Revista Cubana de Ciencias Informáticas
Vol. 13, No. 1, Enero-Marzo, 2019
ISSN: 2227-1899 | RNPS: 2301
Pág. 45-60
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Software Libre
Recibido: 10/05/2018 | Aceptado: 29/10/2018

Método para determinar comunidades de desarrollo y actores más influyentes en repositorios de sistemas operativos libres

Method to determining development communities and most influential actors in free software repositories

Jorge Alejandro Román Donates[1*], Vladimir Milián Núñez[2], Eliana Bárbara Ril Valentín[3], Raynel Batista Tellez[4]

[1]Departamento de Programación Facultad 2, Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 1/2.Torrens, La Lisa, La Habana, Cuba. jardonates@gmail.com
[2]Departamento de Ciencias Básicas Facultad 2, Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 1/2.Torrens, La Lisa, La Habana, Cuba. vmilian@uci.cu
[3]Departamento de Ingeniería de Software Facultad 2, Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 1/2.Torrens, La Lisa, La Habana, Cuba. ebril@uci.cu
[4]Dirección de Ciencia, Tecnología e Innovación, Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 1/2.Torrens, La Lisa, La Habana, Cuba. rainer@uci.cu

*Autor para correspondencia: jardonates@gmail.com

Resumen
Las comunidades de software libre consisten en grupos de usuarios o desarrolladores experimentados que contribuyen a la mejora del sistema operativo, su contribución puede verse de forma práctica en los repositorios de paquetes. El estudio de las interacciones que establecen los desarrolladores de estos paquetes a partir de intereses comunes, contribuye a identificar sus comunidades, promueve la colaboración entre equipos de desarrollo, ayuda a determinar los desarrollos críticos y actores más influyentes. El objetivo de esta investigación es desarrollar un método para determinar comunidades de desarrolladores y actores más influyentes en repositorios de sistemas operativos libres para fortalecer la colaboración entre equipos de desarrollo. En la investigación se realizó un estudio sobre conceptos asociados a la teoría de grafos, análisis de redes colaborativas, detección de comunidades y medidas de centralidad. Además, se describió el procedimiento que sigue el método presentado y se realizaron pruebas en aras de verificar la calidad de la solución. Como resultado final se obtuvo un método que facilitó la búsqueda de paquetes en repositorios de sistemas operativos libres, la extracción de los ficheros de control de cambios de cada uno de estos, la extracción de los nombres de paquetes y sus desarrolladores, así como la creación de una red colaborativa a partir de la relación entre desarrolladores y otra red con la relación paquete - desarrollador. El trabajo con Gephi permitió a su vez visualizar las redes colaborativas y detectar las comunidades y actores más influyentes.

Palabras claves: análisis de redes colaborativas, comunidades, detección de comunidades, medidas de centralidad, repositorios de sistemas operativos libres

Abstract
Communities consist of groups of experienced users or developers who contribute to the improvement of the operating system, the contribution of communities can be seen in a practical way in the package repositories.

Revista Cubana de Ciencias Informáticas
Vol. 13, No. 1, Enero-Marzo, 2019
ISSN: 2227-1899 | RNPS: 2301
Pág. 45-60
http://rcci.uci.cu

The study of the interactions that the developers of these packages establish based on common interests, helps to identify their communities, promotes collaboration between development teams, and helps to determine the critical developments, leaders, experts or most influential actors. The aim of this research is to develop a method for determining communities of developers and more influential actors in free software repositories. A study was carried out on concepts associated with graph theory, collaborative network analysis, algorithms for community detection and centrality measurements. In addition, the implementation process of the presented method was described and different tests were carried out in order to verify the quality of the solution. The final result was a method that facilitated the search for packages in free software repositories and the extraction of change control files from each of these. The method implemented facilitated the extraction of the names of packages and their developers. Gephi Toolkit allowed visualize the detected collaborative networks and distinguish the most influential communities and actors, allowing to strength the collaboration between development teams.

Keywords: *collaborative network analysis, communities, community detection, centrality measures, free software repositories.*

Introducción

Es innegable el papel que está jugando actualmente el software libre en el ámbito empresarial, gubernamental, académico, científico, entre otros. Hace unos pocos años, se consideraba una rareza y aventurarse en un proyecto de código abierto (del inglés *Open Source*) era cuanto menos, bastante arriesgado. En contraste, actualmente es muy difícil encontrar algún proyecto de software donde no exista al menos un componente de código abierto.

Los proyectos de código abierto son aquellos en los que el código está disponible libremente para que pueda ser accedido y copiado. Estos proyectos se desarrollan a través de una estructura descentralizada, donde cualquier persona capacitada puede enviar/proponer cambios y mejoras al código base. Como el software libre se usa comúnmente en muchos proyectos cerrados, también recibe contribuciones de muchas compañías.

Actualmente, en algunos ámbitos y áreas, los proyectos de código abierto están desplazando a sus homólogos privativos. Una de las mayores fortalezas de los proyectos abiertos, es la posibilidad de escalar el producto en base a las aportaciones de múltiples contribuidores, y que en el caso del software privativo queda reducido a los recursos de los que dispone la propia empresa. Es decir, que uno de los factores más importantes en el desarrollo y éxito de un proyecto de código abierto, corresponde al aporte de las comunidades de desarrollo de software.

El objetivo de una comunidad de desarrollo de software, es aglutinar una serie de individuos cuya intención es promover el acceso y distribución de una herramienta de software, permitiendo la libertad de uso, estudio, copia, modificación y redistribución a todo aquel que lo desee. La cooperación entre estas personas en todos los ámbitos de la producción del software (usuarios, desarrolladores, documentadores, testers, traductores, ?) permite generar las sinergias necesarias para conseguir una mejora sustancial de la calidad del software,

así como de una mayor difusión y sostenibilidad en el tiempo, y primando el beneficio de la sociedad sobre cualquier otro.

El principal resultado de este proceso, es la creación de un repositorio donde de aplicaciones de software. Un repositorio de software, no es más que un lugar de almacenamiento del cual pueden ser recuperados e instalados los paquetes de software en un ordenador. La principal función de estos repositorios, es facilitar la integración de código de posibles fuentes diferentes a una unidad operativa coherente e independiente.

En el ámbito del software libre, un repositorio puede contener miles de aplicaciones y/o paquetes, en cuyo desarrollo colaboran un alto número de desarrolladores. Tal es el caso del repositorio de Nova, la distribución cubana de GNU/Linux, donde un paquete puede tener uno o varios desarrolladores y un desarrollador colaborar en uno o más paquetes. Según el centro CESOL[1] encargado del desarrollo de Nova, dicha distribución alcanza la cifra de 70 000 paquetes al cierre del 2017, lo cual significa que un repositorio también puede ocupar una alta capacidad de almacenamiento e incrementarse al agregar nuevos paquetes y/o renovar otros, así como varias versiones de la distribución. Esta información sugiere que la dimensión de las comunidades y el nivel de actividad de sus miembros, ejemplo los desarrolladores de paquetes, sea elevada.

El estudio de las interacciones que establecen los desarrolladores de estos repositorios, a partir de intereses de desarrollo comunes, contribuye a identificar comunidades, promover la colaboración entre equipos de desarrollo, ayudar a determinar los desarrollos críticos, así como detectar los actores más influyentes (desarrolladores líderes o expertos). Sin embargo, actualmente en los repositorios de los sistemas operativos libres publicados en la UCI se accede manualmente a cada paquete para extraer información de sus desarrolladores. Esto además de resultar arduo por el volumen de información a procesar, eleva el margen de error, involucra más recursos, compromete los resultados esperados, el tiempo de ejecución de los análisis y la precisión de los datos obtenidos. Esto conlleva a que, la realización de estos análisis de forma manual pudiera distanciarlos de sus propósitos originales.

Teniendo en cuenta el contexto planteado cabría preguntarse entonces, ¿cómo determinar las comunidades de desarrolladores y actores más influyentes en los repositorios de los sistemas operativos libres para fortalecer la colaboración entre equipos de desarrollo a través del análisis de redes colaborativas? Partiendo de un área determinada de conocimientos científicos, el Análisis de redes colaborativas, y el propósito general de desarrollar un método para determinar comunidades de desarrolladores y actores más influyentes en repositorios de sistemas operativos libres, con el objetivo de fortalecer la colaboración entre equipos de desarrollo a través del análisis de redes colaborativas, el presente trabajo se concentró en la aplicación de métodos de detección de comunidades y actores más influyentes en repositorios de sistemas operativos libres.

[1]Centro de Soluciones Libres de la Universidad de las Ciencias Informáticas

Materiales y métodos o Metodología computacional

Para analizar la colaboración de desarrolladores, hemos utilizado el concepto de análisis de redes sociales. Esto se hace estudiando redes que representan desarrolladores y su colaboración. Realizar un análisis de redes sociales significa aplicar diferentes métricas a las redes, donde cada medida mire una propiedad específica de las redes. Ejemplos de métricas son la cantidad de desarrolladores con los que un desarrollador específico ha colaborado y las medidas de centralidad que muestran la influencia que los desarrolladores tienen entre sí.

Los actores más influyentes, son desarrolladores que destacan a partir de las relaciones que presentan con respecto a otros dentro de la red colaborativa. El término de red colaborativa es equivalente a redes sociales y se utiliza de esta forma en la investigación para evitar ser confundido con redes como Facebook, Twitter, entre otras.

Una red colaborativa, se define como una estructura social compuesta por un conjunto de actores (como individuos u organizaciones) y las relaciones entre ellos. La perspectiva de red colaborativa provee un grupo de métodos para analizar la estructura de entidades sociales completas, así como una variedad de teorías explicando los patrones observados en estas estructuras (Wasserman and Faust, 1994; Kazienko, 2012; Carolan, 2013). También puede verse como un conjunto de actores vinculados entre sí. Para modelar y analizar redes colaborativas se utilizan los grafos. En este contexto los vértices representan a los actores (autores, desarrolladores, etc) y las aristas representan las relacionen existentes entre estos.

En una red colaborativa los actores más influyentes pueden identificarse a través de las medidas de centralidad (Farooq et˜al., 2018).Según Lozano et˜al. (2016) las medidas de centralidad son métricas fundamentales para el análisis de redes. Miden cómo de central e importante es un nodo dentro de la red. En la investigación se trabajaron las medidas de centralidad de grado, cercanía e intermediación. La centralidad de grado asume que los nodos más importantes son los que tienen muchas conexiones con otros nodos(Freeman, 1978). Según Kuz et˜al. (2016) los investigadores de redes sociales miden la actividad en la red usando el concepto de centralidad de grado, es decir el número de conexiones directas que tiene un nodo. Consiste en nodos que, independientemente de la cantidad de conexiones, sus aristas permiten llegar a todos los nodos de la red más rápidamente que desde cualquier otro nodo.

La cercanía se basa en la medida de proximidad y en su opuesta, la lejanía (Sabidussi, 1966). Describe mejor la centralidad general, ya que los actores (nodos) son valorados por su distancia, medida en pasos hacia los demás actores de la red. Un actor tiene gran centralidad cuanto menor sea el número de pasos que a través de la red debe realizar para relacionarse con el resto (Pérez˜Beltrán et˜al., 2015; Kuz et˜al., 2016). La centralidad de cercanía asume que los nodos importantes son aquellos que están a una corta distancia del resto de los nodos de la red.

Por último se trabajó con la centralidad de intermediación que mide la importancia de un elemento de un grafo, ya sea un nodo o una arista, por la fracción de los caminos más cortos que pasan a través de él (Freeman, 1977, 1978). La centralidad de intermediación es una medida de centralidad muy popular que, informalmente, define la importancia de un nodo o borde en la red como proporcional a la fracción de rutas más cortas en la red que pasan por él (Riondato and Upfal, 2018). La centralidad de intermediación asume que los nodos importantes en la red son aquellos que conectan otros nodos.

Para el desarrollo del método propuesto, se decidió usar Python 3.5 para realizar la implementación del método, ya que sobresale en la manipulación de datos y la programación de red por la cantidad de librerías que contiene y que facilitan el trabajo de los desarrolladores (Summerfield, 2010; Beazley and Jones, 2013; González˜Duque, 2014; Tale, 2017). Para la gestión, instalación y configuración de Python y sus módulos, fue utilizada la plataforma Anaconda (Analytics, 2015). Para la extracción y creación de la redes se utiliza el módulo Networkx, el cual brinda una serie de funcionalidades que facilita el proceso de creación de redes (Hagberg et˜al., 2008, 2016).

Para la representación de la red obtenida se utilizó Gephi en su versión 0.9.2 por ser ideal para visualizaciones básicas de la red por la alta calidad que presentan sus resultados (Khokhar, 2015; Cherven, 2015; Heymann, 2017). Para el análisis de redes se utilizó la teoría de grafos ya que los grafos son estructuras que constan de dos partes, el conjunto de vértices o nodos y el conjunto de aristas o bordes, que pueden ser orientados o no. Por lo tanto, también es conocida dicha teoría como análisis de redes (Trudeau, 2013; Aguirre, 2014; Peyró˜Outeiriño, 2015; Erçetin and Neyişci, 2016; O'Malley and Onnela, 2017). Además, para realizar las pruebas unitarias automáticamente en el lenguaje de programación Python se utilizó el módulo"unittest".

La detección de comunidades en la red colaborativa se realizó empleando un algoritmo para este fin, específicamente el algoritmo que implementado en Gephi en su versión 0.9.2. Algunos algoritmos existentes para este propósito son Kernighan-Lin (KL), algoritmos aglomerativos/divisivos, algoritmos espectrales y partición de Grafos Multi-nivel (Cuvelier and Aufaure, 2011; Rao and Mitra, 2014; Rao and Mishra, 2016; Thorat, 2017). La detección de comunidades tiene por objetivo agrupar vértices de conformidad con las relaciones entre ellos para formar subgrafos fuertemente vinculados de todo el grafo. Dado que las redes generalmente se modelan como grafos, la detección de comunidades en redes múltiples es también conocida como el problema de partición de grafo en la teoría de grafos moderna (Wang et˜al., 2015).

Resultados y discusión

Para el cumplimento del objetivo general del presente trabajo se deben detectar las comunidades de desarrolladores y actores más influyentes en una red colaborativa donde se representan los paquetes del repositorio

de los sistemas operativos libres y su relación con los desarrolladores que intervienen en cada uno de ellos. El método propuesto sigue el procedimiento que se describe a continuación:

1. Se extraen los ficheros de control de cambios o changelogs de los paquetes que se encuentran en el repositorio.
2. Se extraen los datos útiles a partir de los ficheros de control de cambios obtenidos del paso anterior, con un algoritmo de extracción de datos que propone el autor del presente trabajo, para seleccionar los nombres de los paquetes y los desarrolladores que intervienen en cada uno.
3. Se crea la red colaborativa a partir de los datos obtenidos.
4. Se detectan las comunidades de desarrolladores y actores más influyentes en la red.

Un fichero de control de cambios o changelog[2] es un archivo en el que se encuentra la información en orden cronológico sobre los cambios realizados en cualquier proyecto de tipo informático. El objetivo del changelog es mostrarle a usuarios y desarrolladores los cambios que sean implementados en las diferentes versiones del producto, así como la información acerca de quien realizó dichos cambios para poder contactarlo si se detectan errores o fallas de seguridad. Los ficheros tienen similar estructura lo cual permite que la salida sea siempre similar independientemente de la cantidad de entradas que se procesen.

Los ficheros de control de cambios se pueden obtener descomprimiendo los paquetes en formato ".deb" que se encuentran en el repositorio de las distribuciones GNU/Linux basadas en Debian. Para acceder a los repositorios es necesario conocer las direcciones web de los mismos. Las empresas encargadas de desarrollar distribuciones GNU/Linux tienen públicas estas direcciones, al tratarse de software libre pueden existir también repositorios locales propios de organizaciones o instituciones con intereses específicos.

En el desarrollo de la propuesta de solución se trabajó primeramente con el módulo "os" cuyas funciones permiten el trabajo con directorios de carpetas y archivos. En el presente trabajo se empleó para crear directorios de carpetas con los que trabaja el método para descargar paquetes del repositorio, copiar los changelogs y generar los ficheros o archivos con las redes colaborativas. Para el trabajo con las URL se utilizaron varias librerías como "urllib.request" para acceder a estas y descargar los paquetes hacia directorios previamente creados de forma automática. Otra librería que se empleó para trabajar con las URL fue "urllib3" para obtener direcciones con las que pudiera tratar la librería "BeautifulSoup", esta última permitió obtener de una

[2]https://es.wikipedia.org/wiki/Changelog

página web todos los enlaces que la componen y de esta forma iterar recursivamente por todo el repositorio de paquetes y lograr la descarga de cada archivo.

También se emplearon módulos como "patoollib" para desempaquetar o extraer archivos ".deb" y dar paso a la utilización de las librerías "shutil" y "gzip" para la descompresión de archivos. Los archivos descomprimidos fueron el archivo "data.tar.xz" y "changelog.Debian.gz" con el objetivo de acceder el fichero de control de cambios de cada paquete, de igual forma se eliminaron los paquetes después de ser extraídos para ahorrar espacio en el disco duro de la computadora donde se ejecute el método.

Para la extracción de los changelogs se propone el pseudocódigo siguiente:

Entrada del algoritmo: Dirección URL de un repositorio de sistema operativo de software libre.
Salida del algoritmo: Directorio con los ficheros de control de cambios de los paquetes del repositorio.

```
1: Si URL == paquete
2:   Llamar método ddcr
3: Fin
4: Sino
5:   Obtener direcciones URL
6:   Para direcciones URL hacer
7:     Si dirección url == "../"
8:       Volver a ejecutar el método
9:     Fin
10:   Fin
11: Fin
```

El método **ddcr** es el encargado de descargar los paquetes del repositorio y para cada uno de estos realizar el proceso de desempaquetado, descompresión de los ficheros comprimidos y extracción del changelogs. Este método además copia los changelogs hacia un directorio que sirve como entrada para el proceso de extracción de datos útiles.

A partir de los ficheros de control de cambios obtenidos de una dirección dentro del sistema operativo como "D:\prueba" (si se trabaja sobre cualquier distribución de Windows) se extrajeron los datos útiles, para posteriormente adicionarlos a una lista. En la presente investigación se definen como datos útiles:

- nombres de los paquetes. Ejemplo:"firefox (46.0+build5-0ubuntu0.14.04.2nova1)".

- nombres de los desarrolladores de los paquetes. Ejemplo:"Juan Manuel Fuentes Rodríguez","Chris Coulson","Alexander Sack".

Python permite cargar archivos de texto usando para ello la función "open" a la que se le pasan como parámetros el archivo o ruta del mismo y el modo en que se desea cargar el archivo (lectura, escritura, etc), la carga de los archivos es necesaria para poder realizar operaciones sobre el texto. Una vez cargado el texto, se procede a extraer los datos útiles, para esto es necesario trabajar con expresiones regulares y puede emplearse el módulo "Re" el cual contiene varias funciones para simplificar el código en este sentido.

En la solución se realizó primeramente un filtro de las oraciones que contengan algún correo electrónico teniendo en cuenta que estos tienen como singularidad la existencia del carácter "@". La estructura definida de los changelogs facilita el uso de expresiones regulares aprovechando que los nombres de desarrolladores se encuentran en oraciones donde aparece al menos un correo electrónico. Posteriormente se definió una expresión regular para identificar los datos útiles, para los nombres de desarrolladores la expresión es"[A-Z][a-z]+ [A-Z][a-z]". Para identificar los nombres de paquetes también se definió una expresión regular pero no es necesario filtrar el texto ya que este dato se encuentra al inicio de cada archivo, en la solución propuesta se utiliza la expresión"\A\w+".

Se propone el siguiente pseudocódigo para la extracción de datos útiles:

Entrada del algoritmo: Directorio de changelogs.
Salida del algoritmo: Lista con nombres de paquetes y desarrolladores de cada paquete.
Aclaraciones: Se debe indicar la dirección donde se encuentran los ficheros de control de cambios al algoritmo propuesto.

```
1: Para cada fichero hacer
2:   Para cada oración hacer
3:     Si carácter == "@"
4:       Si expresión regular desarrollador == palabras
5:         Agregar palabras a la lista de desarrolladores de un paquete
6:       Fin
7:     Fin
8:   Agregar lista de desarrolladores de un paquete a lista de todos los desarrolladores
9:   Fin
10:   Si expresión regular paquet e== palabras
```

```
11:      Agregar palabras a la lista de paquetes
12:    Fin el nombre del paquete
13: Fin
14: Para longitud de lista de todos los desarrolladores hacer
15:   Agregar lista de todos los desarrolladores + lista de paquetes a lista general
16: Fin
```

Para crear la red colaborativa a partir de la información obtenida en el proceso de extracción de datos útiles descrito anteriormente. Los datos útiles procedentes del proceso anterior se obtienen como una lista de listas donde el primer elemento de cada lista es el nombre del paquete y los restantes elementos son el nombre de los desarrolladores asociados a este.

Para el trabajo con grafos en Python se hace uso de la librería "NetworkX" con el objetivo de simplificar el trabajo y garantizar una buena calidad en la implementación. Esta librería permitió diseñar, crear, probar y exportar la red colaborativa a través del trabajo con un conjunto de funciones implementadas en "NetworkX". Se crearon dos redes, en la primera los nodos o vértices fueron cada nombre de paquete y desarrollador, donde las aristas que conectan dichos nodos establecen la relación entre estos según se obtienen los datos en la lista de entrada. La segunda red establece como nodos solamente a los desarrolladores siendo los paquetes asociados a estos los datos que permiten establecer las relaciones en la red mediante aristas. La librería en cuestión exporta las redes colaborativas en varios formatos como por ejemplo "gml" creándose de esta forma dos archivos de este tipo para poder ser posteriormente trabajados con Gephi.

Para crear la red colaborativa el programador puede auxiliarse del siguiente pseudocódigo propuesto por el autor de la investigación:

Entrada del algoritmo: Lista de nombres de paquetes y desarrolladores.
Salida del algoritmo: Fichero ".gml" con la red colaborativa.
Aclaraciones: Se debe tener en cuenta que la lista con los nombres de paquetes y desarrolladores está formada por sublistas.

```
1: Para lista de todos los desarrolladores hacer
2:   Agregar desarrolladores a la lista de nodos de la bipartición 0
3: Fin
4: Para lista de paquetes hacer
5:   Agregar paquetes a la lista de nodos de la bipartición 1
```

Revista Cubana de Ciencias Informáticas
Vol. 13, No. 1, Enero-Marzo, 2019
ISSN: 2227-1899 | RNPS: 2301
Pág. 45-60
http://rcci.uci.cu

```
6: Fin
7: Para A en lista de todos los desarrolladores hacer
8:   Para B en A hacer
9:     Agregar la relación de paquetes a desarrolladores a la lista de aristas
10:   Fin
11: Fin
12: Crear archivo "gml" para la red paquete-desarrollador
13: Para longitud de lista de todos los desarrolladores hacer
14:   Agregar desarrolladores a la lista de nodos con el atributo paquete
15: Fin
16: Para A en lista de todos los desarrolladores hacer
17:   Para B en longitud de A hacer
18:     Para C en A hacer
19:       Agregar la relación entre desarrolladores a la lista de aristas
20:     Fin
21:   Fin
22: Fin
23: Crear archivo "gml" para la red de desarrolladores
```

Una vez que se tiene la red colaborativa (fichero gml) a partir de los datos extraídos del repositorios analizado, se procede a su análisis y visualización. La figura 1 muestra un esquema de los pasos a seguir para la obtención de las comunidades y actores más influyentes.

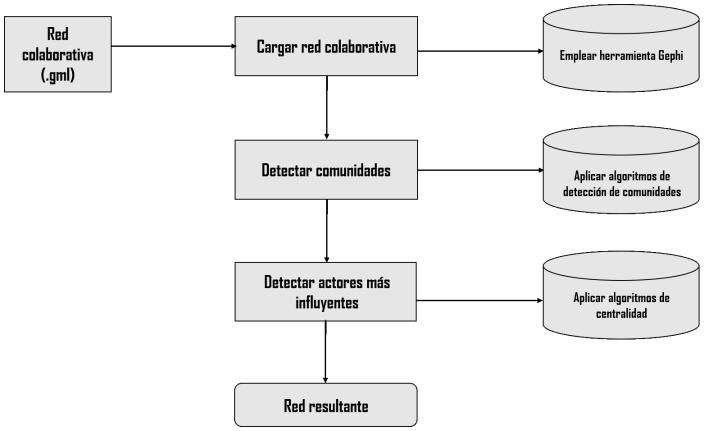

Figura 1: Esquema para la detección de comunidades y actores más influyentes en la red colaborativa (Fuente: Elaboración propia)

El análisis de la red colaborativa se realizó con la herramienta profesional para la visualización de redes Gephi. El análisis con la herramienta antes mencionada permite observar con mayor detalle las relaciones que se presentan en la red colaborativa, moverse por los nodos o vértices con mayor facilidad, así como ajustar la forma en la que se presentan estos sin dañar las relaciones. Gephi permite la detección de comunidades y actores más influyentes a partir de las medidas y algoritmos previamente mencionadas.

La identificación de los desarrolladores por cada paquete dentro de la red colaborativa ayuda a una mejor visualización de esta como se muestra en la figura 2:

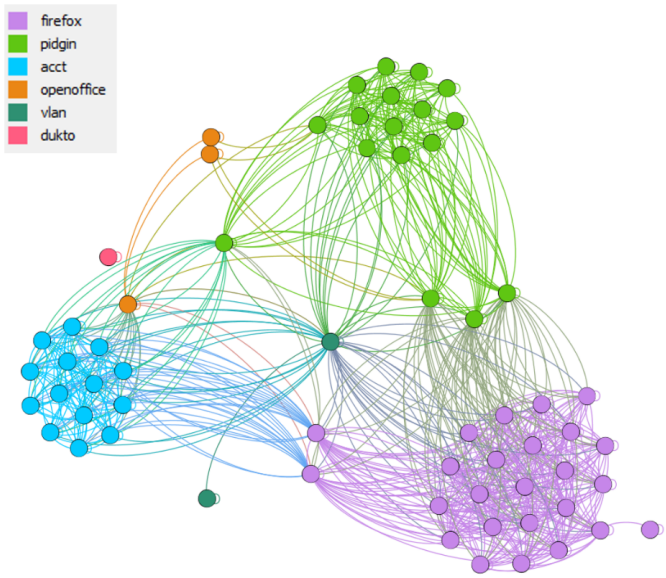

Figura 2: Red colaborativa con los desarrolladores de paquetes identificados (Fuente: Elaboración propia)

De este gráfico podemos tener una idea de la cantidad de personas que colaboran en el desarrollo de los paquetes. De la imagen podemos concluir que la mayor colaboración se encuentra en el desarrollo de Firefox, mientras que solamente dos personas mantienen el paquete vlan. Este hecho nos permite identificar que este paquete puede quedar desatendido si estos dos desarrolladores abandonan el proyecto.

También puede visualizarse la red representándola como un grafo bipartido donde los paquetes del repositorio del sistema operativo libre representan una bipartición del grafo original y los desarrolladores otra, esto permite una mejor vista de la relación paquete-desarrollador. No es más que otra forma de representar la vista anterior, lo que en esta, tenemos además del nombre de los paquetes, el nombre de los desarrolladores. La figura 3 representa los paquetes del repositorio con el color verde (0) y los desarrolladores con el color rosa (1).

Revista Cubana de Ciencias Informáticas
Vol. 13, No. 1, Enero-Marzo, 2019
ISSN: 2227-1899 | RNPS: 2301
Pág. 45-60
http://rcci.uci.cu

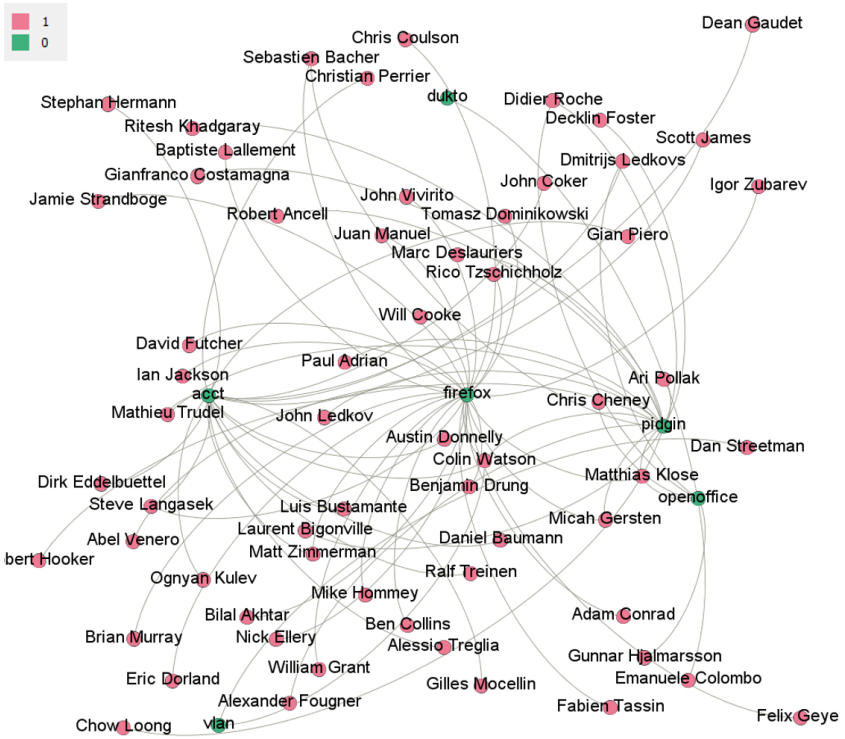

Figura 3: Grafo bipartido de nombres de paquetes y desarrolladores (Fuente: Elaboración propia)

Posteriormente podemos pasar a aplicar algunas de las medidas de centralidad seleccionadas, para identificar cual o cuales son los desarrolladores más influyentes o de mayor impacto en la comunidad.

Basándose en la medida de centralidad se pudieron identificar los actores más influyentes en la red. La figura 4 muestra cómo quedaría la red al aplicarle la medida de intermediación que establece que los nodos más importantes son aquellos que conectan otros nodos.

Luego de aplicarle todos los criterios expuestos a la red, esta quedaría finalmente como se muestra en la figura 5:

En esta ultima imagen, podemos ver como Colin Watson es una persona altamente importante e influyente dentro de la red formada, ya que conecta a varios desarrolladores de diferentes paquetes. De este grafo, podemos identificar a los principales desarrolladores de cada uno de los paquetes analizados, por lo que serían las personas a contactar por los usuarios que quieran colaborar en el desarrollo o mantenimiento de algunos de estos paquetes.

Revista Cubana de Ciencias Informáticas
Vol. 13, No. 1, Enero-Marzo, 2019
ISSN: 2227-1899 | RNPS: 2301
Pág. 45-60
http://rcci.uci.cu

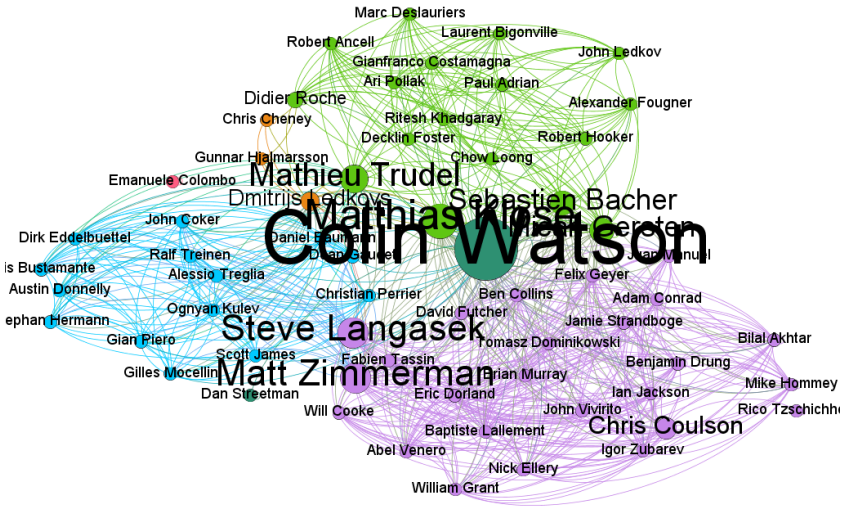

Figura 4: Desarrolladores más importantes de la red colaborativa (Fuente: Elaboración propia)

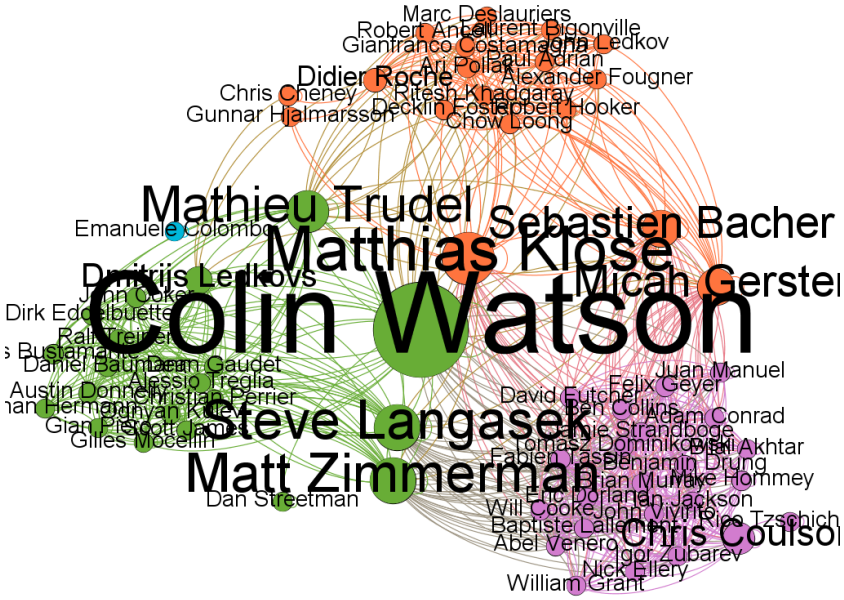

Figura 5: Red colaborativa con todos los criterios aplicados (Fuente: Elaboración propia)

De forma general, se pueden extraer algunas conclusiones de los datos recopilados. Cuantos más desarrolladores tenga un proyecto, mayor será la centralidad promedio. La centralidad promedio sigue una línea muy de cerca, lo que sugiere que las redes tienen una estructura similar. La mayoría de los desarrolladores tienen un pequeño índice de centralidad, mientras que solo unos pocos desarrolladores ocupan un lugar central en los proyectos.

Estos son generalmente los llamados “community manager” y/o lideres de proyectos.

Conclusiones

Al concluir la presente investigación se pueden arribar a las siguientes conclusiones que dan solución al objetivo general de la misma:

1. La detección de comunidades permitió determinar grupos que se establecen en la red a partir de las relaciones entre desarrolladores y las medidas de centralidad permitieron identificar los actores más influyentes dentro de la red.

2. Se obtuvo un método implementado en Python que facilitó: la búsqueda de paquetes en repositorios de sistemas operativos libres, la extracción de los ficheros de control de cambios de cada uno de estos, la extracción de los nombres de paquetes y sus desarrolladores, así como la creación de una red colaborativa a partir de la relación entre desarrolladores y otra red con la relación paquete ? desarrollador.

3. A partir del trabajo con la herramienta Gephi se pudieron visualizar las redes colaborativas y proceder a la detección de comunidades y actores más influyentes en la red colaborativa permitiendo fortalecer la colaboración entre equipos de desarrollo.

   La validación realizada a través de pruebas unitarias y de aceptación permitieron corregir las no conformidades detectadas, probando la calidad de la solución propuesta.
4.

En un futuro, se pretende optimizar los métodos propuestos para su funcionamiento y ejecución en paralelo. Además estudiar la factibilidad de usar otras técnicas de extracción de datos propias del procesamiento de lenguaje natural, como el reconocimiento de entidades, para refinar el proceso de identificación de desarrolladores.

Referencias

Aguirre, J. L. (2014). Actores, relaciones y estructuras: introducción al análisis de redes sociales. *Hologramática*, 20(2):161–187.

Analytics, C. (2015). Anaconda scientific python distribution.

Beazley, D. and Jones, B. K. (2013). *Python Cookbook: Recipes for Mastering Python 3*. .ºReilly Media, Inc.”.

Carolan, B. V. (2013). *Social network analysis and education: Theory, methods & applications*. Sage Publications.

Cherven, K. (2015). *Mastering Gephi network visualization*. Packt Publishing Ltd.

Cuvelier, E. and Aufaure, M.-A. (2011). Graph mining and communities detection. In *European Business Intelligence Summer School*, pages 117–138. Springer.

Erçetin, Ş. Ş. and Neyişci, N. B. (2016). Social network analysis: A brief introduction to the theory. In *Chaos, Complexity and Leadership 2014*, pages 167–171. Springer.

Farooq, A., Joyia, G. J., Uzair, M., and Akram, U. (2018). Detection of influential nodes using social networks analysis based on network metrics. In *Computing, Mathematics and Engineering Technologies (iCoMET), 2018 International Conference on*, pages 1–6. IEEE.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

González Duque, R. (2014). Python para todos. *Creative Commons Reconocimiento*, 2.

Hagberg, A., Schult, D., and Swart, P. (2016). Networkx framework.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Heymann, S. (2017). Gephi. *Encyclopedia of social network analysis and mining*, pages 1–14.

Kazienko, P. (2012). Social network analysis: Selected methods and applications. In *DATESO*, page 151. Citeseer.

Khokhar, D. (2015). *Gephi cookbook*. Packt Publishing Ltd.

Kuz, A., Falco, M., and Giandini, R. (2016). Análisis de redes sociales: un caso práctico. *Computación y Sistemas*, 20(1):89–106.

Lozano, M., García-Martínez, C., Rodríguez, F. J., and Trujillo Mendoza, H. M. (2016). Optimización de ataques a redes complejas mediante un algoritmo de colonias de abejas artificiales. In *Actas de la XVII Conferencia de la Asociación Española Para la Inteligencia Artificial*, page 151. Ediciones Universidad de Salamanca.

O'Malley, A. J. and Onnela, J.-P. (2017). Introduction to social network analysis. *Methods in Health Services Research*, pages 1–44.

Pérez Beltrán, J. E., Valerio Ureña, G., and Rodríguez-Aceves, L. (2015). Análisis de redes sociales para el estudio de la producción intelectual en grupos de investigación. *Perfiles educativos*, 37(150):124–142.

Peyró Outeiriño, M. B. (2015). Conectados por redes sociales: Introducción al análisis de redes sociales y casos prácticos. *Redes. Revista Hispana para el Análisis de Redes Sociales*, 26(2):236–241.

Rao, B. and Mishra, S. (2016). Detection of influential communities and members in a community graph of villages using graph mining techniques. *International Journal of Computer Science and Information Security*, 14(5):85.

Rao, B. and Mitra, A. (2014). A new approach for detection of common communities in a social network using graph mining techniques. In *High Performance Computing and Applications (ICHPCA), 2014 International Conference on*, pages 1–6. IEEE.

Riondato, M. and Upfal, E. (2018). Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):61.

Sabidussi, G. (1966). The centrality index of a graph. *Psychometrika*, 31(4):581–603.

Summerfield, M. (2010). *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional.

Tale, S. (2017). Python 3: The ultimate beginners guide for python 3 programming.

Thorat, A. R. (2017). Detection of influential communities and members in a community using graph mining techniques. *Imperial Journal of Interdisciplinary Research*, 3(3).

Trudeau, R. J. (2013). *Introduction to graph theory*. Courier Corporation.

Wang, M., Wang, C., Yu, J. X., and Zhang, J. (2015). Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment*, 8(10):998–1009.

Wasserman, S. and Faust, K. (1994). *Social network analysis: Methods and applications*, volume 8. Cambridge university press.