Facultad 2

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

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

AUTOR

Jorge Alejandro Román Donates

TUTORES

Ing. Vladimir Milián Núñez

Lic. Raynel Batista Tellez

CO-TUTORA

M. Sc. Eliana Bárbara Ril Valentin

La Habana, 2018

"Año 60 de la Revolución"

"La información es la gasolina del siglo XXI, y la analítica de datos el motor de combustión."

Peter Sondergaard

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Método de influencia social dentro de una red de desarrollo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Vladimir Milián Núñez

_____

Firma del Tutor

Raynel Batista Tellez

_____

Firma del Tutor

Eliana Bárbara Ril Valentin

_____

Firma del Cotutor

Jorge Alejandro Román Donates

_____

Firma del Autor

DATOS DE CONTACTO

Tutor: Ing. Vladimir Milián Núñez, Ingeniero en Ciencias Informáticas desde el 2008. Se desempeña como profesor de Física en la facultad 2 de la Universidad de las Ciencias Informáticas. Posee cursos como Aprendizaje Automático no estándar y Minería de Datos para Bioinformática. Es miembro de la Asociación Cubana para el Reconocimiento de Patrones, así como de la Sociedad Cubana de Matemáticas y Computación, y de la Unión de Informáticos de Cuba. Sus principales resultados se enmarcan en el aprendizaje automático, desarrollo web, minería de datos, minería de texto, extracción de información y aprendizaje estadístico.

Correo electrónico: vmilian@uci.cu Researchgate: Vladimir Milián Núñez Twitter: @vmiliann

Tutor: Lic. Raynel Batista Tellez, Sociólogo, Coordinador General de Ediciones Futuro, sello editorial de la Universidad de las Ciencias Informáticas. Ha impulsado proyectos interdisciplinares que vinculan las ciencias sociales y el cálculo computacional, así como las humanidades digitales, en colaboración con actores de América Latina, Europa y Asia. Ha estado vinculado desde su fundación al Grupo de Estudios Sociales en Ciencia y Tecnología (GECYT), al Grupo de Minería de Proceso y el Grupo de Web Semántica. Dirige un proyecto nacional CITMA del Programa Priorizado en Ciencia, Tecnología e Innovación como parte de un proyecto doctoral. Coordina la Revista Cubana de Ciencias Informáticas, indizada en Scielo, DOAJ, Redalyc. Sus principales resultados se enmarcan en la sociología de la innovación y la antropología digital, con especial interés en la curación de contenidos digitales.

Correo electrónico: rainer@uci.cu Researchgate: Raynel Batista Twitter: @RBatell

Co-tutor: M. Sc. Eliana Bárbara Ril Valentin, Ingeniera en Ciencias Informática desde el 2008 y Máster en Gestión de Proyectos Informáticos desde el 2012. Profesora de la facultad 2 de la Universidad de las Ciencias Informáticas, ostenta la categoría docente de profesor asistente. Se desempeña además como profesora principal del 3er año de la facultad 2. Se ha desarrollado como profesora de asignaturas como Metodología de la Investigación Científica e Ingeniería de Software. Sus principales resultados se enmarcan en el campo de la Gestión de Proyectos Informáticos y la Minería de datos.

Correo electrónico: ebril@uci.cu Researchgate: Eliana Bárbara Ril Valentin

Dedicatoria

En primer lugar, quiero dedicarle este resultado a mi madre por su infinito cariño, por ser cómplice en muchas de mis locuras, por sufrir mis reveses y gozar mis logros como si fueran suyos.

A mi padre que, aunque la vida nos ha privado de la oportunidad de estar juntos desde hace muchos años sé que se sentiría orgulloso hoy.

A mi familia de San José por su preocupación, cariño y apoyo incondicional especialmente a mi abuela Ñaña y a Viki por ser además de un segundo padre, un buen amigo y consejero.

A mi familia de Güines, en especial a Elvita, Daniela y Aramis por adoptarme como un miembro más de la familia, por su apoyo y preocupación en todo momento.

A mi novia por no permitirme desviarme del trabajo en el transcurso de esta investigación, por sus regaños y sus consejos.

A mis tíos Julio y Lourdes por confiar en mí, por cada palabra que hemos intercambiado y que me ha servido de impulso para seguir adelante y ser mejor.

Agradecimientos

Agradezco en primer lugar a mis tres tutores por guiarme en esta investigación, por sus enseñanzas, por su apoyo, por el tiempo invertido, por su dedicación y por enseñarme a confiar más en mis capacidades.

A Dairelys por permitirme contribuir a su tesis doctoral e introducirme en el mundo del análisis de redes sociales.

A los profesores que he tenido en el transcurso de la carrera por permitirme aprender de ellos, especialmente a Abel por servirle de meta al profesional que quiero ser.

A los profesores que contribuyeron con sus críticas, aportes y consideraciones a la culminación de esta investigación, especialmente al doctor José Medina, a los ingenieros Javier Ponce, Osmar Capote y Antonio Hernández.

A mis amigos especiales Félix y José con los que he compartido momentos claves de mi vida, gracias por servirme de ejemplo e impulso para seguir adelante cuando las cosas se han tornado oscuras.

A mis compañeros y amigos Alexis y Oswald con los que he reído y llorado a lo largo de estos cinco años, no existen palabras para agradecer la amistad y todo lo que esto conlleva.

A Bienvenido y Yoan Bernardo a quienes a pesar de no conocer desde que mi llegaba a la UCI, hoy considero mis amigos y de los cuales he aprendido sabías lecciones para la vida.

Por último, agradecer a mis compañeros y amigos de la universidad por todo lo que hemos aprendido los unos de los otros, por las experiencias vividas que nos ayudaron a crecer, especialmente a Yoan Carlos, Victor Alexis, Lenzano, Raidel B, Raydel, Yunior, Glenda, Osniel, Wilber, Pedro, Yosniel, Yanlee, Suan y Danny.

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 clave: 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. 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.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC's) son una parte imprescindible del mundo actual. El término TIC's engloba todo tipo de dispositivos y aplicaciones comunicativas, incluidos la radio, la televisión, los teléfonos móviles, equipos y programas informáticos y de redes, sistemas de satélites, etc., al igual que diferentes servicios y aplicaciones relacionadas con ellos (Bos et al. 2011). Muchos de estos dispositivos incluyen al menos un sistema operativo lo cual facilita el manejo de estos productos por parte de los usuarios. Un sistema operativo "*es el software que actúa como intermediario entre el usuario de una computadora y el hardware, permitiendo al usuario ejecutar programas de manera práctica y eficiente*" (Silberschatz, Galvin y Greg 2006).

Cuba intenta paulatinamente disminuir la brecha tecnológica existente con respecto a países del primer mundo y lograr la mayor conectividad de sus ciudadanos, razón por la que se ejecuta el proceso de informatización de la sociedad cubana. Tal es la importancia de este proceso que en el artículo 108 de la actualización de los lineamientos de la política económica y social del partido en el período 2016-2017 se aborda de la siguiente forma (PCC 2016):

"Avanzar gradualmente, según lo permitan las posibilidades económicas, en el proceso de informatización de la sociedad, el desarrollo de la infraestructura de telecomunicaciones y la industria de aplicaciones y servicios informáticos. Sustentar este avance en un sistema de ciberseguridad que proteja nuestra soberanía tecnológica y asegure el enfrentamiento al uso ilegal de las tecnologías de la información y la comunicación. Instrumentar mecanismos de colaboración internacional en este campo".

En este sentido en el 2004 la Asamblea Nacional acordó que para la informatización de la sociedad cubana era imprescindible emplear plataformas libres. Además, se acordó el desarrollo de un sistema operativo cubano tomando como base una distribución de GNU/Linux adaptándola a los objetivos y necesidades del país (Pierra Fuentes 2011).

Por la importancia que se le atribuía a dicha tarea, el desarrollo del nuevo sistema operativo se le encargó a la Universidad de las Ciencias Informáticas (UCI); es así como la distribución cubana de GNU/Linux Nova comenzó su desarrollo, orientada inicialmente al escritorio. En sus inicios Nova fue un experimento académico para vincular el proceso docente y productivo de un grupo de estudiantes que lo proponían como base tecnológica para los incipientes desarrollos de software libre (Pierra Fuentes 2011).

Al ser Nova una distribución de software libre comparte sus características, una característica esencial de este tipo de sistema es la existencia de cuatro libertades para los usuarios, según (Stallman 2004) estas son:

- Libertad 0: la libertad para ejecutar el programa sea cual sea el propósito.
- Libertad 1: la libertad para estudiar el funcionamiento del programa y adaptarlo a sus necesidades, el acceso al código fuente es condición indispensable para esto.
- Libertad 2: la libertad para redistribuir copias y ayudar así a otros usuarios.
- Libertad 3: la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad, el acceso al código fuente es también condición indispensable para esto.

Dichas libertades permiten el trabajo en comunidades como una ventaja imprescindible de los sistemas operativos libres. En el mundo del software libre, estas comunidades consisten en grupos de usuarios o desarrolladores (experimentados o noveles) que contribuyen a la mejora del sistema operativo. Debido a que el código de las distribuciones es abierto y el uso de internet ha permitido traspasar fronteras, las comunidades crecieron masivamente a través de los años, eliminando las fronteras entre sus miembros. Algunas de las formas de contribuir con el trabajo de las comunidades es realizar mejoras o mantenimiento en el código fuente del sistema y/o componentes, implementar nuevas aplicaciones y funcionalidades, o contribuir con la escritura y traducción de documentación (manuales, ayudas, guías, artículos, etc…).

La contribución de las comunidades puede verse de forma práctica en los repositorios de paquetes (donde también se encuentra el código fuente de estos) que puede utilizar el sistema, al permitir que los usuarios puedan instalarlos en sus computadoras sin necesidad de tener el archivo de instalación almacenado en el disco duro. En GNU/Linux, un repositorio no es más que un conjunto o recopilación de programas preparados para su instalación en una distribución determinada o sus derivados. Suelen ser archivos binarios pre-compilados, aunque también existe la posibilidad de descargarse el código fuente (Tapia Chiroldes 2014).

Un repositorio de software libre puede contener miles de paquetes y colaboradores, como es el caso del repositorio del sistema operativo Nova en la UCI, donde un paquete puede tener uno o varios desarrolladores y un desarrollador colaborar en uno o más paquetes. Según el centro CESOL encargado del desarrollo de Nova, dicha distribución alcanza la cifra de 70 000 paquetes al cierre del 2017 ascendiendo a los 129 Gigabytes, 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. Por otra parte, se constató que los repositorios de Ubuntu en la UCI ocupaban 369 Gigabytes en los servidores hospedados en el nodo central al cierre de 2017, lo cual sugirió que la dimensión de las comunidades y el nivel de actividad de sus miembros, ejemplo los desarrolladores de paquetes sea elevada.

Por tanto, el estudio de las interacciones que establecen los desarrolladores de estos paquetes a partir de intereses de desarrollo comunes contribuye a identificar sus comunidades, promover la colaboración entre equipos de desarrollo, ayudar a determinar los desarrollos críticos y actores más influyentes (desarrolladores líderes o expertos). Además, el análisis de la relación entre desarrolladores permite identificar y sugerir comunidades a partir de otras ya existentes, lo cual contribuye a conformar equipos de trabajo para la implementación de nuevos paquetes y mantenimiento de otros.

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. Por lo que, la realización de estos análisis de forma manual pudiera distanciarlos de sus propósitos originales.

Partiendo del contexto anterior se identificó el siguiente **problema**: ¿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?

Se delimita como **objeto de estudio**: Análisis de redes colaborativas. Para dar solución al problema planteado, se define como **objetivo general**: 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, enmarcado en el **campo de acción**: Métodos de detección de comunidades y actores más influyentes en repositorios de sistemas operativos libres.

Para guiar la investigación se definen las siguientes **preguntas científicas**:

1. ¿Cuál es el estado actual del proceso de desarrollo en comunidades de software libre?
2. ¿Cuáles son los principales fundamentos teóricos y metodológicos acerca del análisis de redes colaborativas, la detección de comunidades y actores más influyentes?
3. ¿Qué estructura tendrá el método para determinar comunidades de desarrolladores en repositorios de sistemas operativos libres a través del análisis de redes colaborativas para identificar los actores más influyentes?
4. ¿Qué resultado tendrá la prueba empírica de factibilidad del método implementado?

Las **tareas de investigación** que se deben cumplir son:

- Análisis de los principales conceptos y trabajos relacionados con el análisis de redes colaborativas, específicamente la detección de comunidades y actores más influyentes en redes de colaboración.
- Diseño de un método para determinar comunidades de desarrolladores en repositorios de sistemas operativos libres a través del análisis de redes colaborativas para identificar los actores más influyentes.
- Implementación del método propuesto para determinar comunidades de desarrolladores en repositorios de sistemas operativos libres a través del análisis de redes colaborativas para identificar los actores más influyentes.
- Generación de los datos necesarios para validar la propuesta mediante pruebas de software.

Para el desarrollo del presente trabajo se emplearon varios **métodos científicos**. Los **métodos teóricos** utilizados son:

Análisis-síntesis: Facilitó el procesamiento de la información obtenida en la investigación realizada sobre el análisis de redes colaborativas y la detección de comunidades y actores más influyentes. Se realizó un estudio de los conceptos asociados a las redes colaborativas y grafos, así como de las herramientas a utilizar en el desarrollo de la solución propuesta basándose en la revisión de sitios web, trabajos de diploma y artículos científicos.

Modelación: Permitió la creación del modelo conceptual para entender el contexto en el que se enmarca la investigación.

Los **Métodos empíricos** empleados fueron:

Entrevista: Permitió obtener la información necesaria relacionada con los problemas presentes en el desarrollo de los paquetes del repositorio del sistema operativo Nova.

El presente documento cuenta con una estructura de tres capítulos, los cuales abordan los siguientes temas:

En el Capítulo 1 Fundamentación teórica sobre el análisis de redes colaborativas: Se definen los conceptos más relevantes relacionados con la teoría de grafos, el análisis de redes colaborativas, algoritmos de detección de comunidades y medidas de centralidad, así como el Proceso de Descubrimiento del Conocimiento con las fases que lo componen. Además, se realiza una descripción de las tecnologías a emplear como lenguaje de programación, librerías para el trabajo con direcciones URL, archivos de texto y análisis de redes sociales. Por último, se analizan las principales herramientas para la visualización de redes.

En el Capítulo 2 Método para la detección de comunidades de desarrolladores y actores más influyentes en repositorios de sistemas operativos libres: Se presenta un método para determinar las comunidades de desarrolladores de software libre a partir de su participación en la implementación de paquetes para los repositorios de las distribuciones GNU/Linux.

En el Capítulo 3 Validación del método implementado: se realiza la validación del método desarrollado mediante varios criterios para determinar que la solución propuesta responde satisfactoriamente a los objetivos que se trazaron. Se realizan varias pruebas al código para asegurar la calidad del método propuesto verificando que no existan fallas.

CAPÍTULO 1. Fundamentación teórica sobre el análisis de redes colaborativas

En el presente capítulo se definen los conceptos más relevantes relacionados con la teoría de grafos, el análisis de redes colaborativas, algoritmos de detección de comunidades y medidas de centralidad, así como el Proceso de Descubrimiento del Conocimiento con las fases que lo componen. Además, se realiza una descripción de las tecnologías a emplear como lenguaje de programación, librerías para el trabajo con direcciones URL, archivos de texto y análisis de redes sociales. Por último, se analizan las principales herramientas para la visualización de redes.

1.1 Desarrollo colaborativo en el software libre

Los pilares del software libre comenzaron a instaurarse en los años 1980, en este proceso Richard Stallman tuvo un singular protagonismo. El término software libre no tiene ninguna relación con el precio sino con la libertad, puede considerarse libre un software que cumpla con las cuatro libertades definidas por Stallman. En 1984 se comienza la implementación del software GNU. En 1985 se crea la *Free Software Foundation* como una organización sin ánimos de lucro dedicada al desarrollo de software libre y en 1992 se combina el sistema GNU con un kernel desarrollado por Linus Torvalds resultando en lo que hoy se conoce como GNU/Linux (Stallman 2004).

En el proceso de desarrollo de una distribución de software libre, trabajan en comunidad usuarios y desarrolladores para lograr un producto final. Para hacer esto posible existe un repositorio que almacena el código fuente de todas las aplicaciones disponibles para la distribución. Una computadora que tenga instalada alguna distribución se conecta al repositorio que contiene los paquetes y puede instalar mientras tenga acceso al mismo cualquiera de estos paquetes. Cuando un usuario de una distribución de software libre detecta un error en un programa tiene varias opciones, dos de ellas son: notificar a los desarrolladores o al equipo de soporte del error encontrado, o intentar solucionar el problema por sí mismo. Para intentar solucionar el problema por sí solo es indispensable contar con conocimientos de programación.

Todos los paquetes del repositorio cuentan con uno o más desarrolladores. Existe el autor del paquete y un conjunto de desarrolladores que han implementado cambios en el. Cuando un desarrollador crea un paquete o realiza cambios en el código, este debe plasmar sus datos para que otros desarrolladores puedan contactarlo en caso de necesitar ayuda con

algún error que se detecte. Los datos de los desarrolladores se almacenan en un fichero que contiene cada paquete nombrado “changelog”. Analizando la información contenida en el “changelog” se puede determinar la relación existente entre cada uno de los desarrolladores que intervienen en cada paquete que incluye el repositorio. La imagen siguiente corresponde a un fragmento de un fichero de control de cambios:

```
1  firefox (53.0.2+build1-0ubuntu0.16.04.3-1nova1) 2017; urgency=medium
2
3    * Change the owner
4
5   -- Abel Venero <avenero@uci.cu>  Tue, 23 May 2017 14:44:38 -0400
6
7  firefox (53.0.2+build1-0ubuntu0.16.04.2) xenial-security; urgency=medium
8
9    * New upstream stable release (53.0.2build1)
10     - see USN-3260-2
11
12  -- Chris Coulson <chris.coulson@canonical.com>  Fri, 05 May 2017 15:29:23 +0100
13
14 firefox (53.0+build6-0ubuntu0.16.04.1) xenial-security; urgency=medium
15
16   * New upstream stable release (53.0build6)
17     - see USN-3260-1
18
19   * Update unity-menubar.patch with the latest version submitted upstream
20   * Build with --disable-rust for now
21   * Stop installing the NPAPI headers and make firefox-dev a transitional
22     package
23     - update debian/build/rules.mk
24     - update debian/rules
25     - update debian/control{,.in}
26     - remove debian/firefox-dev.install.in
27     - remove debian/firefox-dev.links.in
28     - remove debian/pkgconfig/mozilla-plugin.pc.in
29   * Add Urdu language pack
30   * Refresh patches
31     - update debian/patches/revert-upstream-search-engine-changes.patch
32     - update debian/patches/unity-menubar.patch
```

Figura 1. Fragmento de un fichero de control de cambios (Fuente: Elaboración propia)

1.2 Teoría de grafos

La teoría de grafos es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos. 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 (Aguirre 2011; Trudeau 2013).

En (Gross, Yellen y Zhang 2013) se define un grafo como cualquier objeto matemático que involucre puntos y conexiones entre ellos. Si todas las conexiones son unidireccionales, se llama un dígrafo. También se define como un par ordenado, $G = (V, E)$ donde:

- Los elementos de **V** se llaman vértices o nodos.
- Los elementos de **E** se llaman aristas o bordes.
- Cada arista tiene un conjunto de uno o dos vértices asociados, que se llaman puntos finales Se dice que un borde se une a sus puntos finales.

Los grafos pueden clasificarse como ponderados o no ponderados. Un grafo ponderado es aquel al cual se les asigna un valor real positivo a las aristas, que se conoce como peso o costo.

Algunas formas de representar los grafos son a partir de la matriz de adyacencia. En (Gross, Yellen y Zhang 2013) se plantea que una matriz de adyacencia para un grafo simple **G** cuyos vértices están ordenados explícitamente $v_1, v_2, \dots, v_n$ es la matriz $n\ x\ n$ $\mathbf{A_G}$ tal que:

$$A_G(i,j) = \begin{cases} 1\ si\ v_i\ y\ v_j\ son\ adyacentes \\ 0\ en\ otros\ casos \end{cases}$$

En el contexto del análisis de redes colaborativas los vértices representan a los actores (autores, desarrolladores, etc) y las aristas representan las relacionen existentes entre estos.

Aunque existen varios tipos de grafos estos se clasifican fundamentalmente en dirigidos y no dirigidos. Los grafos dirigidos sirven para representar relaciones que pueden ser tanto simétricas como asimétricas, y por su parte los grafos no dirigidos permiten representar solamente relaciones asimétricas.

Otras definiciones importantes dentro de la teoría de grafos son ofrecidas por (Tucker 2004):

Definición 1.2.1: Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos, si $V' \subseteq V, E' \subseteq E$ se dice que **G'** es un subgrafo de **G** y **G** es un super grafo de **G'**.

Definición 1.2.2: Se denomina grado de un vértice **v** a la cantidad de aristas que inciden en **v** y se representa como $\mathrm{g(v)}$.

Definición 1.2.3: Se denomina camino desde el vértice v_i al vértice v_j en un grafo $G = (V, E)$ a la secuencia de vértices $CA = v_1, v_2, \dots, v_n$ si $\exists(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n) \in E$. Además, una arista solo puede aparecer una vez.

Definición 1.2.4: Se denomina longitud de un camino $CA = v_1, v_2, \dots, v_n$, n > 1, a la suma de los costos de todas las aristas presentes en el mismo y se representa como $|CA| = \sum_{h=1}^{n-1} c_{h,h+1}$. En el caso de los grafos no ponderados, la longitud del camino se puede calcular como la cantidad de aristas presentes en el mismo, o sea $|CA| = n - 1$.

La centralidad basada en vértices se define con el fin de medir la importancia de un vértices en la red o grafo. Esta ha llamado mucho la atención como una herramienta para el estudio de las redes sociales. Un vértice con alta centralidad suele ser considerado más altamente influenciable que otros vértices de la red (Fernández Rodríguez y Hidalgo Ruiz 2014) .

A continuación se muestran algunos conceptos definidos por (Aggarwal 2011):

Definición 1.2.5: Sea **A** la matriz de adyacencia de un grafo **G**, y $g(v_i)$ el grado del vértice v_i , entonces el grado de centralidad c_i^{DEG} del vértice v_i se define:

$$c_i^{DEG} = g(v_i)$$

Definición 1.2.6: La medida de cercanía es la más conocida y utilizada de las medidas radiales de longitud. Se basa en calcular el promedio de las distancias más cortas desde un vértice hacia todos los demás. Formalmente se define como:

$$c_i^{CLO} = e_i^T S$$

Donde e_i^T es la transpuesta de un vector columna cuyo i-ésimo elemento es 1 y el resto son 0, **S** es la matriz cuyos elementos en la posición **(i, j)** corresponden a la distancia más corta desde el vértice v_i al vértice v_j.

1.3 Análisis de redes colaborativas y detección de comunidades

El término análisis de redes colaborativas es equivalente a análisis de redes sociales (ARS) y es empleado en la investigación para evitar ser confundido con las plataformas para comunidades de usuarios como Facebook, Twitter, Instagram, Youtube, LinkedIn o Research Gates. El ARS es una metodología novedosa que brinda una colección de métodos, técnicas y herramientas que permiten estudiar las relaciones humanas y predecir comportamientos, en función de inferir redes de interacción social y/o profesional para

apoyar el proceso de toma de decisiones (van der Aalst, Reijers y Song 2005; Nooy, Mrvar y Batagelj 2011; Borgatti y Halgin 2011).

Una red colaborativa se define como una estructura social compuesta por un conjunto de actores sociales (como individuos u organizaciones) y las relaciones diádicas entre estos actores. 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 y Faust 1994). También puede verse como un conjunto de actores vinculados entre sí (Monsalve Moreno 2008).

Algunas definiciones importantes son abordadas por (Wasserman y Faust 1994):

Definición 1.3.1: Se denomina actor a la entidad social que interactúa con otras entidades. Esta puede ser una persona, un grupo de individuos, una organización, una ciudad, una nación, entre otros.

Definición 1.3.2: Un vínculo o conexión corresponde al tipo de relación que conecta un par actores distintos. Estos vínculos pueden ser de naturaleza diferente: relaciones transnacionales, de comunicación instrumental, de poder o autoridad, de parentesco, entre otros.

Definición 1.3.3: Una díada se compone de un par de actores y la posible relación existente entre ellos. Varios estudios parten de la díada como unidad fundamental, analizando para cada par de actores, si el vínculo es recíproco o no.

Definición 1.3.4: Se denomina tríada al conjunto de tres actores y las posibles relaciones existentes entre ellos. Una propiedad interesante de una tríada es la existencia o no, de transitividad entre sus miembros. Es decir, comprobar que, si el actor **i** tiene una relación con el actor **j** y a su vez el actor **j** tiene relación con el actor **k**, entonces **i** también tiene una relación con el actor **k**.

Definición 1.3.5: Se denomina grupo social a un grupo finito de actores con características o intereses similares.

Una red colaborativa es modelada como un grafo $G = (V, E)$. En la presente investigación, el conjunto de vértices **V** se corresponde a los actores y el conjunto de aristas **E** a las relaciones sociales.

Las topologías básicas de red según (Sueur, Deneubourg y Petit 2012) son:

- **Red centralizada:** Todos los nodos, menos uno, son periféricos y sólo pueden comunicarse a través del nodo central. La caída del nodo central priva del flujo a todos los demás nodos.
- **Red descentralizada:** Aparece por interconexión de los nodos centrales de varias redes centralizadas. Como resultado no existe un único nodo central sino un centro colectivo de conectores. La caída de uno de los nodos centralizadores conlleva la desconexión de uno o más nodos del conjunto de la red, mientras que la caída del nodo centralizador produciría necesariamente la ruptura o desaparición de la red.
- **Red distribuida:** Todos los nodos se conectan entre sí sin que tengan que pasar necesariamente por uno o varios centros. Desaparece la división centro/periferia y, por tanto, el poder de filtro sobre la información que fluye por ella. La red es robusta ante la caída de nodos: ningún nodo, al ser extraído, genera la desconexión de otro.

Las comunidades, también llamadas agrupaciones o módulos, son grupos de vértices que probablemente compartan propiedades comunes y / o desempeñan roles similares dentro del grafo (Fortunato 2010).

En los paquetes de las distribuciones GNU/Linux también pueden apreciarse las comunidades ya que los autores y desarrolladores de los mismos se relacionan entre sí a partir de la creación y modificación; la detección de las comunidades puede repercutir grandemente en la toma de decisiones por parte de cualquier institución o individuo. Las distribuciones que tienen como raíz a Debian tienen un fichero en cada paquete llamado “changelog” en el cual se encuentra información sobre cada individuo que ha implementado código en el paquete.

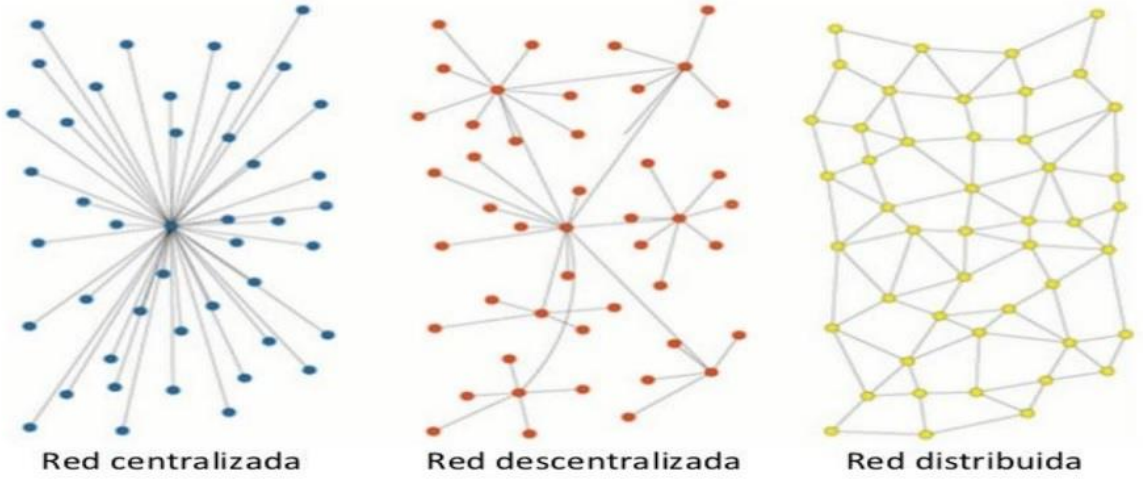

Figura 2. Topologías básicas de las redes (Fuente: Baran 1964)

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. En la última década, muchas soluciones han surgido en la literatura, tratando de resolver este problema desde varias perspectivas (Wang et al. 2015).

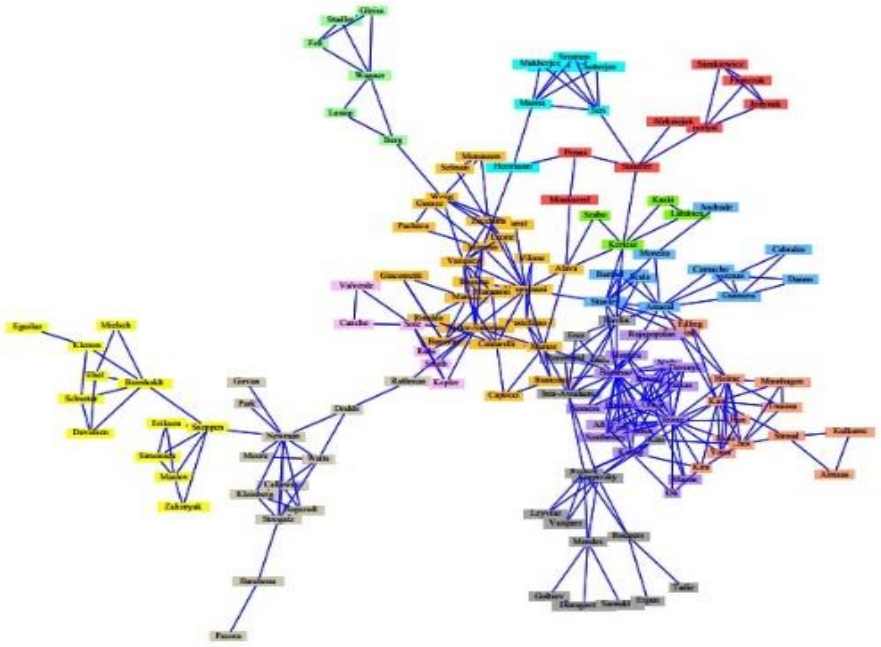

Figura 3. Ejemplo de detección de comunidades (Fuente: Newman y Park 2003)

1.4 Algoritmos de detección de comunidades

Para detectar comunidades en redes o grafos existen un conjunto de algoritmos los cuales se emplean en dependencia de los objetivos que se persigan en la solución de un problema determinado. En (Aggarwal 2011) se encuentran algunos de los algoritmos para la detección de comunidades los cuales se describen a continuación:

1.4.1 Algoritmo Kernighan-Lin (KL)

Es uno de los algoritmos de partición de grafos que optimiza la función objetivo, o sea, minimiza el corte del borde manteniendo equilibrados los tamaños de los grupos. Es iterativo y comienza con una bipartición inicial de la gráfica. En cada iteración, el algoritmo busca un subconjunto de vértices de cada parte del grafo de modo que al intercambiarlos se produzca una reducción en el corte del borde. La ganancia g_v de un vértice v es la reducción en el corte del borde si el vértice v se mueve de su partición actual a la otra partición. El algoritmo KL selecciona repetidamente de la partición más grande el vértice

con la mayor ganancia y lo mueve a la otra partición; no se considera que un vértice se mueva nuevamente si ya se ha movido en la iteración actual. Después de mover un vértice, las ganancias de sus vértices vecinos se actualizarán para recuperar la nueva asignación de vértices a las particiones (Kernighan y Lin 1970).

1.4.2 Algoritmos aglomerativos/divisivos

Los algoritmos de aglomeración comienzan con cada nodo de la red colaborativa en su propia comunidad, y en cada paso fusionan comunidades que se consideran suficientemente similares, que continúan hasta que se obtiene el número deseado de comunidades o se descubre que las comunidades restantes son demasiado diferentes. Los algoritmos de división operan en reversa; comienzan con toda la red como una comunidad, y en cada paso, eligen una comunidad determinada y la dividen en dos partes (Aggarwal 2011).

Los dos tipos de algoritmos de agrupamiento jerárquico suelen generar un dendrograma que es un árbol binario, donde las hojas son nodos de la red y cada nodo interno es una comunidad. En el caso de los algoritmos divisivos, una relación padre-hijo indica que la comunidad representada por el nodo padre se dividió para obtener las comunidades representadas por los nodos hijos. En el caso de los algoritmos de aglomeración, una relación padre-hijo en el dendrograma indica que las comunidades representadas por los nodos secundarios se aglomeraron (o fusionaron) para obtener la comunidad representada por el nodo padre.

La agrupación divisiva suele ser la estrategia de creación de comunidad más rápida y más fácil de paralelizar mientras que el agrupamiento aglomerativo permite más flexibilidad porque permite al usuario proporcionar una función arbitraria que define lo que constituye un buen par para agrupar. Esto es especialmente conveniente para datos que combinan diferentes tipos de propiedades y en dimensiones superiores. La agrupación incremental es útil cuando no se conocen inicialmente todos los puntos de datos, pero normalmente construye árboles de menor calidad y no se considerarán más aquí (Walter et al. 2008).

1.4.2.1 Algoritmo divisivo de Girvan y Newman

M.E.J. Newman y M. Girvan propusieron un algoritmo divisivo para el descubrimiento de comunidades, utilizando ideas de distancia entre bordes. Las medidas de distancia entre bordes se definen de manera tal que los bordes con puntajes de intersección altos tienen

más probabilidades de ser los bordes que conectan las diferentes comunidades. Es decir, los bordes entre las comunidades están diseñados para tener puntajes de intermediación más altos que los límites intracomunitarios. Por lo tanto, identificando y descartando tales bordes con altas puntuaciones de intermediación, se puede desconectar la red colaborativa en sus comunidades constituyentes (Newman y Girvan 2004).

1.4.2.2 Optimización codiciosa de la modularidad de Newman

El algoritmo opera con principios diferentes al de Girvan y Newman pero obtiene resultados cualitativamente similares. Consiste en un algoritmo de agrupamiento aglomerativo codicioso para optimizar la modularidad. La idea básica del algoritmo es que, en cada etapa, los grupos de vértices se fusionan sucesivamente para formar comunidades más grandes, de modo que la modularidad de la división resultante de la red aumenta después de cada fusión. Al principio, cada nodo en la red se encuentra en su propia comunidad, y en cada paso se eligen las dos comunidades cuya fusión conduce al mayor incremento en la modularidad. Solo se consideran aquellas comunidades que comparten al menos una ventaja, ya que la fusión de comunidades que no comparten ningún límite no puede dar como resultado un aumento en la modularidad (Newman 2004).

1.4.3 Algoritmos espectrales

Los algoritmos espectrales se encuentran entre los métodos clásicos para agrupamiento y descubrimiento de comunidades. Los métodos espectrales generalmente se refieren a algoritmos que asignan nodos a las comunidades en función de los vectores propios de las matrices, como la matriz de adyacencia de la propia red u otras matrices relacionadas. Los principales vectores propios definen una incrustación de los nodos de la red como puntos en un espacio k-dimensional, y posteriormente se pueden usar técnicas clásicas de agrupamiento de datos, como agrupamiento K-means para derivar la asignación final de nodos a grupos. La idea principal detrás del agrupamiento espectral es que la representación de baja dimensión, inducida por los vectores propios superiores, expone la estructura del grupo en el gráfico original con mayor claridad (Von Luxburg 2007).

1.4.4 Algoritmos de Partición de Grafos Multi-nivel

El algoritmo de particionamiento de gráficos multinivel reduce el tamaño del gráfico gradualmente al colapsar vértices y bordes en varios niveles, divide el gráfico más pequeño y luego lo desarrolla para construir una partición para el gráfico original. Además, en cada

paso, la partición se refina a medida que aumenta el grado de libertad (Aurora 2007). Los algoritmos de niveles múltiples proporcionan un marco poderoso para la partición de grafos de forma rápida y con alta calidad. La idea principal es reducir el grafo de entrada de manera sucesiva para obtener un grafo más pequeño, particionar este pequeño grafo y luego, volver a proyectar esta partición en el grafo original, volviendo a la partición en cada paso del camino (Aggarwal 2011).

1.5 Medidas de centralidad

Existen medidas para entender las redes y sus actores, las cuales ayudan a determinar la importancia y el rol de un actor en la red, en la presente investigación son utilizadas para determinar los autores más influyentes. Algunas de las más usadas son las medidas de centralidad. *"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"* (Lozano et al. 2016).

Las medidas de centralidad más usadas en el análisis de redes sociales son:

1.5.1 Centralidad de grado

La centralidad de grado asume que los nodos más importantes son los que tienen muchas conexiones con otros nodos. "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" (Kuz, Falco y Giandini 2016). Se calcula de la siguiente forma:

$C_{deg}(v) = \frac{d_v}{|N|-1}$ donde N es la cantidad de nodos en la red y d_v es el grado del nodo v.

$0 \leq C_{deg} \leq 1$ donde mientras más cerca esté de 1 más conexiones tendrá el nodo y mientras más cerca de 0 menos conexiones.

1.5.2 Centralidad de cercanía

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. 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

(Beltrán et al. 2015; Kuz, Falco y Giandini 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.

$C_{close}(v) = \frac{|N|-1}{\sum_{u \in N \setminus \{v\}} d_{(v,u)}}$ donde N va a ser la cantidad de nodos de la red y $d_{(v,u)}$ la longitud más corta de v a u.

1.5.3 Centralidad de intermediación

La centralidad de intermediación 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 (Kourtellis, Morales y Bonchi 2014). 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 y Upfal 2016). La centralidad de intermediación asume que los nodos importantes en la red son aquellos que conectan otros nodos.

$C_{btw}(v) = \sum_{s,t \in N} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$ donde $\sigma_{s,t}$ es el número de caminos más cortos entre los nodos s y t, $\sigma_{s,t}(v)$ es el número de caminos más cortos entre los nodos s y t que pasan por el nodo v.

1.6 Proceso de Descubrimiento del Conocimiento (KDD)

El Proceso de Descubrimiento del Conocimiento o KDD (Knowledge Discovery in Databases, por sus siglas en inglés) es definido por (Fayyad, Piatetsky-Shapiro y Smyth 1996) como el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles en los datos. El KDD se organiza entorno a cinco fases (Figura 2):

1. **Integración y recopilación de datos:** Se determinan las fuentes de información que pueden ser útiles, seguidamente se transforman los datos a un formato común, unificando toda la información recogida, detectando y resolviendo inconsistencias.
2. **Selección, limpieza y transformación:** En esta fase se eliminan o corrigen los datos incorrectos y se decide la estrategia a seguir con los datos incompletos. También se proyectan los datos para considerar solo las variables y atributos que van a ser relevantes para hacer más fácil la tarea propia de la minería y así sean más útiles los resultados de la misma.

3. **Minería de Datos:** Esta es la fase donde se define cual es la tarea a realizar y se elige el método que se va a utilizar.
4. **Evaluación e interpretación de los datos:** Se evalúan los patrones y se analizan los expertos.
5. **Difusión y uso:** En esta última fase se hace uso del nuevo conocimiento y se hace partícipe de él a todos los posibles usuarios.

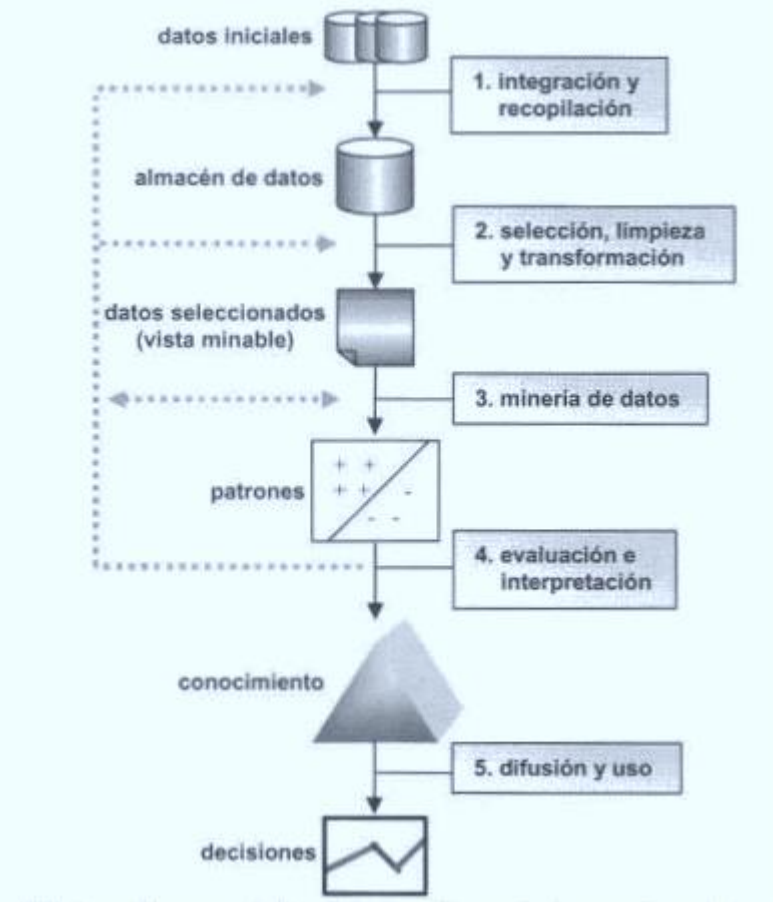

Figura 4. Fases del Procesos KDD (Fuente: Hernández Orallo, Ramírez Quintana y Ferri Ramírez 2004)

1.7 Tecnologías y herramientas

Para la realización de un software se deben tener en cuenta las tendencias actuales en torno a las tecnologías y herramientas a utilizar para el desarrollo del mismo. El objetivo es lograr que el trabajo se realice de forma más sencilla sin afectar la calidad del producto final. Se deben utilizar tecnologías avanzadas y en continuo progreso, para que de esta forma la aplicación cumpla con las expectativas de los usuarios finales y además se pueda actualizar la misma con gran facilidad. Las tecnologías que se describen en el presente capítulo son las que serán utilizadas para el desarrollo de la solución propuesta.

1.7.1 Lenguaje de programación y plataforma

Python es un lenguaje de programación interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos (González Duque 2015). Sobresale en la manipulación de datos y la programación de red por la cantidad de librerías que contiene y que facilitan el trabajo para los desarrolladores. Python ha desarrollado una gran y activa comunidad científica de computación y análisis de datos. Ha pasado de ser un lenguaje informático científico a ser uno de los lenguajes más importantes para la ciencia, el aprendizaje automático y el desarrollo general de software en la academia y la industria (McKinney 2012). En el presente trabajo se decidió usar Python 3.5.0 para realizar la implementación de la solución propuesta por la variedad de librerías que contiene para el trabajo con redes colaborativas.

Anaconda

La distribución de código abierto Anaconda facilita el proceso de implementar soluciones relacionadas con manipulación de datos en Python y cuenta con más de seis millones de usuarios. Incluye cientos de paquetes de código abierto, así como el paquete conda y el administrador de entorno virtual para Windows, Linux y MacOS. Conda facilita y agiliza el proceso de instalar, ejecutar y actualizar entornos complejos como SciPy. Anaconda es la base de millones de proyectos, así como de aprendizaje automático de Amazon Web Services y Anaconda para Microsoft en Azure y Windows (Anaconda, Inc 2018). Se decidió que la versión Anaconda3-2.5.0 es la más adecuada para trabajar con la versión lenguaje de programación seleccionado, debido a que contiene una amplia colección de librerías empleadas para facilitar el trabajo con Python.

Algunos de los módulos y librerías más destacadas que incluye Anaconda3-2.5.0 son:

- **MatplotLib**: Es una librería de trazado 2D de Python que produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos en todas las plataformas. Matplotlib se puede utilizar en scripts Python, el shell Python e IPython, el bloc de notas jupyter, entre otros (matplotlib.org 2017).
- **NetworkX**: Es una biblioteca de Python muy útil para la creación, manipulación y estudio de redes complejas.
- **Re**: Este módulo proporciona facilidades en el trabajo con expresiones regulares. Una expresión regular especifica un conjunto de cadenas que coinciden con ella; las

funciones en este módulo permiten verificar si una cadena en particular coincide con una expresión regular dada (Python Software Foundation 2018a).

- **BeautifulSoup**: Es una biblioteca de Python para extraer datos de archivos HTML y XML. Funciona con un analizador sintáctico para proporcionar formas idiomáticas de navegar, buscar y modificar el árbol de análisis sintáctico (Richardson 2015).
- **Urllib.request**: Este módulo define funciones y clases que ayudan a abrir URL (principalmente HTTP), en la autenticación, re direccionamiento, cookies y más (Python Software Foundation 2018b).
- **Urllib.parse**: Este módulo define una interfaz estándar para dividir las cadenas de localizador uniforme de recursos (URL) en componentes (esquema de direcciones, ubicación de red, ruta, etc.). El módulo urllib.parse define funciones que se dividen en dos amplias categorías: análisis de URL y cotización de URL (Python Software Foundation 2018c).

1.7.2 Herramientas existentes para la visualización y análisis de redes

En el análisis de redes colaborativas, un elemento clave es la visualización de los resultados para lograr un mayor entendimiento de los mismos. Diferentes soluciones han sido desarrolladas para el trabajo con redes sociales, no solo para la visualización sino también para lograr que la aplicación de las técnicas de ARS sea más efectiva. Entre las soluciones que más se destacan se encuentran las siguientes herramientas:

- **SocNetV (Social Network Analysis and Visualization Software:** Es una aplicación de software libre multiplataforma y fácil de usar para el análisis y la visualización de redes sociales. Permite dibujar redes sociales con unos pocos clics en un lienzo virtual, cargar datos de campo de un archivo en un formato compatible (GraphML, GraphViz, Adjacency, EdgeList, GML, Pajek, UCINET, etc.) o rastrear Internet para crear una red colaborativa de conectado páginas web (Grand Jean 2018).
- **NodeXL Basic**: Es una plantilla gratuita de código abierto para Microsoft Excel 2007, 2010, 2013 y 2016 que facilita la exploración de redes. NodeXL permite ingresar listas de bordes de red en hojas de trabajo, y mediante botones ver el grafo. La herramienta cuenta con una versión que ofrece funciones adicionales, proporcionando un acceso fácil a las secuencias de datos de las redes sociales,

métricas de red avanzadas, análisis de mensajes y de texto, y generación de informes potentes (Smith 2017).

- **Gephi:** Es el software líder de visualización y exploración para todo tipo de gráficos y redes. Gephi es de código abierto y gratuito (Gephi 2017). Algunas de sus principales aplicaciones es el análisis de redes sociales y el análisis exploratorio de Datos para la intuición mediante la manipulación de redes en tiempo real. El software trabaja con los formatos GDF (GUESS), GraphML (NodeXL), GML, NET (Pajek) y GEXF. Usa un motor de renderizado 3D para mostrar grandes redes en tiempo real y acelerar la exploración. Tiene una arquitectura flexible y de múltiples tareas que ofrece nuevas posibilidades para trabajar con conjuntos de datos complejos y producir resultados visuales valiosos. Proporciona un acceso fácil y amplio a los datos de la red permitiendo espaciar, filtrar, navegar, manipular y agrupar (Bastian, Heymann y Jacomy 2009).

  La herramienta elegida para representar la red y los resultados que se derivan de esta es Gephi en su versión 0.9.2 por ser software líder de visualización. Además, Gephi es ideal para visualizaciones básicas de la red por la alta calidad que presentan sus resultados.

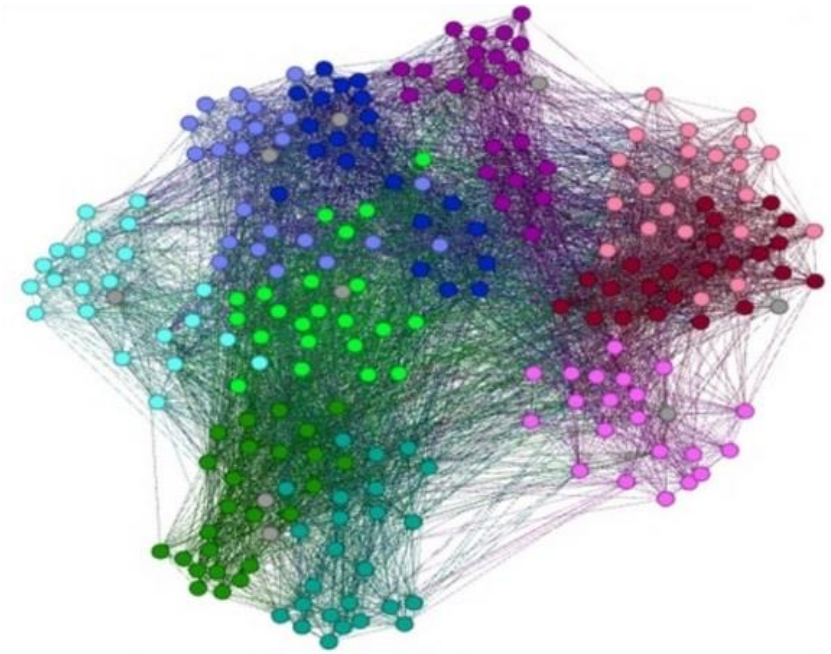

Figura 5. Ejemplo de visualización de una red compleja con Gephi (Fuente: Cherven 2013)

1.7.3 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés) es un software que facilita el desarrollo de aplicaciones. Están diseñados para abarcar todas las tareas de programación en una aplicación y ofrecen una interfaz que presenta todas las herramientas que un desarrollador necesita, incluyendo entre otras: compilador y depurador de errores.

PyCharm

Se decidió utilizar PyCharm 2017.3.3 para el desarrollo del método ya que es un IDE que proporciona el completamiento inteligente de códigos, inspecciones de códigos, resaltado de errores sobre la marcha y soluciones rápidas, junto con refactorizaciones automáticas de códigos y capacidades de navegación avanzadas. Se integra con IPython Notebook, tiene una consola interactiva de Python y es compatible con Anaconda, así como con múltiples paquetes científicos, incluidos matplotlib y NumPy. Además de Python, PyCharm admite JavaScript, CoffeeScript, TypeScript, Cython, SQL, HTML / CSS, lenguajes de plantilla, AngularJS, Node.js y más. También posee una gran colección de herramientas lista para usar: un depurador integrado y un corredor de prueba; Python Profiler; una terminal incorporada; e integración con sistemas de control de versiones y herramientas de base de datos incorporadas (JetBrains 2017).

1.7.4 Sistema de control de versiones

Git es un sistema de control de versiones creado por Linus Torvalds y la comunidad de desarrollo de Linux en 2005 con el objetivo de sustituir el software que empleaban para realizar esta función. El control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante. Entre las ventajas que ofrece Git están la velocidad, el diseño sencillo y la capacidad de manejar grandes proyectos (Chacon y Straub 2004). En el presente trabajo se escogió la versión Git 2.8.0 para el control de versiones por la compatibilidad que tiene con el IDE seleccionado anteriormente.

1.7.5 Lenguaje de modelado unificado y herramienta CASE

El lenguaje de modelado unificado (UML) es un lenguaje estándar para escribir diseños de software. Puede usarse para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo (Pressman 2010). En la investigación se empleó el

UML para la construcción del modelo conceptual y se trabajó utilizando una herramienta CASE.

La ingeniería de software asistida por computadora (CASE, por sus siglas en inglés) consiste en diversas aplicaciones que pueden ayudar en el ciclo de vida de desarrollo de software. Visual Paradigm es una herramienta de diseño y administración poderosa, multiplataforma y fácil de usar para los sistemas de tecnología de información. Ofrece a los desarrolladores de software una plataforma de desarrollo innovadora para crear aplicaciones de calidad de manera más rápida. Facilita una excelente interoperabilidad la mayoría de los entornos de desarrollo integrado (Visual Paradigm 2018). En la investigación se trabajó con la versión 8.0 de la herramienta y se empleó para la realización del modelo conceptual.

1.8 Conclusiones del capítulo

A partir del análisis de la información presentada en este capítulo puede arribarse a las siguientes conclusiones:

1. Los datos contenidos en el fichero changelog de cada paquete del repositorio del sistema operativo Nova ofrecen una entrada de datos para el método que se pretende implementar como propuesta de solución.
2. El análisis de redes colaborativas es la técnica que puede dar solución al problema planteado ya que permite representar las relaciones entre los desarrolladores de paquetes. La detección de comunidades permite determinar grupos que se establecen en la red a partir de las relaciones entre desarrolladores y las medidas de centralidad permiten identificar los actores más influyentes dentro de la red.
3. Todos los algoritmos presentados en la investigación pueden ser empleados para resolver el problema planteado.
4. El KDD se utiliza en la propuesta de solución para identificar las fuentes de información que pueden ser útiles, así como los patrones válidos dentro de la fuente de datos y eliminar o corregir los datos incorrectos.
5. Como resultado del estudio de las tecnologías apropiadas para el desarrollo de la solución se determinó emplear Python como lenguaje de programación en su versión 3.5 y, y PyCharm en su versión 2017.3.3 como entorno de desarrollo integrado para la implementación de la solución.

6. El estudio de las tendencias actuales sobre las herramientas que permiten el trabajo y visualización de redes sociales, permitió identificar a Gephi como software líder para visualizar grafos y redes.

CAPÍTULO 2. Método para la detección de comunidades de desarrolladores y actores más influyentes en repositorios de sistemas operativos libres

En el presente capítulo se presenta un método para determinar las comunidades de desarrolladores de software libre a partir de su participación en la implementación de paquetes para los repositorios de las distribuciones GNU/Linux.

2.1 Modelo conceptual

Para la correcta implementación de la solución propuesta es necesario conocer el contexto en el que se enmarca la investigación. Existen formas de expresar los conceptos asociados al dominio del problema que se pretende resolver, como es el modelo conceptual o modelo del domino. Un modelo conceptual muestra clases conceptuales del mundo real significativas en un dominio del problema y no componentes de software, la identificación de las clases conceptuales forma parte del estudio del dominio del problema. Un modelo conceptual puede verse también como una representación visual de las clases conceptuales en un dominio de interés o como un diccionario visual de las abstracciones relevantes, vocabulario del dominio e información del mismo (Larman 2003).

Con el objetivo de describir y expresar el contexto en que se desarrolla la investigación se elaboró un modelo conceptual general que puede apreciarse en la siguiente figura:

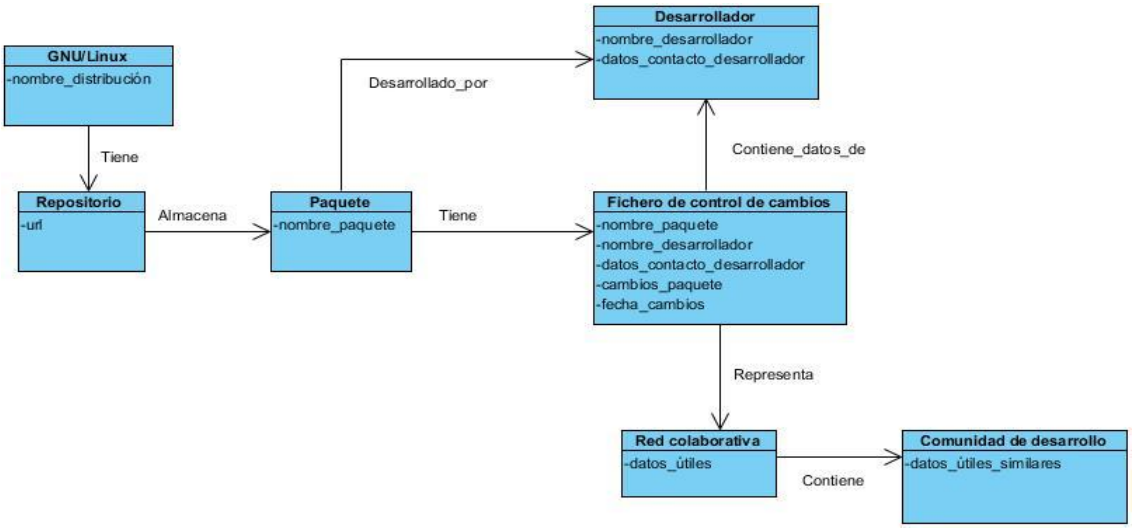

Figura 6. Modelo Conceptual (Fuente: Elaboración propia)

Para una mejor comprensión del modelo conceptual, se proporciona un marco conceptual con las definiciones identificadas en el contexto de la detección de comunidades de desarrolladores en repositorios de sistemas operativos de software libre, estas son:

Desarrollador: Persona encargada de implementar o desarrollar paquetes para los repositorios de las distribuciones GNU/Linux y que interactúa directamente con el fichero de control de cambios.

GNU/Linux: Es un sistema operativo libre llamado GNU donde el programa que asigna los recursos de la computadora a todos los demás programas para que puedan funcionar recibe el nombre de Linux (Stallman 2002). Cuenta con repositorios de paquetes desde donde se puede instalar o descargar paquetes para mejorar el trabajo con el software.

Paquete: Un paquete se refiere a un archivo comprimido que contiene todos los archivos que vienen con una aplicación en particular. Los archivos generalmente se almacenan en el paquete de acuerdo con sus rutas de instalación relativas en su sistema. La mayoría de los paquetes también contienen instrucciones de instalación para el sistema operativo, así como una lista de otros paquetes que son dependencias (InternetBlog.Org.Uk 2010).

Fichero de control de cambios: Es el archivo donde se almacena información sobre los cambios realizados en el paquete, así como los datos de contacto del desarrollador que los realizó.

Las definiciones de **comunidad de desarrollo**, **repositorio** y **colaborativa** son abordadas con anterioridad en la investigación.

2.2 Presentación de la solución

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 (ver epígrafe 2.3).

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 (ver epígrafe 2.4).

3 Se crea la red colaborativa a partir de los datos obtenidos (ver epígrafe 2.5).

4 Se detectan las comunidades de desarrolladores y actores más influyentes en la red (ver epígrafe 2.6).

A continuación, se muestra en la figura 7 como se realiza el método antes mencionado para la implementación de la solución:

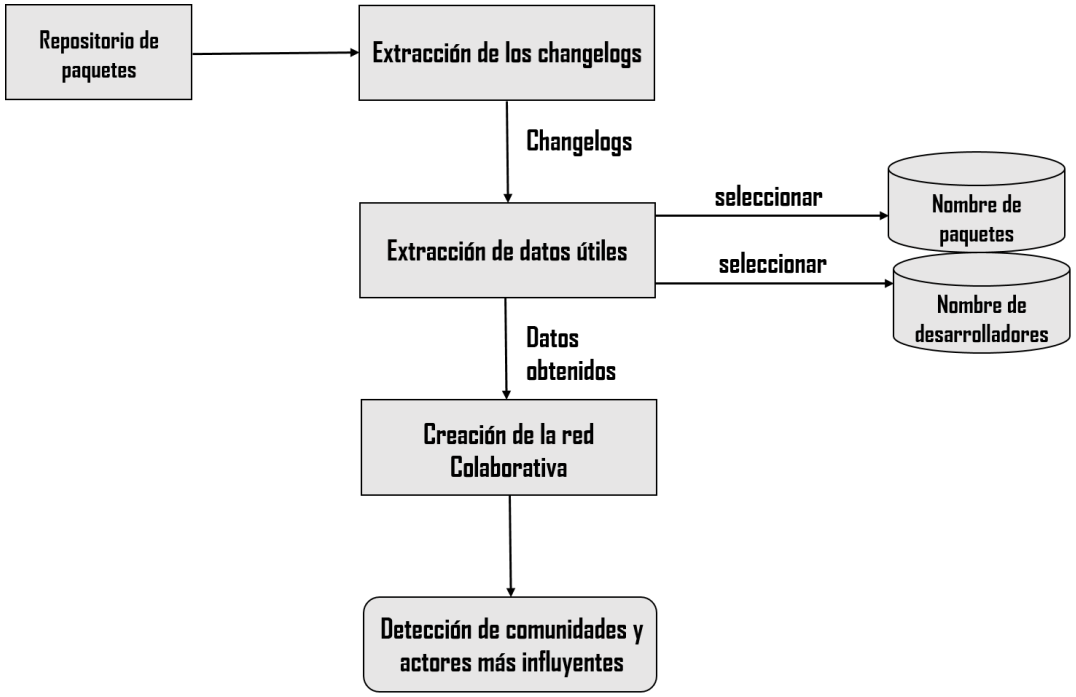

Figura 7. Esquema del método propuesto (Fuente: Elaboración propia)

2.3 Extracción de los ficheros de control de cambios

Un fichero de control de cambios o changelog 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 (ver figura 8) lo cual permite que la salida sea siempre similar independientemente de la cantidad de entradas que se procesen.

```
Nombre del paquete (Versión del paquete) + Información adicional

 * Información del paquete

-- Nombre del desarrollador 1 <email> fecha y hora del cambio

Nombre del paquete (versión donde se realizaron los cambios) + Información adicional

 *Información sobre los cambios que implementó el desarrollador 1 del paquete
.
.
.
.
.
-- Nombre del desarrollador n <email> fecha y hora del cambio

Nombre del paquete (versión donde se realizaron los cambios) + Información adicional

 *Información sobre los cambios que implementó el desarrollador n del paquete
```

Figura 8. Estructura básica de un fichero de control de cambios (Fuente: Elaboración propia)

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. Para acceder a los repositorios es necesario conocer las direcciones de los mismos las cuales suelen ser URL's (Localizador Uniforme de Recursos, por sus siglas en español), 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. Algunas URL's donde pueden encontrarse repositorios de paquetes son las ofrecidas por la comunidad de software libre de Cuba "HumanOS", ejemplo de estas son:

http://novarepo.uci.cu/nova

http://ubuntu.uci.cu/ubuntu

http://debian.uci.cu/debian

La siguiente figura muestra cómo se organizan los paquetes dentro de un repositorio:

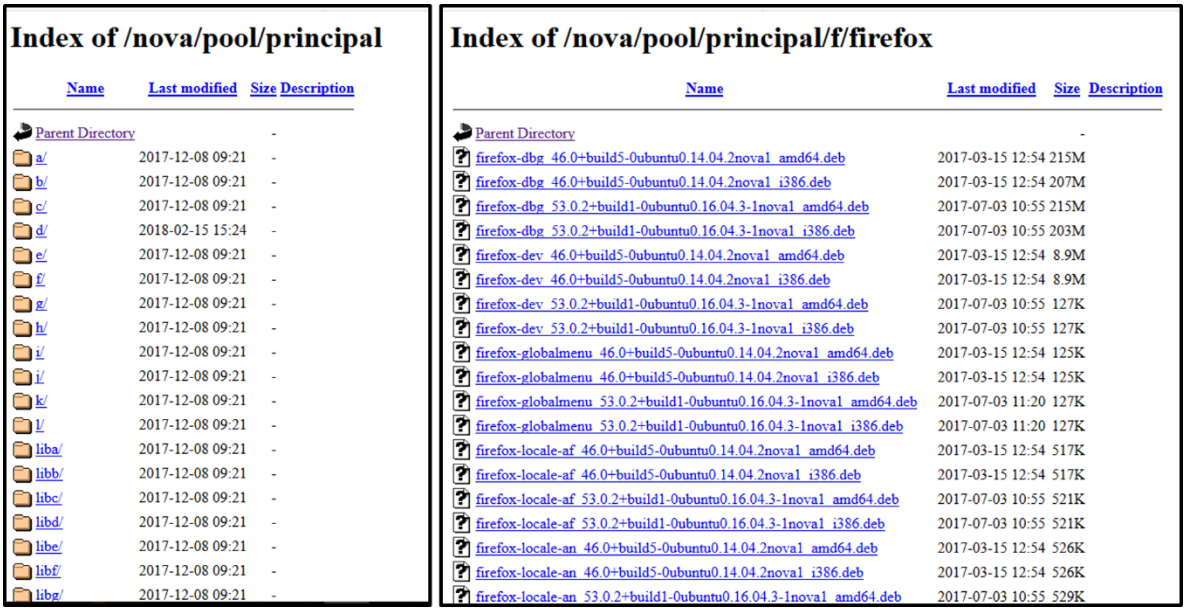

Figura 9. Vista de un repositorio de paquetes (Fuente: Elaboración propia)

Después de descomprimir los archivos que se observan en la imagen anterior puede accederse el changelog mediante el archivo “data.tar.xz”, de la forma que se muestra en la figura siguiente:

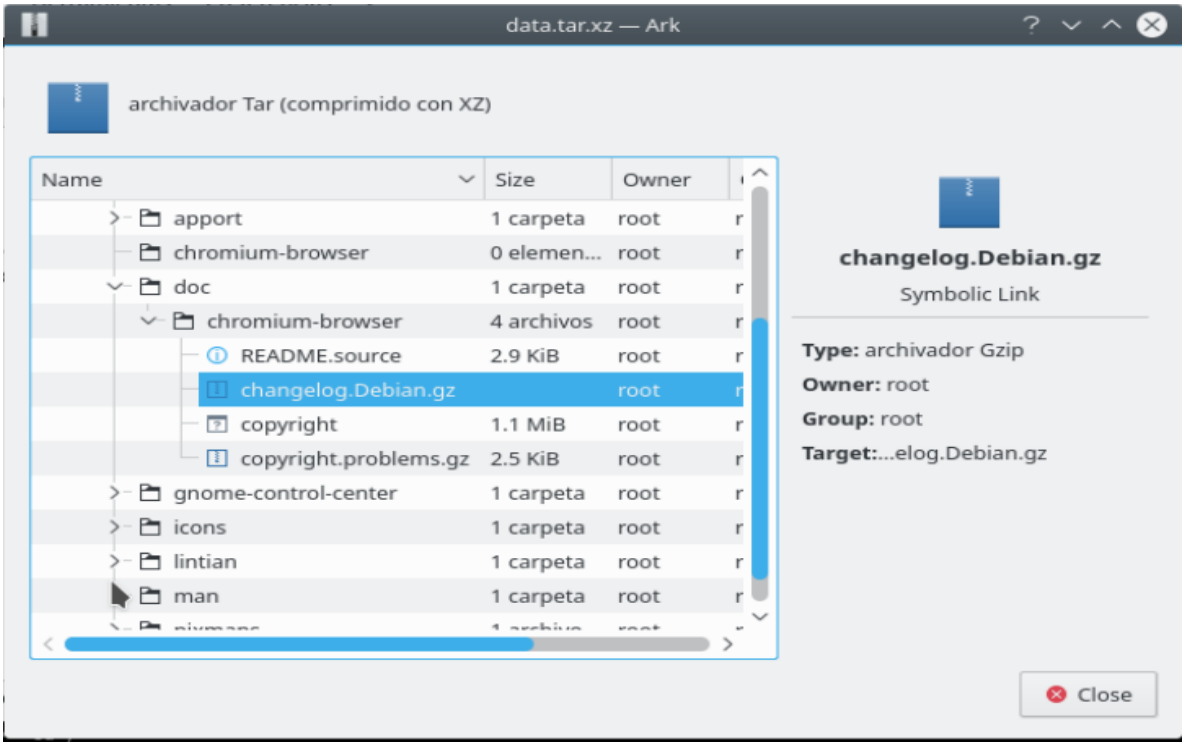

Figura 10. Ubicación del changelog dentro de un paquete “.deb” (Fuente: Elaboración propia)

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 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.

A continuación, se muestra un esquema general de cómo se realiza la extracción de los changelogs del repositorio:

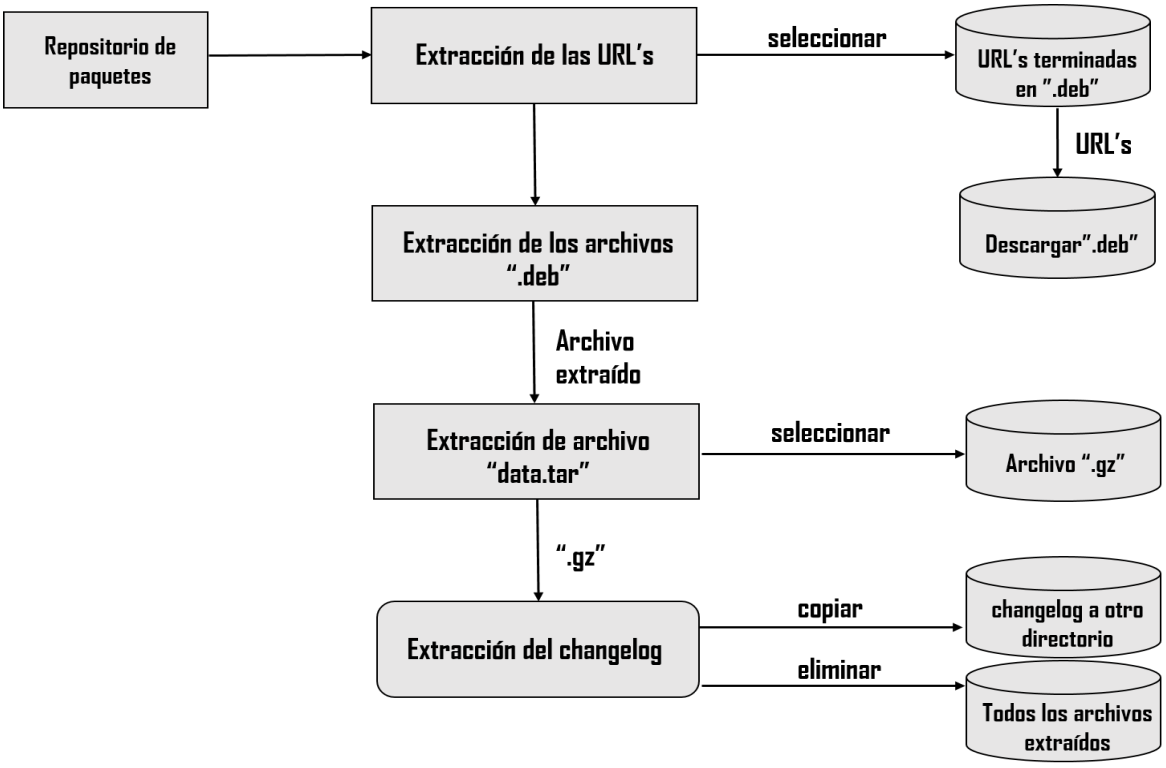

Figura 11. Esquema de extracción de los changelogs (Fuente: Elaboración propia)

2.3.1 Pseudocódigo del algoritmo propuesto para la extracción de los changelogs

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 Entonces
2:    Llamar método ddcr
3:  Fin
4:  Si no Entonces
5:    Obtener direcciones URL
6:    Para direcciones URL Hacer
7:      Si dirección url == "../" Entonces
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.

2.4 Extracción de datos útiles a partir del fichero de control de cambios

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 (ver figura 8) 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+”.

A continuación, se muestra un esquema general de cómo se realiza la extracción de datos útiles:

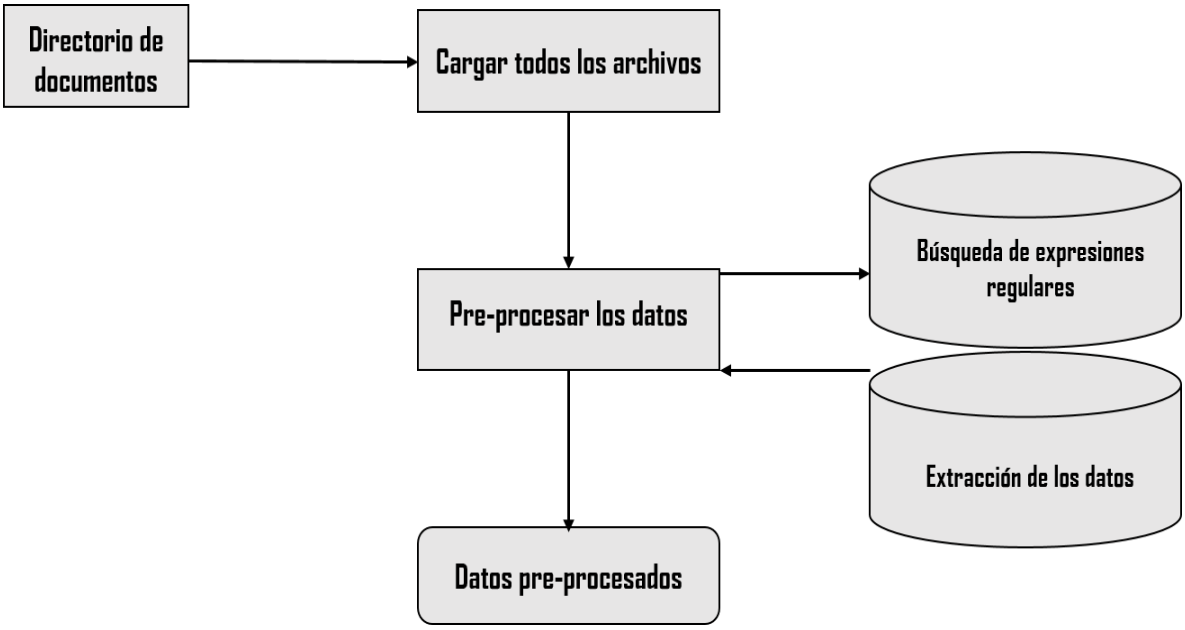

Figura 12. Esquema de extracción de datos útiles (Fuente: Elaboración propia)

Para realizar la carga y el pre-procesamiento de los datos el programador puede auxiliarse del siguiente seudocódigo propuesto por el autor de la investigación.

2.4.1 Pseudocódigo del algoritmo propuesto para la extracción de datos

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 == "@" Entonces
4:       Si expresión regular desarrollador == palabras Entonces
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 paquete== palabras Entonces
11:     Agregar palabras a la lista de paquetes
12:   Fin
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
```

2.5 Creación de la red colaborativa

En este paso se procede a 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.

A continuación, se muestra un esquema general de cómo se realiza la creación de la red colaborativa:

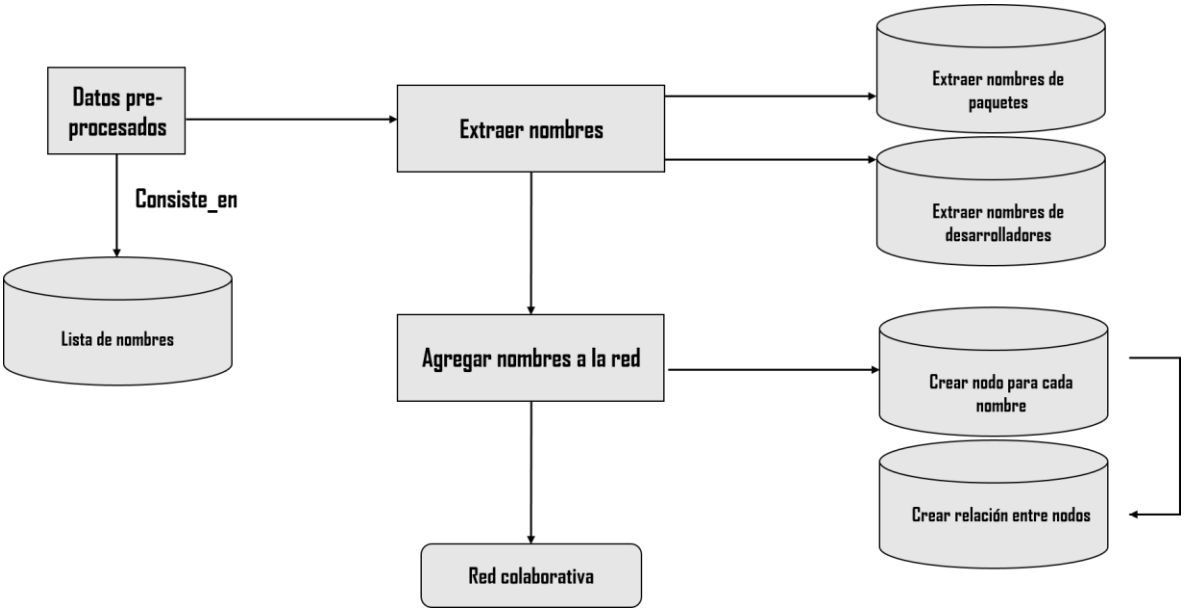

Figura 13. Esquema de creación de la red colaborativa (Fuente: Elaboración propia)

Es importante señalar que la lista de nombres de paquetes y desarrolladores que se usa como entrada para crear la red colaborativa con NetworkX tiene la siguiente estructura:

```
[['Nombre_paquete_1', 'Nombre_desarrollador_1',....,'Nombre_desarrollador_N1'],
...,['Nombre_paquete_2','Nombre_desarrollador_2',...,'Nombre_desarrollador_N2']]
```

Figura 14. Estructura de lista de nombres de paquetes y desarrolladores (Fuente: Elaboración propia)

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

2.5.1 Pseudocódigo del algoritmo propuesto para la creación de la red colaborativa

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 con la estructura que muestra la figura 14.

```
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
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
```

2.6 Detección de las comunidades de desarrolladores y actores más influyentes

En este paso se procede a analizar la red colaborativa con la herramienta profesional para la visualización de redes Gephi. El análisis con la herramienta antes mencionada permite observar con mayor detalle la 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 los diferentes algoritmos y medidas existentes para ello, alguno de los cuales han sido explicados anteriormente (ver epígrafe 1.4 y 1.5).

A continuación, se muestra un esquema general de cómo se realiza la creación de la red colaborativa:

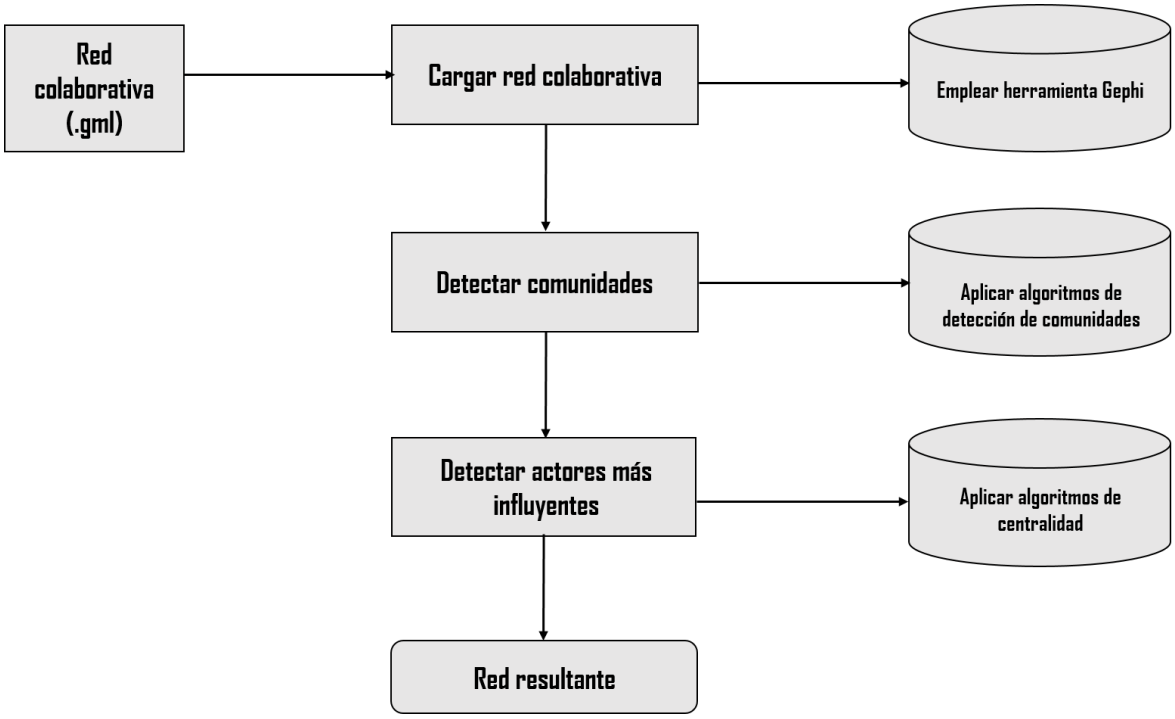

Figura 15. Esquema de detección de comunidades y actores más influyentes en la red colaborativa (Fuente: Elaboración propia)

Para cargar la red con Gephi basta con acceder al menú archivo, hacer clic en la opción abrir y buscar el archivo ".gml"; después de realizada esta operación se mostrará la red en el área de trabajo de la herramienta. Para detectar las comunidades Gephi cuenta con un apartado de estadísticas que se ubica en la parte derecha de la interfaz desde donde pueden elegirse criterios para la detección de comunidades y actores más influyentes en la red, estos trabajan con los algoritmos mencionados en el epígrafe 1.4 y con las medidas mencionadas en el epígrafe 1.5.

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 siguiente figura:

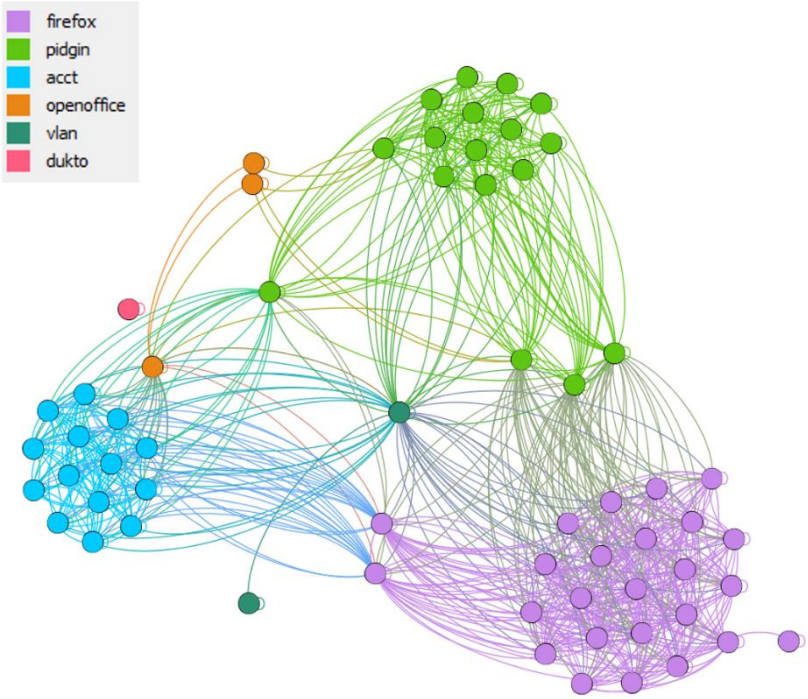

Figura 16. Red colaborativa con los desarrolladores de paquetes identificados (Fuente: Elaboración propia)

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. La siguiente figura representa los paquetes del repositorio con el color verde (0) y los desarrolladores con el color rosa (1).

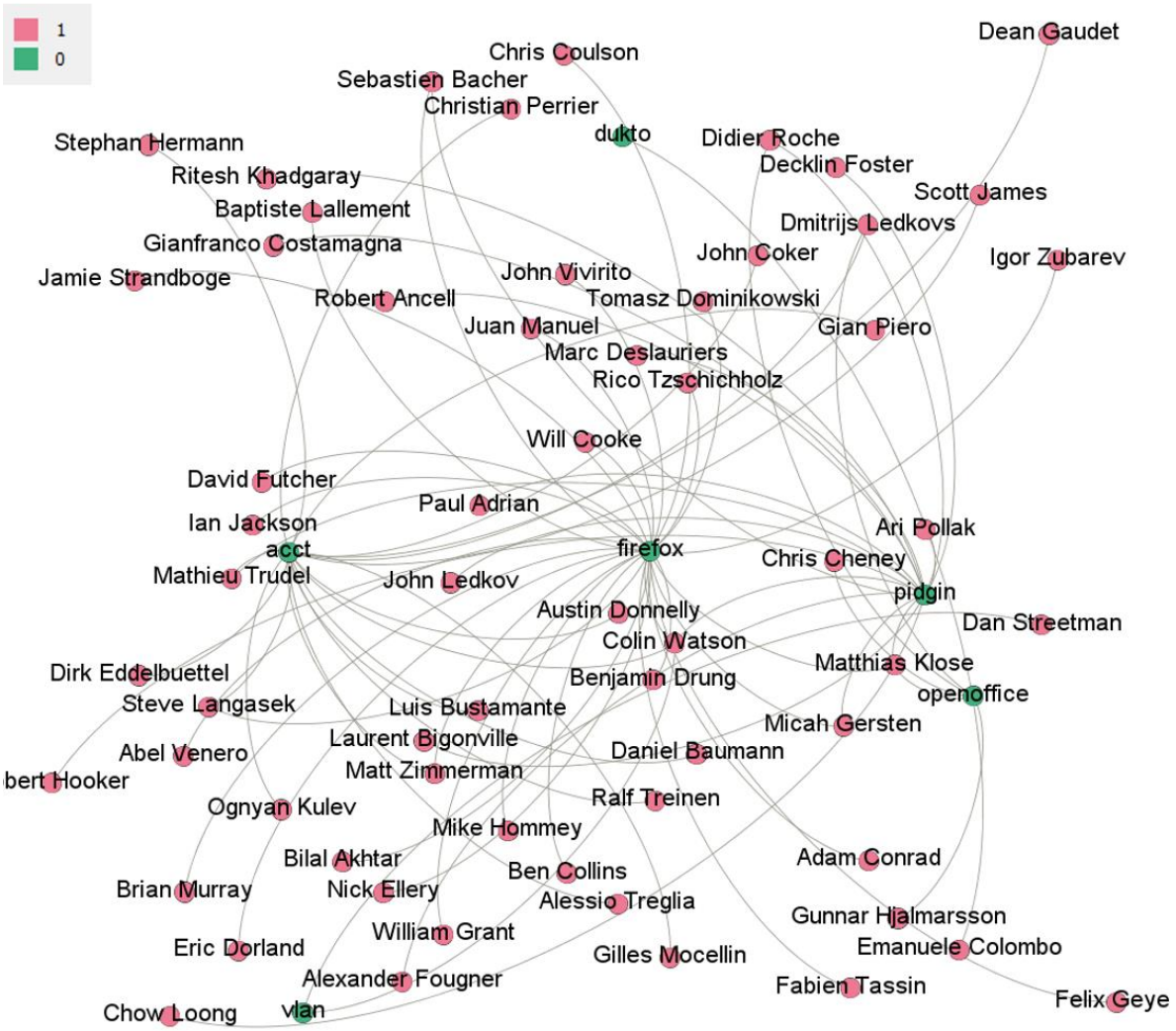

Figura 17. Grafo bipartido de nombres de paquetes y desarrolladores (Fuente: Elaboración propia)

Después de realizada la detección de las comunidades estas pueden ser identificadas en la red colaborativa principalmente a través de colores como muestra la siguiente figura:

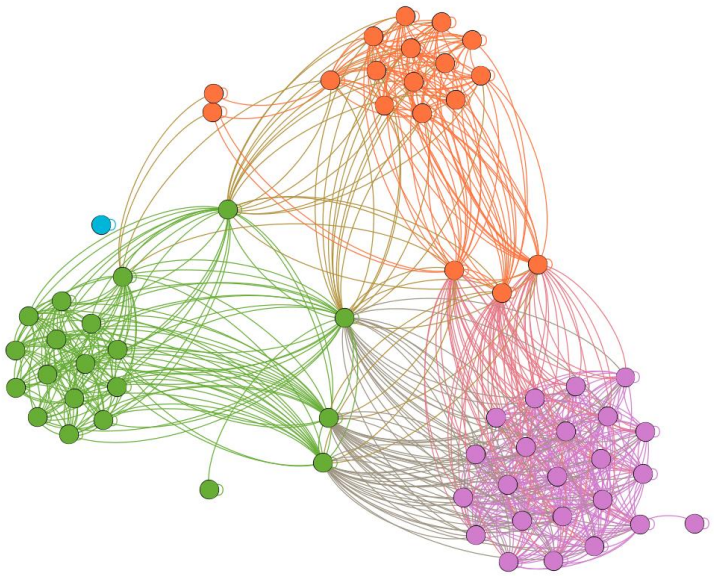

Figura 18. Comunidades detectadas en la red colaborativa (Fuente: Elaboración propia)

Además de las comunidades se logró identificar los desarrolladores más importantes dentro de la red basándose en el criterio o centralidad de intermediación que establece que los nodos más importantes son aquellos que conectan otros nodos. La siguiente figura muestra los desarrolladores más importantes:

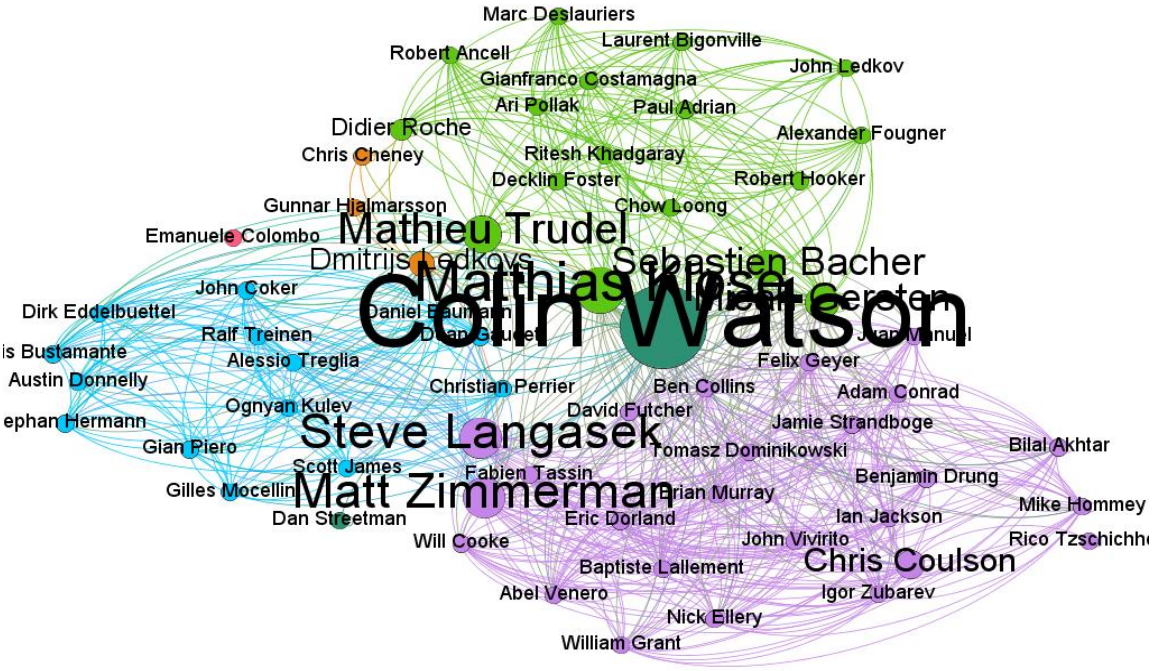

Figura 19. Desarrolladores más importantes de la red colaborativa (Fuente: Elaboración propia)

Luego de aplicarle todos los criterios expuestos a la red, esta quedaría finalmente de la siguiente forma:

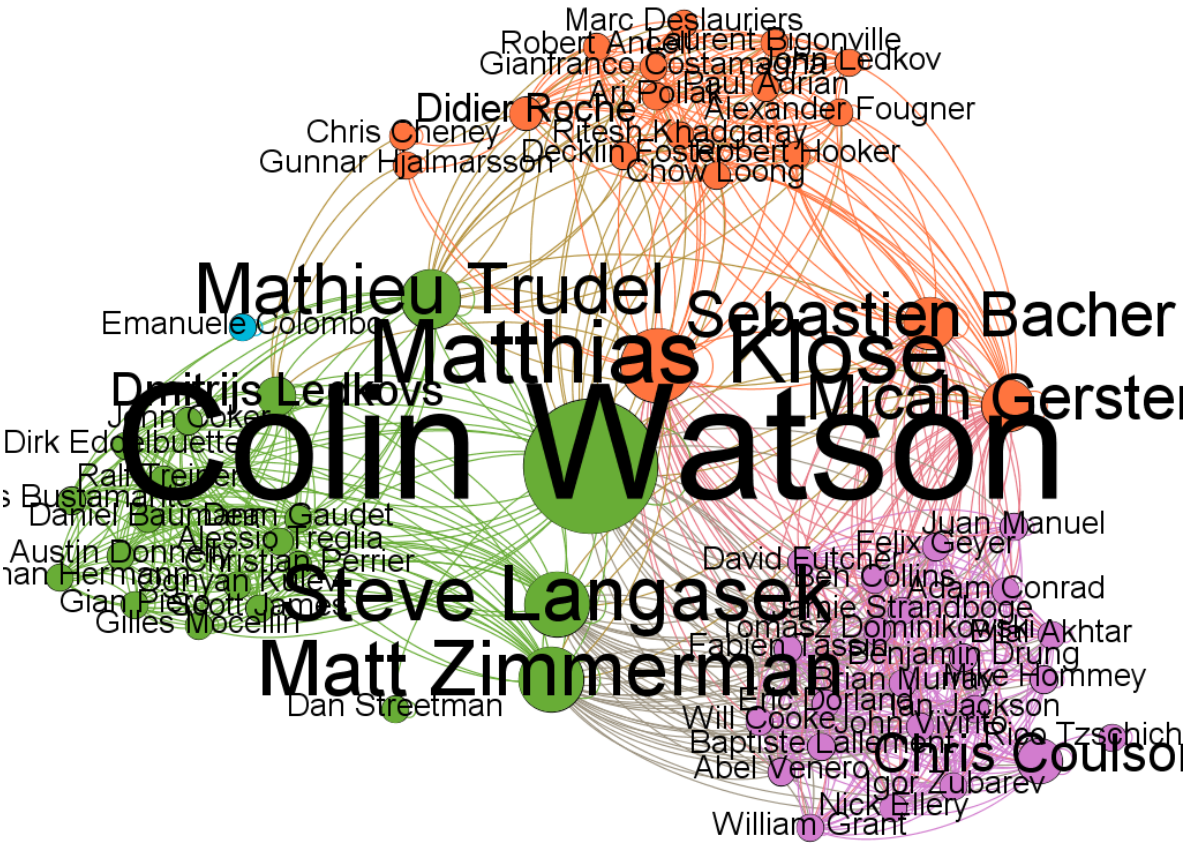

Figura 20. Red colaborativa con todos los criterios aplicados (Fuente: Elaboración propia)

2.7 Conclusiones del capítulo

En este capítulo se propuso un método para la detección de comunidades de desarrolladores en repositorios de sistemas operativos libres. Luego de presentada la propuesta, se concluye que:

1. El uso del modelo conceptual permitió describir el contexto a partir de la identificación de los principales conceptos asociados a la investigación.
2. Se logró detectar las comunidades de desarrolladores y los actores más influyentes en una red colaborativa donde se representan los paquetes del repositorio de los sistemas operativos de libres y su relación con los desarrolladores que intervienen en cada uno de ellos.

CAPÍTULO 3: Validación del método desarrollado

En el presente capítulo se realiza la validación del método desarrollado mediante varios criterios para determinar que la solución propuesta responde satisfactoriamente a los objetivos que se trazaron. Se realizan varias pruebas al código para asegurar la calidad del método propuesto verificando que no existan fallas.

3.1 Entorno de pruebas

El objetivo de probar el software es encontrar errores, y una buena prueba es aquella donde la probabilidad de encontrar uno es alta. Un sistema informático debe diseñarse e implementarse teniendo en cuenta la facilidad con la que debe probarse posteriormente.

Para la realización de las pruebas de software a la solución desarrollada se trabajó con las tecnologías y herramientas seleccionadas (ver epígrafe 1.7) empleando como plataforma el sistema operativo Windows 10 Pro de 64 bits. Las características de hardware sobre el que se trabajó fue una laptop TOSHIBA Satellite C55-B con un microprocesador Intel(R) Celeron(R) N2830 a 2.16 GHz y una memoria RAM DDR 3 de 4GB.

Se tuvo en cuenta los niveles de pruebas de unidad y pruebas de aceptación que a continuación se describe.

3.2 Pruebas de unidad

El alcance de las pruebas en este nivel está dado por la intención de probar el código fuente. Las pruebas se realizaron de forma automática. Para realizar las pruebas unitarias automáticamente en el lenguaje de programación Python se utilizó el módulo “unittest” al cual se accedió mediante Anaconda3-2.5.0. La siguiente figura muestra el resultado de aplicar las pruebas unitarias de forma automática al código implementado:

```
67  class Test_URL_dfs(unittest.TestCase):
68      def test_dfs2(self):
69          self.assertIsNone(dfs("http://ubuntu.uci.cu/ubuntu/pool/main/a/a11y-profile-manager/"))
```

Figura 21. Código de la prueba unitaria del método implementado (Fuente: Elaboración propia)

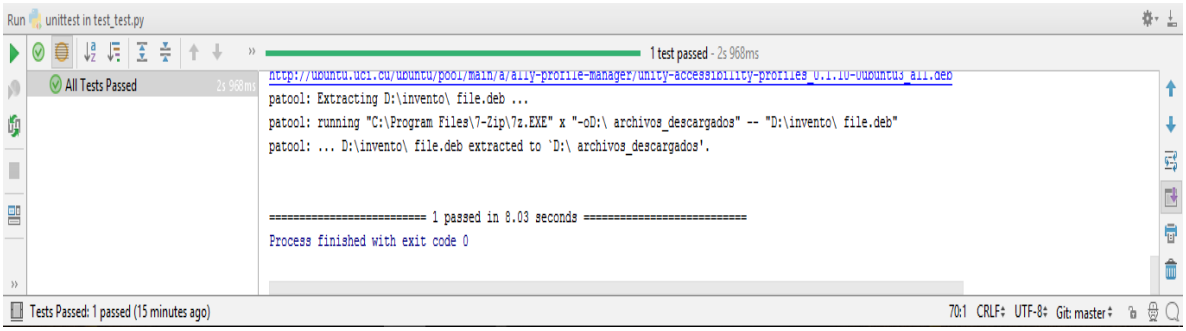

Figura 22. Cuadro de respuesta a las pruebas unitarias (Fuente: Elaboración propia)

3.3 Pruebas de aceptación

Según (IEEE 2008) las pruebas de aceptación son pruebas realizadas para establecer si un sistema satisface sus criterios de aceptación y para permitirle al cliente determinar si acepta el sistema. También se define como una prueba formal realizada para permitir que un usuario, cliente u otra entidad autorizada determine si acepta un sistema o componente.

A continuación, se muestra un escenario para los casos de prueba realizados de la propuesta de solución conformada por tres iteraciones representadas por tablas:

Iteración 1

Tabla 1. Caso de prueba de aceptación PA1-HU (Fuente: Elaboración propia)

Caso de prueba de aceptación	
Código: PA1-HU	**Historia de usuario:** Extraer changelogs de los paquetes de los repositorios de sistemas operativos libres.
Nombre: Extraer changelogs de los paquetes de los repositorios de sistemas operativos libres.	
Descripción: Permite extraer los changelogs de los paquetes que se encuentren en la URL definida.	
Condición de ejecución: Tiene que existir al menos un paquete en el repositorio**.**	
Entrada/ Pasos de ejecución: • El usuario inicia la corrida del método.	
Resultado esperado: Todos los changelogs pertenecientes a los paquetes de la URL definida se copian en un directorio.	
Evaluación de la prueba: Satisfactoria	

Iteración 2

Tabla 2. Caso de prueba de aceptación PA2-HU (Fuente: Elaboración propia)

Caso de prueba de aceptación	
Código: PA2-HU	**Historia de usuario:** Extraer datos útiles de los changelogs.
Nombre: Extraer datos útiles de los changelogs	
Descripción: Permite extraer los datos útiles de los changelogs de los paquetes que encuentran en un directorio definido.	
Condición de ejecución: Tiene que existir al menos un changelog en el directorio**.**	
Entrada/ Pasos de ejecución: • El usuario inicia la corrida del método.	
Resultado esperado: Se obtienen dos listas con los nombres de paquetes y desarrolladores contenidos en los changelogs.	
Evaluación de la prueba: Satisfactoria	

Iteración 3

Tabla 3. Caso de prueba de aceptación PA3-HU (Fuente: Elaboración propia)

Caso de prueba de aceptación	
Código: PA3-HU	**Historia de usuario:** Crear red colaborativa.
Nombre: Crear red colaborativa	
Descripción: Permite crear una red colaborativa con las relaciones entre desarrolladores de un mismo paquete de un repositorio de sistema operativo libre.	
Condición de ejecución: Tienen que existir nombres de paquetes y desarrolladores**.**	
Entrada/ Pasos de ejecución: • El usuario inicia la corrida del método.	
Resultado esperado: Se obtiene una red colaborativa donde se puede visualizar la relación entre los desarrolladores de un mismo paquete del repositorio del sistema operativo libre.	
Evaluación de la prueba: Satisfactoria	

Como resultado de cada iteración se obtuvieron recomendaciones y no conformidades por parte de los clientes. Las no conformidades se clasificaron en significativas y no significativas en dependencia de la complejidad que tendría darle solución a la misma. También se tuvo en cuenta la afectación que supondría para el usuario no resolver dichas no conformidades. Después de cada iteración se arreglaron todas las no conformidades señaladas. A continuación, se representa en un gráfico las recomendaciones y no conformidades que arrojó cada iteración:

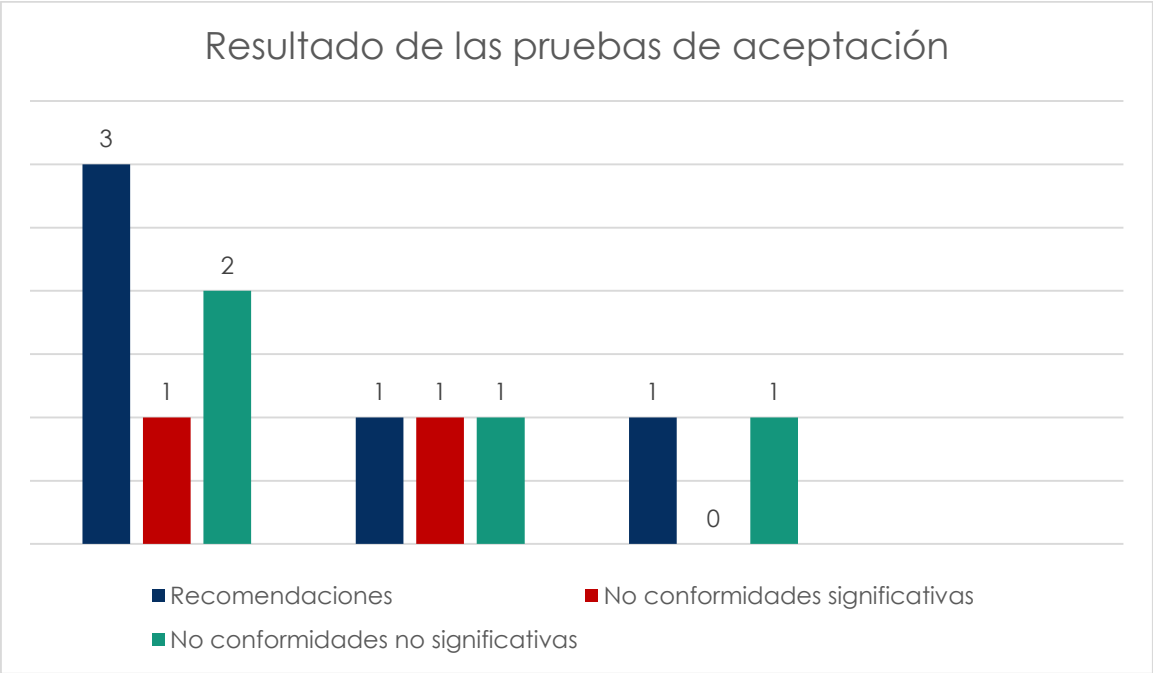

Figura 23. Gráfico de los resultados de las pruebas de aceptación (Fuente: Elaboración propia)

A continuación, se listan las recomendaciones detectadas en cada iteración.

Iteración 1:

1. Realizar uno a uno la descarga de los paquetes del repositorio.
2. Copiar hacia otro directorio los archivos extraídos para simplificar el trabajo.
3. Verificar si existe conexión antes de iniciar la descarga de los paquetes.

Iteración 2:

4. Eliminar cada paquete descargado después de extraer los changelogs.

Iteración 3:

5. Mostrar donde se encuentran los archivos que contienen las redes colaborativas.

A continuación, se listan las no conformidades significativas en cada iteración:

Iteración 1:

1. No se verifica que el paquete descargado se encuentre o no dañado antes de extraer el fichero de control de cambios.

Iteración 2:

2. No se tienen en cuenta todos los desarrolladores que aparecen en los ficheros de control de cambios independientemente que estos pueden encontrarse en áreas como la descripción de los cambios implementados por algún otro desarrollador.

A continuación, se listan las no conformidades no significativas en cada iteración:

Iteración 1:

1. No se tiene en cuenta si se presentan fallos en la red al realizarse la búsqueda y descarga de los archivos.
2. El usuario debe introducir la dirección del repositorio directamente en el código del método.

Iteración 3:

3. No se realiza una vista previa de la red colaborativa antes de generar los ficheros “gml” de la red.

Con el objetivo de verificar que se solucionaron todas las no conformidades detectadas se realizaron pruebas de regresión. Estas pruebas están enfocadas a comprobar que los problemas detectados en una iteración fueron resueltos correctamente por el equipo de proyecto, para poder pasar a la siguiente iteración, en la que se comprobará que no se introdujeron errores al corregir los encontrados.

3.4 Validación del método propuesto

Para realizar la validación del método propuesto se compararon los datos útiles que se encuentran en los changelogs con la información que se representa en la red colaborativa con el fin de verificar que esta última representa todos los nombres de desarrolladores y paquetes. Como caso de prueba se tomaron tres ficheros de control de cambios ficticios creados con el fin de implementar la validación del método propuesto.

Una vez creados los changelogs ficticios y agrupados en un mismo directorio, se procedió a cargar estos archivos utilizando el método implementado. Para verificar si el resultado correspondía con el esperado se analizó la red resultante. Se procedió a comprobar que todos los datos útiles se encontraran visibles y pudieran ser identificados fácilmente. También se tuvo en cuenta que las relaciones entre los paquetes y desarrolladores fuera correcta. Los nombres de los paquetes y desarrolladores para cada fichero de control de cambios se muestran en la siguiente tabla:

Tabla 4. Nombre de desarrolladores y paquetes para la prueba de validación (Fuente: Elaboración propia)

Paquetes	Nombres
Paqueteuno	Nombreuno Apellidouno Nombredos Apellidodos Nombretres Apellidotres
Paquetedos	Nombrecuatro Apellidocuatro Nombrecinco Apellidocinco Nombreuno Apellidouno
Paquetetres	Nombreseis Apellidoseis Nombresiete Apellidosiete Nombretres Apellidotres

La siguiente figura muestra la red colaborativa que resulto de representar los nombres de desarrolladores de los changelogs ficticios (ver tabla 4) relacionados por los paquetes a los que corresponden:

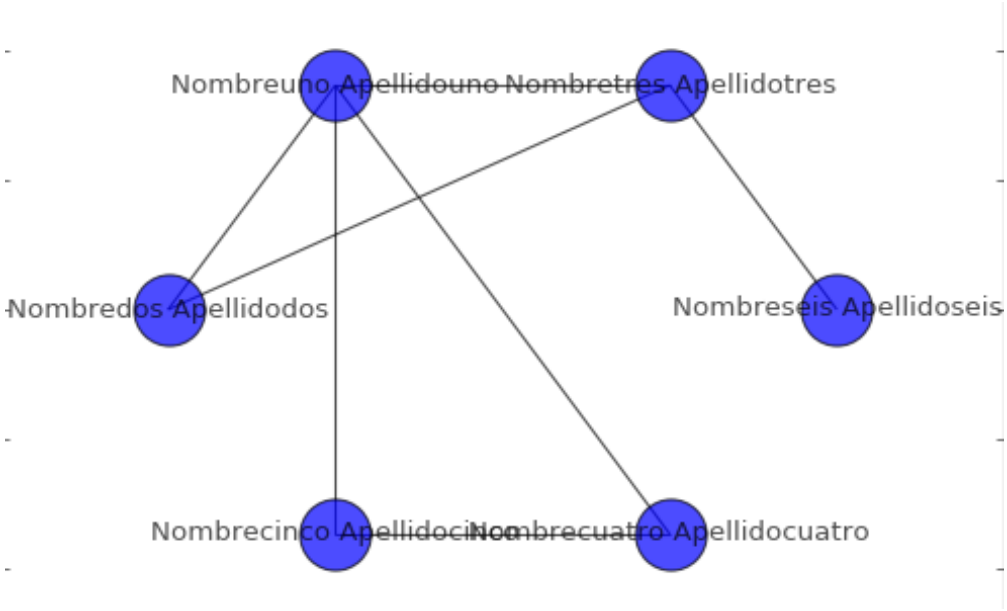

Figura 24. Red colaborativa de relaciones entre desarrolladores ficticios (Fuente: Elaboración propia)

También se representó la red colaborativa como un grafo bipartido atendiendo a las relaciones que se establecen entre los paquetes y los desarrolladores de los mismos:

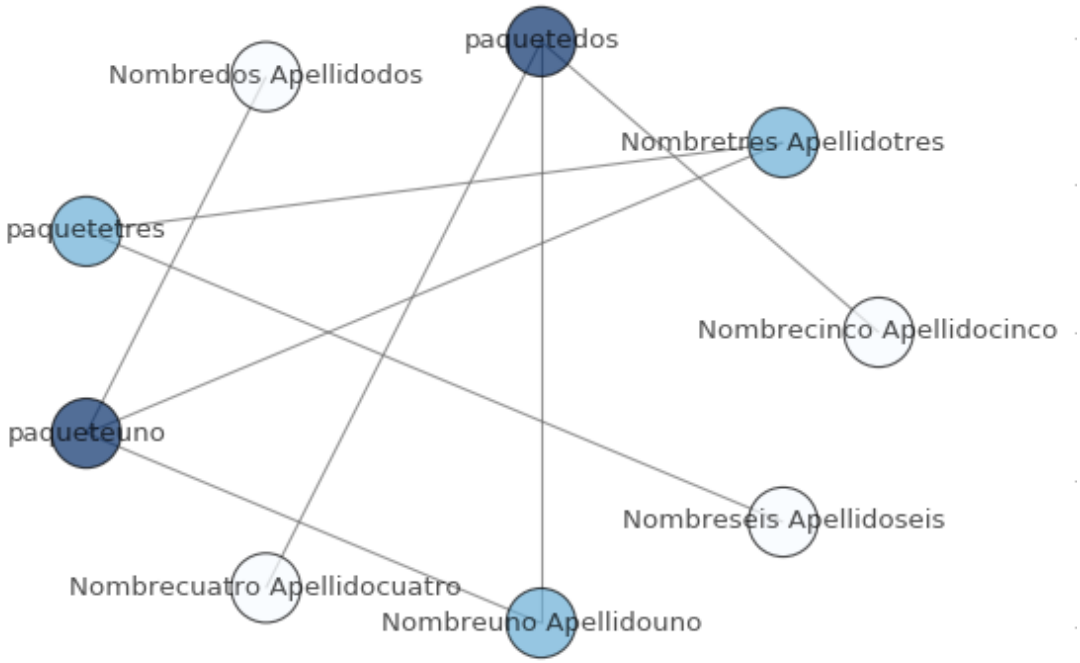

Figura 25. Red colaborativa de relaciones entre los paquetes ficticios y sus desarrolladores (Fuente: Elaboración propia)

3.5 Validación de la solución en función del tiempo

Se realizaron pruebas en función del tiempo para verificar que el proceso de extracción de los ficheros de control de cambios se aceleraba considerablemente empleando el método implementado. Se tomó una muestra de cinco personas familiarizadas con el trabajo en el sistema operativo Windows 10 específicamente en la descarga de archivos, manejo de herramientas de descompresión de archivos y gestión de ficheros. Teniendo en cuenta que el número de archivos que contiene un repositorio de un sistema operativo libre es elevado, se seleccionó una muestra de trece archivos.

Se empleó el navegador Mozilla Firefox en su versión 59.0 para acceder y descargar archivos desde el repositorio de un sistema operativo libre de la distribución Ubuntu. Dicho repositorio se encuentra en los servidores de la Universidad de las Ciencias Informáticas y puede accederse mediante la dirección http://ubuntu.uci.cu/ubuntu/pool/main/a/a11y-profile-manager/. Para desempaquetar o descomprimir los paquetes ".deb" se hizo uso de la herramienta AnyToIso y para extraer los archivos internos se trabajó con Winrar.

La prueba consistió en que las personas seleccionadas descargaran los paquetes de la dirección URL especificada anteriormente. Posteriormente, debían extraer el changelog de cada uno de los trece archivos de forma manual y copiarlos hacia un directorio. Se cronometró el tiempo que tardaron estas personas y posteriormente se analizaron los resultados alcanzados con los generados automáticamente por el método propuesto. El siguiente gráfico muestra los resultados alcanzados en función del tiempo:

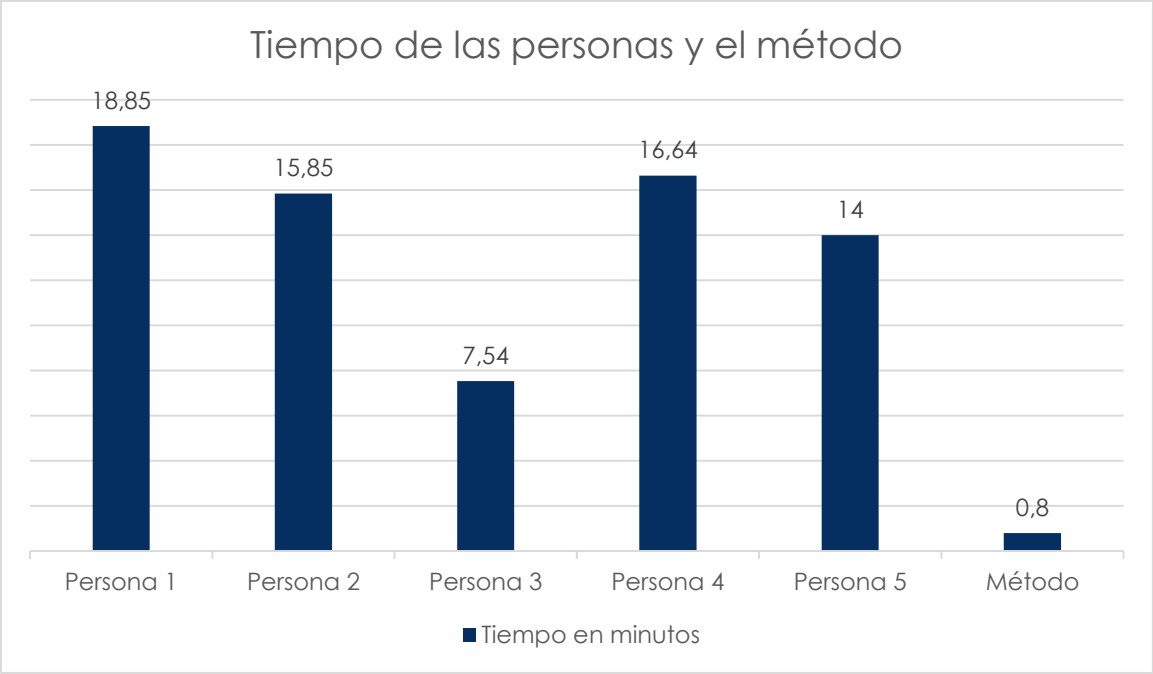

Figura 26. Gráfico de los tiempos personas - método implementado (Fuente: Elaboración propia)

3.6 Conclusiones del capítulo

El proceso de validación y pruebas de software a la solución propuesta se desarrolló sin inconvenientes. Luego de realizadas las pruebas de software y de corregirse las no conformidades que fueron detectadas, puede concluirse que el método se encuentra listo para su utilización. Las validaciones realizadas prueban la calidad de la solución desarrollada.

CONCLUSIONES GENERALES

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

RECOMENDACIONES

Una vez cumplido el objetivo de la presente investigación se recomienda:

1. Implementar y probar si el modelo en paralelo mejora el proceso de descarga de los ficheros de control de cambios.
2. Implementar una suite para el trabajo con el método desarrollado donde puedan realizarse por separado los principales procesos que lo componen (extracción de los changelogs, creación de la red colaborativa, detección de comunidades y actores más influyentes) e integrarlo con la consola de Gephi para el análisis de la red sin necesidad de abrir esta herramienta.
3. Realizar un estudio de la información que pudiera obtenerse del análisis de las redes obtenidas a partir de los paquetes de los repositorios en las diferentes versiones de un mismo sistema operativo.

REFERENCIAS BIBLIOGRÁFICAS

AGGARWAL, C.C., 2011. *An Introduction to Social Network Data Analytics* [en línea]. New York, NY, USA: Springer, Boston, MA. [Consulta: 13 enero 2018]. ISBN 978-1-4419-8462-3. Disponible en: https://link.springer.com/chapter/10.1007/978-1-4419-8462-3_1.

AGUIRRE, J.-L., 2011. Introducción al análisis de redes sociales. *Documentos de Trabajo del Centro Interdisciplinario para el Estudio de Políticas Públicas*, vol. 82, pp. 1-59.

ANACONDA, INC, 2018. Anaconda. *What is Anaconda?* [en línea]. [Consulta: 9 enero 2018]. Disponible en: https://-www.anaconda.com/-what-is-anaconda/.

AURORA, P.K., 2007. *Multi-level graph partitioning*. S.l.: University of Florida.

BARAN, P., 1964. On distributed communications networks. *IEEE transactions on Communications Systems*, vol. 12, no. 1, pp. 1-9.

BASTIAN, M., HEYMANN, S. y JACOMY, M., 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks. *AAAI Publications, Third International AAAI Conference on Weblogs and Social Media* [en línea]. S.l.: s.n., pp. 2. [Consulta: 9 enero 2018]. Disponible en: https://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

BELTRÁN, P., EDUARDO, J., VALERIO UREÑA, G. y RODRÍGUEZ-ACEVES, L., 2015. Análisis de redes sociales para el estudio de la producción intelectual en grupos de investigación. *Perfiles educativos*, vol. 37, no. 150, pp. 124-142. ISSN 0185-2698.

BORGATTI, S.P. y HALGIN, D.S., 2011. On Network Theory. *Organization Science*, vol. 22, no. 5, pp. 1168-1181. ISSN 1047-7039. DOI 10.1287/orsc.1100.0641.

BOS, J., BRENNENREADTS, R., CAMPBELL, S.W., AZEEZ ERUMBAN, A., GREENSTEIN, S., GUPTA, A., GODOY-ETCHEVERRY, S., HELSPER, E., PAGER, C.-J., KATZ, R.L., KWAK, N., LING, R., MANOLE, V., MCDEVITT, R., SHULTZ, I. y TE VELDE, R., 2011. *Un mundo conectado: las TIC transforman sociedades, culturas y economías*. S.l.: Fundación Telefónica. ISBN 84-08-10329-6.

CHACON, S. y STRAUB, B., 2004. *Pro Git*. 2da. California, Estados Unidos: Apress.

CHERVEN, K., 2013. *Network Graph Analysis and Visualization with Gephi*. S.l.: Packt Publishing Ltd. ISBN 978-1-78328-014-8.

FAYYAD, U., PIATETSKY-SHAPIRO, G. y SMYTH, P., 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, vol. 17, no. 3, pp. 37. ISSN 0738-4602.

FERNÁNDEZ RODRÍGUEZ, A.W. y HIDALGO RUIZ, J.J., 2014. *Análisis de influencia social en el contexto universitario.* [en línea]. Trabajo final presentado en opción al título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Universidad de las Ciencias Informáticas. Disponible en:

https://repositorio_institucional.uci.cu/jspui/bitstream/ident/9007/2/TD_07391_14.pdf.

FORTUNATO, S., 2010. Community detection in graphs. *Physics Reports*, vol. 486, no. 3, pp. 75-174. ISSN 0370-1573. DOI https://doi.org/10.1016/j.physrep.2009.11.002.

GEPHI, 2017. Gephi - The Open Graph Viz Platform. [en línea]. [Consulta: 9 enero 2018]. Disponible en: https://gephi.org/.

GONZÁLEZ DUQUE, R., 2015. *Python para todos* [en línea]. España: s.n. Disponible en: http://mundogeek.net/tutorial-python/.

GRAND JEAN, M., 2018. Free and Open-Source Tool for Social Network Analysis. *Social Network Visualizer* [en línea]. Disponible en: http://socnetv.org/.

GROSS, J.L., YELLEN, J. y ZHANG, P., 2013. *Handbook of graph theory*. 2da. S.l.: Chapman and Hall/CRC. ISBN 1-4398-8019-0.

HERNÁNDEZ ORALLO, J., RAMÍREZ QUINTANA, M.J. y FERRI RAMÍREZ, C., 2004. *Introducción a la Minería de Datos*. Madrid, España: Pearson Educación. ISBN 84-205-4091-9.

IEEE, 2008. IEEE Standard for Software and System Test Documentation. *IEEE Std 829-2008*, pp. 1-150. ISSN 978-0-7381-5746-7. DOI 10.1109/IEEESTD.2008.4578383.

INTERNETBLOG.ORG.UK, 2010. What is a Linux Package? [en línea]. [Consulta: 5 marzo 2018]. Disponible en: https://www.internetblog.org.uk/post/1520/what-is-a-linux-package/.

JETBRAINS, 2017. PyCharm: Python IDE for Professional Developers by JetBrains. [en línea]. [Consulta: 12 noviembre 2017]. Disponible en: https://www.jetbrains.com/pycharm/.

KERNIGHAN, B.W. y LIN, S., 1970. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, vol. 49, no. 2, pp. 291-307.

KOURTELLIS, N., MORALES, G.D.F. y BONCHI, F., 2014. Scalable Online Betweenness Centrality in Evolving Graphs. En: arXiv: 1401.6981, *arXiv:1401.6981 [cs]* [en línea], [Consulta: 24 marzo 2018]. Disponible en: http://arxiv.org/abs/1401.6981.

KUZ, A., FALCO, M. y GIANDINI, R., 2016. Análisis de redes sociales: un caso práctico. *Computación y Sistemas*, vol. 20, no. 1, pp. 89-106.

LARMAN, C., 2003. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2da. USA: Prentice Hall.

LOZANO, M., GARCÍA-MARTÍNEZ, C., RODRÍGUEZ, F.J. y TRUJILLO, H.M., 2016. Optimización de ataques a redes complejas mediante un algoritmo de colonias de abejas artificiales. [en línea], Disponible en: http://helvia.uco.es/bitstream/handle/10396/15775/articulo-Actas-maeb2016.pdf?sequence=1&isAllowed=y.

MATPLOTLIB.ORG, 2017. Matplotlib: Python plotting — Matplotlib 2.1.0 documentation. [en línea]. [Consulta: 10 noviembre 2017]. Disponible en: https://matplotlib.org/.

MCKINNEY, W., 2012. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. S.l.: O'Reilly Media, Inc. ISBN 1-4493-2361-8.

MONSALVE MORENO, M., 2008. Análisis de Redes Sociales: un Tutorial. , pp. 6.

NEWMAN, M.E., 2004. Fast algorithm for detecting community structure in networks. *Physical review E*, vol. 69, no. 6, pp. 066133.

NEWMAN, M.E. y GIRVAN, M., 2004. Finding and evaluating community structure in networks. *Physical review E*, vol. 69, no. 2, pp. 026113.

NEWMAN, M.E. y PARK, J., 2003. Why social networks are different from other types of networks. *Physical Review E*, vol. 68, no. 3, pp. 036122.

NOOY, W. de, MRVAR, A. y BATAGELJ, V., 2011. *Exploratory Social Network Analysis with Pajek*. S.l.: Cambridge University Press. ISBN 978-1-139-50108-8.

PCC, 2016. *Actualización de los Lineamientos de la Política económica y social del Partido y la Revolución para el periodo 2016-2021 aprobados en el 7mo. Congreso del Partido en abril de 2016 y por la Asamblea Nacional de Poder Popular en Julio de 2016* [en línea]. 2016. S.l.: s.n. Disponible en: http://www.granma.cu/file/pdf/gaceta/01Folleto.Lineamientos-4.pdf.

PIERRA FUENTES, A., 2011. *Nova, distribución cubana de GNU/Linux: reestructuración estratégica de su proceso de desarrollo* [en línea]. Trabajo final presentado en opción al título de Máster en Informática Aplicada. La Habana, Cuba: Universidad de las Ciencias Informáticas. Disponible en: http://catalogoenlinea.uci.cu/cgi-bin/koha/opac-detail.pl?biblionumber=11161.

PRESSMAN, R.S., 2010. *Ingenieria del Software. Un Enfoque Práctico*. 7ma. S.l.: McGraw-Hill. ISBN 978-607-15-0314-5.

PYTHON SOFTWARE FOUNDATION, 2018a. 7.2. re — Regular expression operations — Python 2.7.14 documentation. [en línea]. [Consulta: 26 febrero 2018]. Disponible en: https://docs.python.org/2/library/re.html.

PYTHON SOFTWARE FOUNDATION, 2018b. 21.6. urllib.request — Extensible library for opening URLs — Python 3.6.5 documentation. [en línea]. [Consulta: 17 mayo 2018]. Disponible en: https://docs.python.org/3/library/urllib.request.html#module-urllib.request.

PYTHON SOFTWARE FOUNDATION, 2018c. 21.8. urllib.parse — Parse URLs into components — Python 3.6.5 documentation. [en línea]. [Consulta: 17 mayo 2018]. Disponible en: https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse.

RICHARDSON, L., 2015. Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation. [en línea]. [Consulta: 17 mayo 2018]. Disponible en: http://beautiful-soup-4.readthedocs.io/en/latest/.

RIONDATO, M. y UPFAL, E., 2016. ABRA: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. S.l.: ACM, pp. 1145-1154. ISBN 1-4503-4232-9.

SILBERSCHATZ, A., GALVIN, P.B. y GREG, G., 2006. *Fundamentos de Sistemas Operativos*. 7. España: McGraw-Hill. ISBN 84-481-4641-7.

SMITH, M., 2017. NodeXL: Network Overview, Discovery and Exploration for Excel. *CodePlex* [en línea]. [Consulta: 12 enero 2018]. Disponible en: http://nodexl.codeplex.com/Wikipage?ProjectName=nodexl.

STALLMAN, R., 2004. *Software libre para una sociedad libre*. S.l.: Madrid: Traficantes de Sueños, 2004. ISBN 84-933555-1-8.

STALLMAN, R.M., 2002. What is free software. *Free Society: Selected Essays of*, vol. 23.

SUEUR, C., DENEUBOURG, J.-L. y PETIT, O., 2012. From Social Network (Centralized vs. Decentralized) to Collective Decision-Making (Unshared vs. Shared Consensus). *PLOS ONE* [en línea], Disponible en: https://doi.org/10.1371/journal.pone.0032566.

TAPIA CHIROLDES, A., 2014. *Herramienta para gestionar los repositorios locales de software en Nova*. Trabajo final presentado en opción al título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Universidad de las Ciencias Informáticas.

TRUDEAU, R.J., 2013. *Introduction to graph theory*. S.l.: Courier Corporation. ISBN 0-486-31866-4.

TUCKER, A.B., 2004. *Computer Science Handbook, Second Edition*. S.l.: CRC Press. ISBN 978-0-203-49445-5.

VAN DER AALST, W.M.P., REIJERS, H.A. y SONG, M., 2005. Discovering Social Networks from Event Logs. *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 6, pp. 549–593. ISSN 0925-9724, 1573-7551. DOI 10.1007/s10606-005-9005-9.

VISUAL PARADIGM, 2018. Visual Paradigm Product Overview. [en línea]. [Consulta: 27 abril 2018]. Disponible en: https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

VON LUXBURG, U., 2007. A tutorial on spectral clustering. *Statistics and computing*, vol. 17, no. 4, pp. 395-416.

WALTER, B., BALA, K., KULKARNI, M. y PINGALI, K., 2008. Fast agglomerative clustering for rendering. *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on*. S.l.: IEEE, pp. 81-86. ISBN 1-4244-2741-X.

WANG, M., WANG, C., YU, J.X. y ZHANG, J., 2015. Community Detection in Social Networks: An In-depth Benchmarking Study with a Procedure-oriented Framework. *Proc. VLDB Endow.*, vol. 8, no. 10, pp. 998–1009. ISSN 2150-8097. DOI 10.14778/2794367.2794370.

WASSERMAN, S. y FAUST, K., 1994. *Social Network Analysis: Methods and Applications* [en línea]. 1ra. Reino Unido: Cambridge University Press. Structural Analysis in the Social Sciences, 8. ISBN 978-0-521-38707-1. Disponible en: http://www.google.com.cu/books?hl=en&lr=&id=CAm2DpIqRUIC&oi=fnd&pg=PR21&dq=stanley+wasserman&ots=HvHpAfWIR9&sig=XZaTkWrq0ehCeond9h9bz1VE2C8&redir_esc=y#v=onepage&q&f=false.

ANEXOS

Anexo 1. Medidas de centralidad que se pueden aplicar a la red

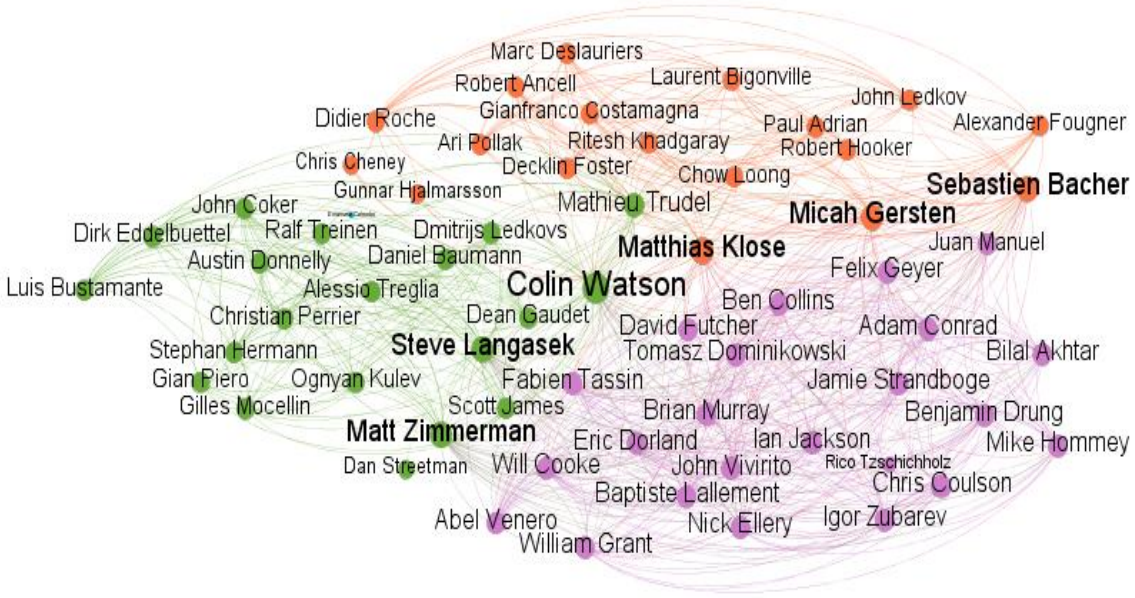

Figura 27. Red colaborativa representada por centralidad de cercanía (Fuente: Elaboración propia)

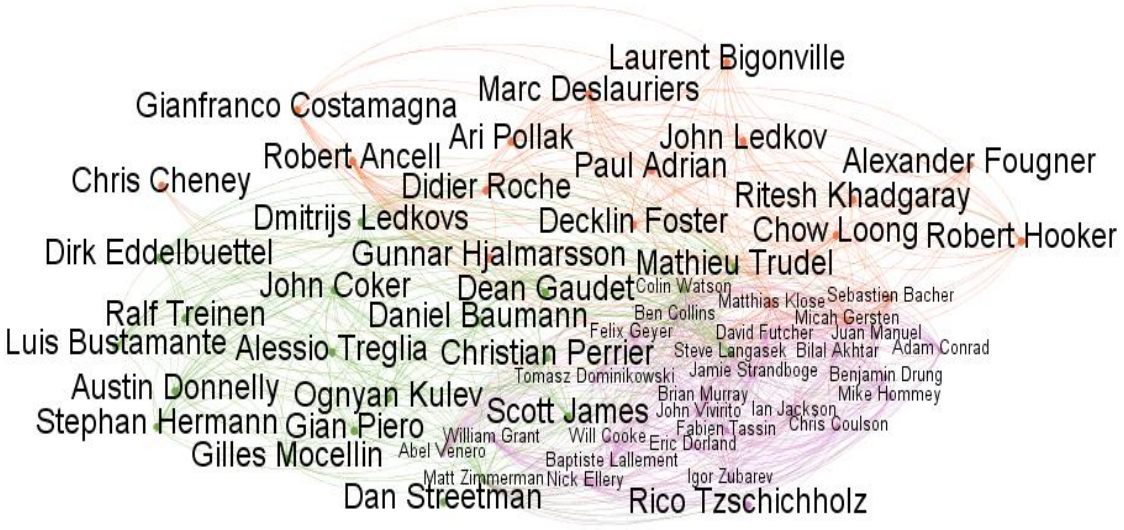

Figura 28. Red colaborativa representada por centralidad de grado (Fuente: Elaboración propia)

Anexo 2. Tabla de datos que genera Gephi para una red colaborativa

Tabla 5. Tabla de datos con la información de una red colaborativa (Fuente: Elaboración propia)

Id	Label	Interval	pakage	Eccentricity	Closeness Centrality	Harmonic Closeness Centrality	Betweenness Centrality
0	Luis Bustama...		acct	3.0	0.582524	0.647222	0.0
1	Dmitrijs Ledk...		openoffice	3.0	0.606061	0.680556	39.8
2	Baptiste Lalle...		firefox	2.0	0.645161	0.725	0.125
3	Abel Venero		firefox	2.0	0.638298	0.716667	0.0
4	Chris Coulson		firefox	2.0	0.652174	0.733333	59.125
5	David Futcher		firefox	2.0	0.645161	0.725	0.125
6	Chow Loong		pidgin	3.0	0.576923	0.638889	0.0
7	Tomasz Domi...		firefox	2.0	0.645161	0.725	0.125
8	Christian Per...		acct	3.0	0.582524	0.647222	0.0
9	Dirk Eddelbu...		acct	3.0	0.582524	0.647222	0.0
10	Emanuele Col...		dukto	0.0	0.0	0.0	0.0
11	Jamie Strand...		firefox	2.0	0.645161	0.725	0.125
12	Benjamin Drung		firefox	2.0	0.645161	0.725	0.125
13	Didier Roche		pidgin	3.0	0.594059	0.663889	19.333333
14	Alexander Fo...		pidgin	3.0	0.576923	0.638889	0.0
15	Ian Jackson		firefox	2.0	0.645161	0.725	0.125
16	Ari Pollak		pidgin	3.0	0.576923	0.638889	0.0
17	Dan Streetman		vlan	3.0	0.491803	0.5	0.0
18	Ognyan Kulev		acct	3.0	0.582524	0.647222	0.0
19	Brian Murray		firefox	2.0	0.645161	0.725	0.125
20	Daniel Baumann		acct	3.0	0.582524	0.647222	0.0
21	John Vivirito		firefox	2.0	0.645161	0.725	0.125
22	Ben Collins		firefox	2.0	0.645161	0.725	0.125
23	Mike Hommey		firefox	2.0	0.645161	0.725	0.125
24	Bilal Akhtar		firefox	2.0	0.645161	0.725	0.125
25	Decklin Foster		pidgin	3.0	0.576923	0.638889	0.0
26	Austin Donnelly		acct	3.0	0.582524	0.647222	0.0
27	Sebastien Ba...		pidgin	2.0	0.759494	0.841667	83.841667
28	Felix Geyer		firefox	2.0	0.645161	0.725	0.125
29	Stephan Her...		acct	3.0	0.582524	0.647222	0.0
30	Colin Watson		vlan	2.0	0.952381	0.975	357.725
31	Ralf Treinen		acct	3.0	0.582524	0.647222	0.0
32	Igor Zubarev		firefox	2.0	0.645161	0.725	0.125
33	Mathieu Trudel		pidgin	3.0	0.689655	0.780556	106.933333
34	Paul Adrian		pidgin	3.0	0.576923	0.638889	0.0
35	Scott James		acct	3.0	0.582524	0.647222	0.0
36	Juan Manuel		firefox	2.0	0.625	0.7	0.0
37	Robert Hooker		pidgin	3.0	0.576923	0.638889	0.0
38	Gilles Mocellin		acct	3.0	0.582524	0.647222	0.0
39	Ritesh Khadg...		pidgin	3.0	0.576923	0.638889	0.0
40	Will Cooke		firefox	2.0	0.638298	0.716667	0.0
41	Adam Conrad		firefox	2.0	0.645161	0.725	0.125
42	Gianfranco C...		pidgin	3.0	0.576923	0.638889	0.0
43	Fabien Tassin		firefox	2.0	0.645161	0.725	0.125
44	Matthias Klose		pidgin	2.0	0.789474	0.866667	156.925
45	John Ledkov		pidgin	3.0	0.576923	0.638889	0.0
46	Gian Piero		acct	3.0	0.582524	0.647222	0.0
47	Alessio Treglia		acct	3.0	0.582524	0.647222	0.0
48	Matt Zimmer...		firefox	2.0	0.779221	0.858333	127.175
49	Steve Langasek		firefox	2.0	0.779221	0.858333	127.175
50	William Grant		firefox	2.0	0.638298	0.716667	0.0
51	Nick Ellery		firefox	2.0	0.645161	0.725	0.125
52	Dean Gaudet		acct	3.0	0.582524	0.647222	0.0
53	John Coker		acct	3.0	0.582524	0.647222	0.0
54	Rico Tzschich...		firefox	3.0	0.397351	0.419444	0.0
55	Micah Gersten		pidgin	2.0	0.759494	0.841667	83.841667
56	Laurent Bigo...		pidgin	3.0	0.576923	0.638889	0.0
57	Gunnar Hjalm...		openoffice	3.0	0.508475	0.527778	0.0
58	Marc Deslauri...		pidgin	3.0	0.576923	0.638889	0.0
59	Chris Cheney		openoffice	3.0	0.508475	0.527778	0.0
60	Eric Dorland		firefox	2.0	0.645161	0.725	0.125
61	Robert Ancell		pidgin	3.0	0.576923	0.638889	0.0

Anexo 3. Vista de funcionamiento del método implementado

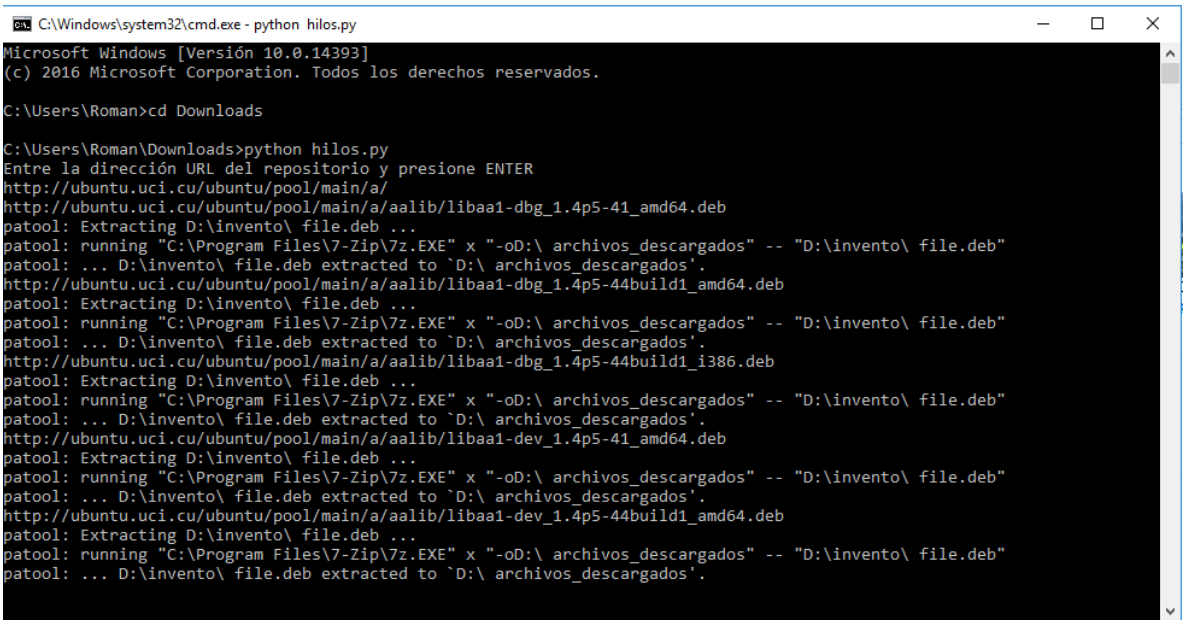

Figura 29. Ejecución del método implementado (Fuente: Elaboración propia)