas en esos proyectos.

Palabras clave: Evolución del Software, Software Libre y de Código Abierto, Métricas, estudio exploratorio

1 Introducción

El presente trabajo se enmarca en la búsqueda de elementos que permitan evaluar y -eventualmente- comparar diferentes aplicaciones de Software Libre y Código Abierto (F/OSS, por sus siglas en inglés) con miras a su reutilización, a partir del estudio de las características de las distintas versiones de esas aplicaciones.

Nos planteamos alcanzar un conocimiento más profundo de la evolución de proyectos de software en busca de indicios que brinden información sobre el diseño de una aplicación a partir de su historia.

Específicamente, nuestra indagación se orienta hacia la caracterización del diseño de una aplicación F/OSS en función de métricas extraídas del código fuente, a fin de obtener información relativa a su reusabilidad potencial. Entre las actividades que realizamos en esta dirección, estamos explorando la evolución de proyectos de F/OSS, los cambios en la estructura del código a lo largo de las distintas versiones según se refleja en distintas métricas.

La información disponible públicamente sobre el desarrollo de proyectos F/OSS es heterogénea en contenido y calidad[1] y -por ende- difícil de utilizar para estudios

comparativos; sin embargo, todas las aplicaciones de este tipo se distribuyen junto al código fuente, debido justamente a la definición de las licencias de Software Libre y de Código Abierto[2][3].

En un trabajo anterior[4] habíamos observado cómo un software en particular (SweetHome 3D) había evolucionado de manera tal que un conjunto de métricas e indicadores seleccionados, obtenidos a partir del código fuente, mostraba una tendencia de crecimiento marcadamente lineal a lo largo de las sucesivas versiones de esa aplicación; el incremento verificado en esos indicadores mostró, además, una pendiente suave.

Aquella exploración tomó en consideración la replicabilidad del estudio[5], por lo que se utilizaron datos públicamente accesibles y herramientas libres o de acceso gratuito. Las métricas observadas intentan caracterizar globalmente el diseño de una aplicación a partir del código fuente, según la propuesta de Lanza y Marinescu[6]

En términos de tamaño, la evolución de aquel software se asemeja al de varios casos de F/OSS estudiados por diferentes autores[7][8]; al mismo tiempo, muestra marcadas diferencias con la forma en que evolucionan otros proyectos que tienden a crecer a una velocidad mayor a la lineal.

Se han verificado crecimientos aparentemente atípicos -es decir, diferentes al esperable según las denominadas “leyes de la evolución de software” de Lehmann- en varios proyectos de Software Libre y de Código Abierto. Por ejemplo, Godfrey y Tu observaron que el núcleo de Linux crece de manera supra-lineal a lo largo del tiempo, medido en líneas de código [9], en tanto que los estudios empíricos informados por Lehman y otros encontraron que los sistemas de software tienden a crecer linealmente o según la inversa del cuadrado del tamaño[10]. Las leyes propuestas por Lehman [11]afirman que el Software tiende a cambiar con el tiempo (ley I), aumentar su complejidad (ley II) y aumentar la cantidad de funciones que realiza (ley VI), entre otras. El incremento de la complejidad sería el responsable de la tendencia a disminuir la velocidad de crecimiento del software[12].

Los resultados del trabajo mencionado anteriormente[4] nos llevaron a preguntarnos:

- si el comportamiento observado en la evolución de SweetHome 3D es frecuente, si es característico en el software libre y de código abierto o si -por el contrario – constituye una excepción
- si hay métricas o combinaciones de métricas cuyo comportamiento a lo largo de la evolución de este tipo de software muestra una tendencia definida, que podría servir de base para la predicción de las características de futuras versiones.
- si hay métricas o combinaciones de métricas que podrían ser representativas para caracterizar la evolución de F/OSS desde el punto de vista del diseño.

Para comenzar a abordar estos interrogantes, nos planteamos recabar de un conjunto de versiones publicadas por diferentes proyectos F/OSS, los mismos datos que en el trabajo mencionado más arriba[4], con el fin de reunir nuevos elementos sobre la evolución del F/OSS en relación con los cambios en la estructura del código fuente.

El resto del trabajo se organiza de la siguiente forma: en la siguiente sección repasamos un conjunto de trabajos relacionados con la evolución del software, enfatizando aquellos que presentan resultados de investigaciones empíricas sobre F/OSS y la evolución del código fuente; posteriormente, explicamos la metodología seguida para recabar los datos analizados en el presente trabajo; a continuación, presentamos un resumen de los datos obtenidos; en el siguiente apartado exponemos un análisis de los mismos. Finalmente, exponemos las conclusiones y delineamos las hipótesis que guiarán nuestra investigación en el futuro.

2 Trabajos Relacionados

La primera formulación de las leyes de la evolución del software[13] surgió del estudio conducido por Lehman sobre la historia de las diferentes versiones del sistema operativo IBM OS360/OS 370. Años más tarde, los mismos autores exploraron la aplicabilidad de los enunciados propuestos en el trabajo anterior, mediante el análisis de las distintas versiones de una aplicación comercial[11]; a partir de este trabajo, y del proyecto FEAST, Lehman y sus colegas reformularon los enunciados de las leyes, destacando el papel de la retroalimentación (feedback) en el proceso evolutivo del software[14][15]. Estas experiencias llevaron a sugerir que el crecimiento del software tiende a seguir un modelo lineal, en algunos casos, o según la inversa del cuadrado del tamaño, en otros[16]. La disminución en la velocidad del crecimiento se explicaría por la creciente complejidad del software a lo largo de las sucesivas versiones[14].

La disponibilidad pública del código fuente posibilitó que muchos investigadores estudiaran empíricamente la evolución del software libre y de código abierto.

Godfrey y Tu[9] analizaron el crecimiento del núcleo de Linux a lo largo de 96 versiones; estos autores computaron el tamaño en líneas de código no comentadas, en lugar de módulos (como sugiere Lehman), tomando como variable independiente al tiempo transcurrido desde la primera versión considerada (en lugar de número de secuencia de versión). Los autores observaron que el tamaño del núcleo de Linux crece a tasas superiores a la lineal, a diferencia de los casos reportados por Lehman y Turski.

Koch[7] realizó una investigación a gran escala sobre evolución del F/OSS, aprovechando la disponibilidad del código de las sucesivas versiones de 8621 proyectos alojados en el repositorio SourceForge. El autor observa que una parte de los proyectos analizados tienden a crecer a tasas supralineales.

Un enfoque con mayores similitudes a nuestro planteamiento se presenta en un trabajo de Capiluppi[17] en el que se estudia la evolución de 12 proyectos de F/OSS respecto de una serie de atributos, entre los que se destaca el tamaño (medido en número de carpetas, número de módulos y cantidad de archivos en el código fuente), la cantidad de desarrolladores involucrados, el lapso entre la publicación de versiones, etc., cotejando por pares los distintos indicadores en busca de relaciones; el escrito

profundiza en las características de la evolución de 4(cuatro) proyectos, observando en todos ellos un crecimiento lineal respecto del número de secuencia de versión.

Capiluppi y otros[18] observaron que la evolución del F/OSS parece presentar etapas bien diferenciadas, alternando períodos de alto crecimiento del tamaño con otros en los que ese incremento se produce en menor medida. Los autores plantean la hipótesis de que esos últimos períodos se vincularían con correcciones de errores, en tanto que los otros reflejarían la incorporación de nuevas funcionalidades.

Pei-Breivold y otros[19] publicaron una revisión sistemática de estudios sobre la evolución del F/OSS. Los autores muestran que se han utilizado diferentes métricas para medir el crecimiento del software a lo largo del tiempo, aunque la mayoría utiliza líneas de código o módulos. Del relevamiento realizado por estos autores también surgen cuáles fueron las temáticas abordadas con mayor frecuencia y qué métricas se utilizaron; de allí se evidencia la falta de consenso en cuanto a criterios, metodologías e incluso vocabulario que aún existen en la comunidad científica en relación con la evolución del software.

Entre los trabajos empíricos sobre la evolución del F/OSS hay diferencias en cuanto a la métrica utilizada para medir el tamaño (LOC, SLOC, cantidad de módulos, cantidad de carpetas)[20]; a su vez, el tamaño suele contrastarse con el número de secuencia de versión (RSN) o con la fecha de publicación de la versión[21][9]

3 Metodología

Como señalamos más arriba, este trabajo parte del resultado de un estudio de caso[4] realizado con anterioridad.

Nos propusimos observar si las características evolutivas de SweetHome 3D se verifican en las versiones publicadas por otros proyectos, por lo que computamos -para un conjunto mayor de programas-, las mismas métricas que en el trabajo citado.

3.1 Selección de los proyectos

En este trabajo observamos 15 proyectos de F/OSS. Buscamos proyectos desarrollados en Java, que hubieran producido 7 versiones o más, y en los que el código fuente correspondiente estuviera disponible. Fijamos este umbral a partir de la cantidad de versiones disponibles del proyecto FreePlane, que es un fork de un proyecto estable (Freeplane) y que cuenta con una cantidad importante de usuarios y desarrolladores[22]. La restricción de utilizar aplicaciones en Java se origina en las posibilidades de la herramienta utilizada[23] y en la intención de comparar los datos con los obtenidos en el trabajo[4] que precede a este estudio.

Las aplicaciones elegidas se detallan en la Tabla 1. Incluimos a SweetHome 3D puesto que algunos de los análisis realizados aquí difieren de los efectuados en el trabajo anterior.

Tabla 1. Proyectos seleccionados para el presente trabajo

Nombre	Descripción	Versiones
Vuze	Cliente Bittorrent	14
PMD	Analizador de código fuente	13
Hodoku	Generador de Sudoku	12
iText	Librería para gestionar formato PDF	17
Art of Illusions	Modelador gráfico 3D	28
FreeMind	Herramienta para crear mapas mentales	16
FreePlane	Herramienta para crear mapas mentales	7
Chemistry Development Kit	Librería orientada a bioquímica y química	28
Tiled	Editor de mapas para juegos	11
Batik	Librería para gestionar formato SVG	19
jHotDraw	Framework para gráficos estructurados y técnicos	16
Plotdigitizer	Digitalización de información gráfica	11
jPicEdt	Editor de dibujo vectorial	18
JfreeChart	Librería para gráficos	20
Sweethome3D	Aplicación para diseño de interiores	19

Dado el carácter exploratorio del presente trabajo, y no previendo realizar inferencia estadística o generalizaciones, descartamos aquellos para los cuales el código fuente no estuviera inmediatamente accesible desde la Web, éste no estuviera completo en la mayoría de las versiones publicadas, o no se dispusiera fácilmente de la fecha de en que se habían lanzado las diferentes versiones.

3.2 Métricas utilizadas

Las métricas fueron obtenidas mediante la herramienta iPlasma[23], que puede descargarse libremente de http://loose.upt.ro/reengineering/research/iplasma.

Las métricas registradas por esta aplicación se detallan en la Tabla 2.

Tabla 2. Métricas recogidas por la herramienta iPlasma

Métrica	Descripción
CYCLO	Complejidad ciclomática
LOC	Líneas de código
NOP	Número de operaciones (métodos y otras operaciones definidas por el usuario)
NOC	Número de clases
NOP	Número de paquetes
CALLS	Cantidad de invocaciones diferentes a operaciones (métodos y funciones globales). Si un mismo método llama varias veces a la misma operación, se cuenta una sola vez
FANOUT	Clases cuyos métodos son invocados desde las distintas operaciones

La herramienta extrae valores globales para estas métricas, es decir, números correspondientes a la totalidad del código fuente examinado. Para obtener un panorama general del diseño, Lanza y Marinescu proponen una serie de cocientes que vinculan pares de métricas, según se detalla en la Tabla 3.

Los datos obtenidos se copiaron en planillas de cálculo, a fin de realizar análisis estadísticos y comparar con los resultados obtenidos anteriormente sobre SweetHome 3D[4].

En las planillas incorporamos el número de secuencia de versión (RSN) y la fecha de lanzamiento de cada versión; este último dato se registra para análisis ulteriores.
En este trabajo optamos por utilizar líneas de código (LOC) como medida del tamaño, por resultar simple de computar, aunque registramos también la cantidad de clases y de métodos, como se menciona más arriba.

El estudio se basa en el Número de Secuencia de Versión (RSN) como sugiriera Lehman[12]. La razón de esta elección, en lugar de la fecha de publicación, es que el estudio que nos planteamos apunta a obtener conocimiento de la evolución de las versiones, atendiendo al posible aumento paulatino de la complejidad[11] y de la dificultad de modificar o adaptar el código fuente.

3.3 Herramientas

Como se mencionó anteriormente, se utilizaron herramientas que pudieran obtenerse libremente. Además de la mencionada iPlasma, se emplearon LibreOffice y OpenOffice para el tratamiento estadístico de la información obtenida.

Tabla 3. Proporciones calculadas entre las métricas y su descripción

Métrica	Cociente	Descripción
CI	CYCLO/LOC	**Complejidad Intrínseca**: cantidad de caminos por línea de código
EO	LOC/NOM	**Estructuración de Operaciones**: líneas de código por método.
EC	NOM/NOC	**Estructuración de Clases**: promedio de métodos por clase.
EAN	NOC/NOP	**Estructuración de Alto Nivel**: cantidad promedio de clases por paquete
IAc	CALLS/NOM	**Intensidad de Acoplamiento**: promedio de llamadas hechas desde cada método a otros métodos
DA	FANOUT/CALLS	**Dispersión de Acoplamiento**: indica el promedio de las invocaciones provenientes de otras clases

4 Datos obtenidos

No podemos incluir aquí la totalidad de los datos obtenidos, debido a la magnitud de los mismos.

En los proyectos Vuze, Hodoku e iTex no fue posible recuperar el código fuente de algunas versiones; no obstante, los incluimos en el análisis asumiendo que los valores faltantes podrían incidir en el estudio específico de esos proyectos, pero al contar con la información de la mayoría de las versiones podemos tener un panorama de la evolución de estos proyectos.

En la propia planilla se calcularon los cocientes entre métricas propuestos por Lanza y Marinescu[6].

A modo de ilustración, presentamos en la Tabla 4 los valores registrados para FreeMind

Tabla 4. Valores de las métricas obtenidas con iPlasma para las distintas versiones de la aplicación FreeMind. Se presentan a título ilustrativo

Versión	RSN	Fecha	CYCLO	LOC	NOM	NOC	NOP	FOUT	CALL
0.0.2	1	27/06/00	532	2268	392	28	7	599	922
0.0.3	2	09/07/00	575	2448	421	25	6	773	1194
0.1.0	3	05/08/00	706	2932	477	41	8	887	1369
0.2.0	4	02/11/00	725	3024	475	40	8	891	1406
0.3.0	5	24/03/01	998	4229	647	58	10	1190	1907
0.3.1	6	27/03/01	1010	4290	649	58	10	1205	1935
0.4	7	07/07/01	1271	5378	783	64	10	1512	2419
0.5	8	24/08/02	1569	6856	897	67	10	1758	2952
0.6	9	01/02/03	1750	7216	1017	69	10	1945	3272
0.6.1	10	08/02/03	1764	7264	1023	69	10	1951	3285
0.6.5	11	04/09/03	1996	8425	1092	70	10	2133	3664
0.6.7	12	25/10/03	2270	9535	1229	74	10	2412	4189
0.7.1	13	21/03/05	2733	11613	1448	92	10	2828	4972
0.8.0	14	07/09/05	4660	22033	2645	199	16	4786	8286
0.8.1	15	26/02/08	7939	41682	4982	513	30	7738	13226
0.9.0	16	18/02/11	7982	36911	4475	361	32	8507	14027

5 Análisis de los datos

El estudio de caso de SweetHome 3D[4] mostró que a lo largo de 18 versiones el código fuente publicado por ese proyecto mostraba un crecimiento lineal tanto en el conjunto de métricas obtenidas como en la mayoría de los cocientes con los que caracterizábamos el diseño[6], según se observa en los gráficos estadísticos descriptivos.

Para comprobar la hipótesis de linealidad, calculamos el coeficiente de correlación de Pearson[24] para cada una de las métricas y e indicadores en relación con las líneas de código de cada versión. El cómputo muestra que para la Estructuración de Operaciones, la Estructuración de Clases, la Intensidad de Acoplamiento y la Dispersión de Acoplamiento se obtiene un coeficiente de Pearson superior a 0,9 (véase al final de la Tabla 5). En el caso de la Complejidad Intrínseca, en cambio, se registra un valor aproximado de 0,8331.

Realizamos la misma operación para el conjunto de proyectos considerados (es decir, computamos el coeficiente de Pearson entre los valores obtenidos para los indicadores y la cantidad de Líneas de Código de cada versión), encontrando los valores que se muestran en la Tabla 5.

Tabla 5. Coeficientes de correlación de los distintos indicadores de diseño en las diferentes versiones respecto de las líneas de código de cada una

Proyecto	CI	EO	EC	EAN	IAc	DA
jPicEdt	0,2882	-0,3251	0,0553	0,1515	-0,3685	-0,5813
AOI	-0,2810	-0,0330	0,7990	-0,5896	-0,3562	0,8520
PMD	-0,9238	-0,7310	-0,1740	-0,1987	0,6742	-0,4647
iText	-0,9257	0,1119	-0,2774	0,6361	0,6088	-0,8479
FreeMind	-0,8936	0,7618	-0,3405	0,9377	-0,4704	-0,0251
plotdigitizer	-0,8840	0,2062	0,4747	0,9442	-0,1964	0,8159
CDK	0,3643	0,5929	-0,6768	-0,8170	-0,7796	0,5755
Tiled	-0,3437	0,8542	-0,7688	0,9202	-0,2212	0,8908
Hodoku	-0,9932	0,8587	-0,5822	-0,6209	-0,3055	0,7079
Vuze	-0,8789	0,8681	0,6989	0,5737	-0,7052	-0,4220
Batik	0,5244	0,8794	0,9288	0,3755	-0,6558	0,9163
JfreeChart	-0,8054	0,7060	0,9526	0,3216	0,7133	0,9632
FreePlane	-0,9452	0,9141	-0,0141	0,9841	-0,7539	0,8763
jHotDraw	-0,8349	0,8762	0,6348	-0,7537	-0,8174	0,7070
SH3D	0,8331	0,9850	0,9588	0,4503	-0,9079	0,9255
Promedios	-0,4466	0,5017	0,1779	0,2210	-0,3028	0,3926
Desviación	0,6369	0,5291	0,6315	0,6582	0,5490	0,6573

La penúltima fila muestra el promedio de los coeficientes de Pearson obtenidos, en tanto que la última muestra la desviación estándar de esos valores

Se observa que la mayoría de los proyectos no exhibe la tendencia lineal que muestra SweetHome 3D en varios indicadores; únicamente FreePlane exhibe valores que sugieren una relación lineal entre varios indicadores y las líneas de código. Más aún, estas cifras se distribuyen según una amplia dispersión de los datos, según revela la última fila, donde se registra la desviación estándar correspondiente a cada columna.

De los 15 proyectos, 9 presentan una correlación negativa posiblemente significativa entre la Estructuración de Operaciones y las Líneas de Código, con valores del coeficiente menores a -0,8. Por otra parte, en 7 proyectos encontramos una posible correlación positiva entre la Estructuración de Operaciones (EO) y el tamaño en Líneas de Código, y también 7 exhiben similar comportamiento entre la Dispersión de Acoplamiento y el tamaño. De estos dos grupos, hay cuatro proyectos que muestran coeficientes de correlación superiores 0,8 tanto para la Estructuración de Operaciones como para la Dispersión de Acoplamiento: Sweethome3D, FreePlane, Batik y Tiled. Esto sugiere que para esos proyectos los métodos tienden a crecer linealmente con el tamaño de la versión (en LOC, recordemos) y el acoplamiento entre clases tiende a aumentar su dispersión con el crecimiento de la aplicación (lo que apoya la afirmación de Lehman sobre el incremento paulatino de la complejidad[12])

A fin de observar de qué manera evolucionan los indicadores globales (CI, EO, EC, EAN, IAc y DA), elaboramos gráficos de líneas reflejando el valor de estos indicadores en proporción al valor inicial (es decir, cada uno de los valores computados del indicador divididos por el valor que alcanzaba en la primera versión).

En todos los casos estudiados, encontramos que las proporciones parecen tender a un valor constante. En las figuras 1 y 2 mostramos estas gráficas para los proyectos JFreeChart y Tiled

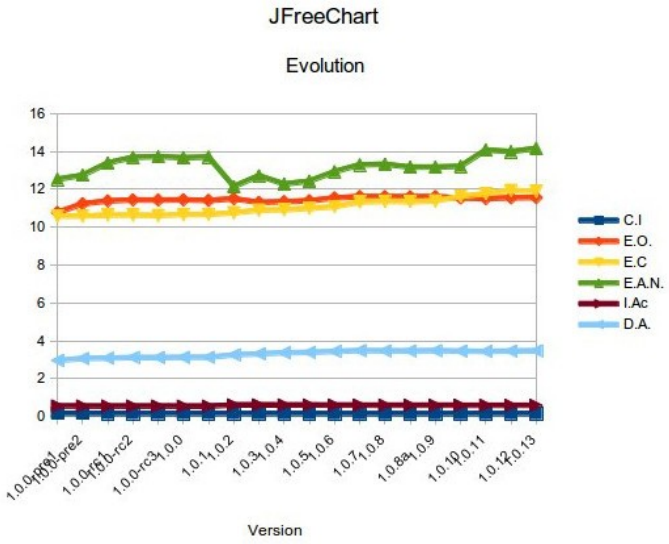

Fig. 1: Evolución de los indicadores para JFreeChart a lo largo de las versiones

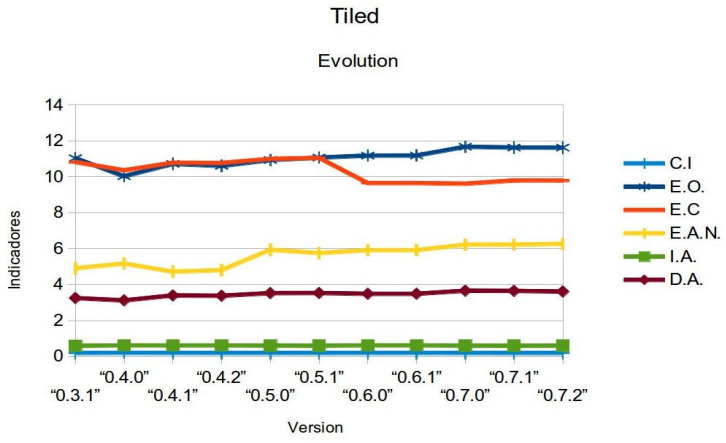

Fig. 2: Evolución de los indicadores para Tiled a lo largo de las versiones

6 Conclusiones

Este estudio de carácter exploratorio nos permitió observar la evolución de indicadores del diseño basadas en el código fuente a lo largo de distintas versiones de varios proyectos de F/OSS.

El análisis nos permite comprobar que las características de evolución lineal de diversos indicadores observados anteriormente en SweethHome 3D[4], no se aprecia en las sucesivas versiones publicadas por otros proyectos de F/OSS.

Por otra parte, ninguno de los indicadores calculados presenta el mismo comportamiento en la totalidad de proyectos.

Sin embargo, la Estructuración de Operaciones y la Dispersión de Acoplamiento parecen mostrar un crecimiento lineal respecto del tamaño de las versiones de varios proyectos. Esto sugiere que estos indicadores podrían caracterizar la evolución de los proyectos de F/OSS; un análisis más masivo podría determinar si es posible agrupar o tipificar la evolución del diseño de aplicaciones F/OSS a partir de los valores de estas métricas derivadas.

Los gráficos que muestran la variación del conjunto de indicadores analizados respecto del número de versión, podrían indicar tendencias hacia valores constantes, al menos en ciertas etapas de la evolución de proyectos F/OSS, etapas que aún están por caracterizarse.

El número de proyectos y la forma en que fueron seleccionados no permiten arriesgar generalizaciones; sin embargo, a partir de las observaciones presentadas aquí, nos planteamos a futuro ampliar el número de proyectos estudiados con miras a realizar un análisis estadístico relevante, indagar acerca de las posibles tendencias de los valores de los indicadores a lo largo de las versiones y, eventualmente, esbozar criterios para la caracterización de la evolución de proyectos F/OSS.

Referencias

1. Howison J & Crowston K:The perils and pitfalls of mining sourceforge. *Proc. of mining software repositories workshop at the international conference on software engineering (icse)* (2004)
2. Free software definition. http://www.fsf.org/licensing/essays/free-sw.html.
3. Open source initiative. http://www.opensource.org/docs/osd.
4. Ramirez J, Gimson L & Gil G:Evaluación de la evolución del diseño en f/oss:un caso de estudio. *Vii workshop ingeniería de software - xvi congreso argentino de ciencias de la computación* (2010)
5. González-Barahona, M.; Robles, G.:On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Softw. Engg.* .1-2. pp.75-89.(2012)
6. Lanza M, Marinescu R & Ducasse S.:*Object-oriented metrics in practice*. . Springer-Verlag New York, Inc., (2005).
7. Koch S:Software evolution in open source projects---a large-scale investigation. *J. Softw. Maint. Evol.* .6. pp.pp. 361-382.(2007)
8. Herraiz I:A statistical examination of the evolution and properties of libre software. *Icsm* (2009)

9. Godfrey MW & Tu Q:Evolution in open source software: a case study. *In proceedings of the international conference on software maintenance* (2000)
10. Lehman, M.M.; Ramil,J.F.; Perry, D.E.:On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution.. *IEEE METRICS* (1998)
11. Lehman M:Laws of software evolution revisited. *Ewspt '96: proceedings of the 5th european workshop on software process technology* (1996)
12. Lehman MM, Ramil JF, Wernick PD, Perry DE & Turski WM:Metrics and laws of software evolution - the nineties view. *Proceedings of the 4th international symposium on software metrics* (1997)
13. Lehman, M.; Belady, L..:*Program evolution: processes of software change*. Lehman, M. M. and Belady, L. A.. Academic Press Professional, Inc., (1985).
14 Lehman M & Fernández-Ramil JC:Software evolution. *Software evolution and feedback*. Nazim H Madhavji, Juan C Fernández-Ramil, Dewayne E Perry (Ed.). pppp. 7-40(2006)
15. Lehman MM, Ramil JF & Perry DE:On evidence supporting the feast hypothesis and the laws of software evolution. *Proceedings of the 5th international symposium on software metrics* (1998)
16. Turski WM:Reference model for smooth growth of software systems. *IEEE Trans. Softw. Eng.* .. pp.pp. 599-600.(1996)
17. Capiluppi A:Models for the evolution of os projects. *Icsm '03: proceedings of the international conference on software maintenance* (2003)
18. Capiluppi A, Gonzalez-Barahona JM, Herraiz I & Robles G:Adapting the staged model for software evolution to free/libre/open source software. *Ninth international workshop on principles of software evolution: in conjunction with the 6th esec/fse joint meeting* (2007)
19. Pei-Breivold H, Chauhan MA & Babar MA:A systematic review of studies of open source software evolution. *17th asia pacific software engineering conference (apsec), ieee* (2010)
20. Herraiz, I.; Robles,G.; Gonzalez-Barahona, J.M.:Comparison between SLOCs and number of files as size metrics for software evolution analysis. *Proceedings of the Conference on Software Maintenance and Reengineering* (2006)
21. Robles G, Amor JJ, Gonzalez-Barahona JM & Herraiz I:Evolution and growth in large libre software projects. *Iwpse '05: proceedings of the eighth international workshop on principles of software evolution* (2005)
22. Public directory of Free and Open Source Software. http://www.ohloh.net.
23. Marinescu, C.; Marinescu,R.; Mihancea,P.F.; Ratiu,D.; Wettel, R.:iPlasma: An Integrated Platform for Quality Assessment of Object-Oriented Design. *ICSM (Industrial and Tool Volume)* (2005)
24. Devore J.:*Probabilidad y estadística para ingeniería y ciencias*. Sergio Cervantes González (Ed.). CENGAGE Learning, (2005).