UNIDAD 3: Usando Ingeniería de Software Libre: Primera Fase

Objetivos de la Unidad:

Utilizar de forma eficiente la primera fase de la Ingeniería de software libre en los procesos de un proyecto de desarrollo

Temas:

- Extracción de información del proyecto de software libre: Primera Fase
  - Lo que el proyecto necesita
    1. Código fuente
    2. CODD
    3. SLOCcount
    4. Métricas de software
  - Intercambio directo de información
    1. Listas de correo
    2. IRC
  - Herramientas de desarrollo distribuido
    1. Sistema de control de versiones
    2. Sistema de seguimiento errores
- Otras herramientas
  - Parser y otras herramientas
  - Empaquetadores
- Otras fuentes de información
- Formato de intercambio de información
  - Conversores a y de XML

Extracción de información del proyecto de software libre: Primera Fase

Lo que el proyecto necesita

El primer paso incluye agrupar, ordenar y analizar de forma conveniente el código fuente y los flujos de información existentes en los proyectos de software libre. La finalidad principal es conseguir que todo esto se haga de la forma más automática posible. De hecho, se pretende recabar todo tipo de información para que pueda ser analizada y estudiada detenidamente con posterioridad. Como puede ver, se trata de un proceso iterativo, ya que los resultados de los primeros análisis indicarán por dónde seguir buscando y cuáles deben ser los siguientes pasos lógicos dentro del estudio del software libre. A continuación, se muestran las diferentes fuentes que se pueden analizar, así como las diversas herramientas que existen para obtener resultados a partir de esas fuentes.

1. Código Fuente

El código fuente es, con diferencia, el que mayor información contiene en cuanto al desarrollo de proyectos de software libre se refiere. De éste se pueden extraer no sólo los parámetros globales como el tamaño, el número de archivos, sino que puede ser investigado con la finalidad de encontrar parámetros de participación como el número de desarrolladores; de programación como el lenguaje de programación, además de la posibilidad de utilizar diferentes métricas de programación; de líneas de código tanto lógicas como físicas; el número de comentarios, etc. Una de las primeras aproximaciones existentes hoy en día es el cálculo del número de líneas físicas de proyectos de software libre y el uso del modelo COCOMO (clásico) para obtener resultados con respecto al tiempo, al coste y a los recursos humanos necesarios para su desarrollo. Es evidente que durante este primer análisis se encuentra en una fase bastante primitiva, pero la correlación con otras fuentes permitirá mejorar y/o adaptar los resultados en el futuro.

2. CODD

El CODD es una herramienta diseñada por Vipul Ved Prakash y Rishab Aiyer Ghosh con la que se puede analizar el código fuente de los paquetes de software libre y se le puede asignar las cuotas de autoría (en bytes) a los que han participado en el desarrollo. También ha sido diseñada e implementada para extraer y resolver dependencias entre paquetes tal y como se verá a continuación.

El CODD cuenta con una serie de procesos que deben ser ejecutados de forma consecutiva, donde los resultados se guardan en archivos denominados codds, en minúsculas. Para cada paquete de software libre se generará un codd, que contendrá información sobre el mismo, ya sea extraída del propio paquete o de la correlación con codds de otros paquetes.

Al final del proceso, cada codd debería contener la siguiente información: nombre del paquete, generalmente más versión (p.ej. evolution-0.13) créditos de autoría (y bytes de contribución) archivos de código,archivos de documentación, interfaces código compartido implementaciones externas o no resueltas implementaciones resueltas metainformación.

Es muy importante notar que los codds se crean durante la primera iteración con la información que extraen directamente de los paquetes. Como todos los resultados de los procesos intermedios se guardan en el mismo codd, no se puede saber a simple vista en qué paso dentro del algoritmo de CODD se encuentra. La única forma de saber esto es al inspeccionar el contenido del codd.

A continuación se muestra el proceso que sigue CODD para obtener estos datos:

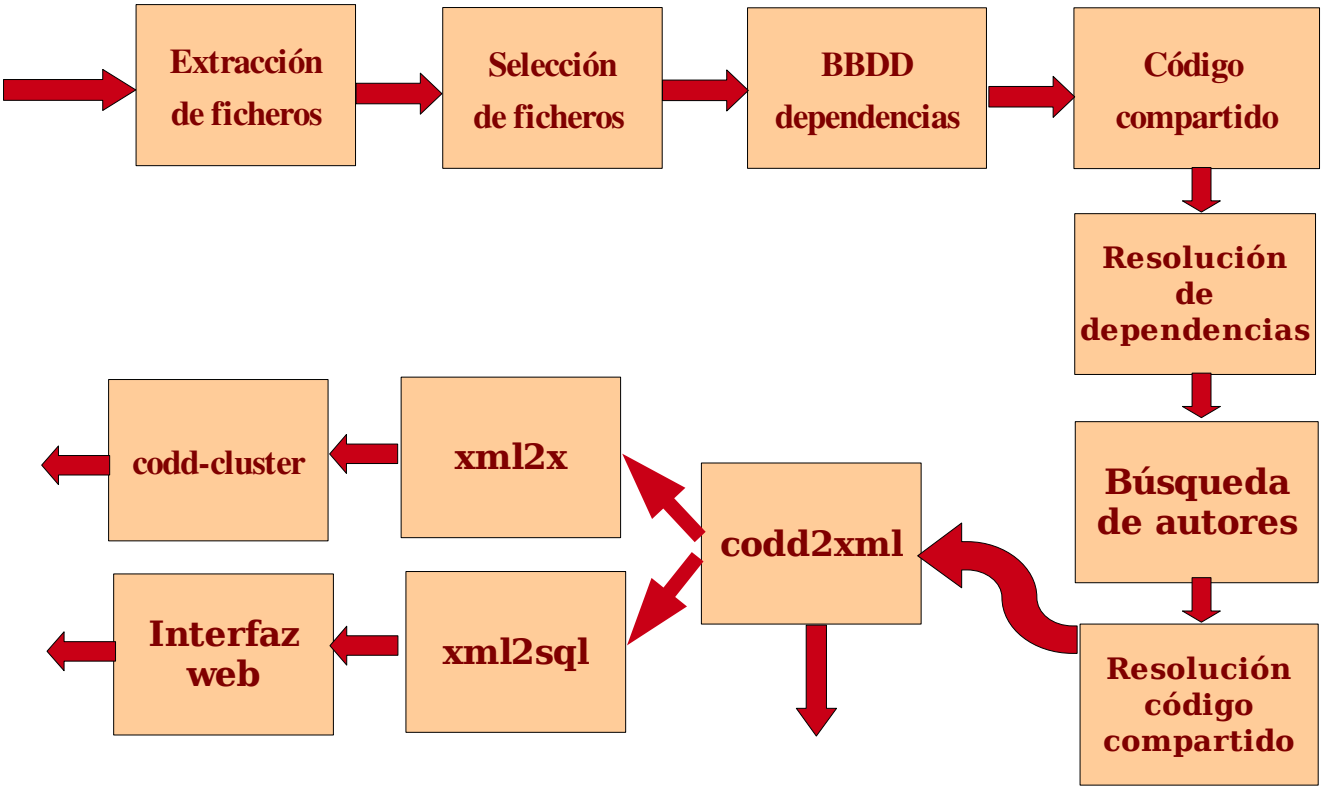

Figura 3.1: Proceso de obtención de datos en CODD

Extracción de archivos: en esta etapa la subrutina init toma el paquete, o paquetes, que se le ha pasado por la línea de instrucciones, lo descomprime y explota si es necesario, e intenta identificar recursivamente el tipo de archivos que contiene el paquete.

Selección de archivos: durante esta etapa se toman los archivos de código, de documentación, las interfaces e implementaciones no resueltas junto con su tamaño en bytes, su suma MD5 y su ruta relativa dentro del paquete. Esto se lleva a cabo al comparar las extensiones de los archivos del paquete. El CODD cuenta con una serie de arrays en los que se encuentran almacenadas las extensiones que pueden tener los archivos de código (por ejem. ".c" para C o ".pl" para Perl), de documentación, etc. El CODD almacena como interfaces aquéllos archivos ".h" que tienen un archivo ".c" en el paquete, el algoritmo que se utiliza aquí depende parcialmente del lenguaje de programación que se esté analizando. Las llamadas a interfaces en los archivos de código, por ejem. ".c" para C, que no tengan su correspondiente interfaz en el paquete, por ejem. ".h" para C, pasarán a englobar la categoría de implementaciones no resueltas, que en futuros pasos dará pie a la resolución de dependencias.

Base de datos de dependencias: durante el tercer paso se crean dos bases de datos para encontrar código compartido y las dependencias. Durante la primera, de nombre

'codefile_signatures', se almacenan todas las sumas MD5 de los archivos de código. Para ello, CODD se recorre todos los codds, mira en las entradas correspondientes a los archivos de código y añade un par, suma MD5 y nombre de archivo, paquete. Del mismo modo insertará pares para los interfaces en la base de datos de nombre 'interfaces'. En este caso, los pares son del tipo, nombre de archivo, paquete.

Código compartido: Durante el siguiente paso, el CODD recorre otra vez todo los codds y mira si los archivos de código aparecen más de una vez en la base de datos (en realidad, si coinciden su nombre y su suma MD5). Si esto ocurre, el archivo se encuentra como mínimo en dos paquetes, por lo que se añadirá en la sección del codd dedicada al código compartido ('shared') la siguiente información por cada paquete que también contenga el archivo: (archivo de código, ruta, MD5, tamaño) => nombre del paquete.

Resolución de dependencias: el CODD llevará a cabo un proceso parecido al punto anterior para la resolución de dependencias. Buscará las implementaciones no resueltas en los codds y comparará sus sumas MD5 con las que existen en la base de datos de interfaces. De encontrar alguna coincidencia, la interfaz se borrará de la sección de interfaces no resueltas ('unresolved interfaces') a la de interfaces resueltas ('resolved'). Además, se proporcionará una lista con todos los paquetes en los que se encuentra implementada.

Búsqueda de autores: durante esta etapa es cuando CODD realiza la tarea que ha hecho que sea conocido: la búsqueda de autoría ('ownergrep'). Para llevarlo a cabo, recorre todos los archivos de código y documentación cuya ruta se encuentra almacenada en cada codd y extrae, siguiendo ciertos algoritmos de comparación de patrones, a los autores. La información sobre los autores se almacenan en la sección de créditos ('credits') del codd.

Resolución de código compartido: durante la sección del código compartido ('shared') del codd se tienen todavía archivos y una lista de paquetes que también lo tienen. Como este archivo sólo puede ser asignado a un paquete, lo que hace CODD es buscar por su autor ('ownergrep') en el propio archivo y asignarlo al paquete del cual el autor es el autor principal.

En los últimos bloques de la figura se muestra que los codds deberán ser entonces transformados a un formato intermedio e independiente en XML a partir del cual se pueden realizar ya las técnicas de análisis, correlación o clustering como se verá más adelante. Como se puede ver de la figura, existe una herramienta de transformación de codds a SQL, que realiza también tareas de normalización de las tablas. Esta herramienta ha sido concebida para crear CODDWeb, una interfaz Web para poder visualizar los resultados de CODD a través del navegador.

El CODD es una herramienta muy potente, aunque tiene algunos puntos flacos. El más importante es que todavía no tiene ninguna forma de agrupar las diferentes formas en las que aparece un autor. Por ejemplo, Miguel de Icaza aparece varias veces con diferentes nombres o direcciones de correo. En este caso, aglutinar todas sus referencias en una única sería lo más correcto. En este sentido, se está creando una base de datos con relaciones 1 a N por autor.

Por otro lado, el proceso de ejecución de CODD no es lo más simple que podría ser. Le falta un buen sistema de configuración, ya que se debe ejecutar como superusuario y la secuencia de procesos que se debe ejecutar no está agrupada bajo un único guión de shell, sino que de debe correr de forma manual.

3. SCLOCcount

SLOCcount, es una herramienta creada por David A. Wheeler, ésta cuenta el número de líneas físicas ofreciendo como resultado básicamente el lenguaje de programación utilizado. También, utiliza el modelo COCOMO clásico para, a partir de una serie parámetros y algoritmos preconfigurados, obtener el coste, los plazos y los recursos humanos necesarios para haber realizado una única entrega del software.

El algoritmo utilizado por SLOCcount cuenta con una serie de fases: en un primer paso SLOCCount busca los archivos de código por su extensión dentro del árbol de archivos del proyecto. Al encontrar un archivo con código, utiliza una serie de métricas para determinar si de verdad lo que contiene el archivo es código y si se encuentra en el lenguaje de programación que se determina de su extensión. En caso de ser así, contará las líneas de código que no sean comentarios ni espacios en blanco, líneas físicas, e incrementará el contador para dicho lenguaje en su número.

A continuación se presentan los datos que retorna SLOCcount: nombre del proyecto; número de líneas del proyecto; número de líneas en un lenguaje de programación; el tiempo estimado de esfuerzo de desarrollo (COCOMO básico); la estimación del tiempo de desarrollo (COCOMO básico); los número estimado de desarrolladores (COCOMO básico); la estimación del coste de desarrollo (COCOMO básico).

En un sentido estricto, las estimaciones realizadas a partir de COCOMO básico no deberían corresponder a la fase de extracción de datos, sino a una posterior de análisis.

SLOCCount retorna, hoy en día, los resultados en texto plano, aunque existe la posibilidad de que los devuelva entre tabuladores para que puedan ser introducidos en una base de datos. Existe una herramienta, de nombre sloc2html, la cual permite transformar los resultados a páginas HTML muy vistosas.

4. Métricas de software

Con la disponibilidad de todo el código se permite que se le puedan pasar todo tipo de métricas de software, como por ejemplo el cálculo de puntos de función. Este aspecto todavía no ha sido estudiado con mucho detenimiento, pero parece que de la información que se puede obtener por medio de estos métodos se podrán realizar comparaciones entre proyectos, lenguajes de programación, etc. etc.

Intercambio directo de información

El intercambio más importante de información no incluido en el código se lleva acabo por medio de listas de correo electrónico, canales IRC y documentación. En el caso de las listas de correo electrónico, los mensajes son almacenados en archivos que deben ser analizados. En lo que respecta a la documentación y al IRC todavía no está muy claro lo que se busca, y sobre todo, cómo se debe hacer de forma automática.

Listas de correo

Las listas de correo toman los archivos de texto que generan GNU Mailman, majordomo u otros gestores de listas de correo electrónico. Este tipo de archivos suelen encontrase disponibles por medio de HTTP. Se descarga el archivo con los mensajes durante un cierto espacio temporal, que en general es de un mes, de la lista y toma de las cabeceras de los mensajes tanto el autor como la fecha de envío. Los datos que se pueden extraer de las estadísticas de las listas de correo electrónico son: Nombre y dirección del autor; Fecha Nombre de la lista, de forma que se puedan adjudicar las estadísticas a un proyecto o metaproyecto. En un futuro se pretende añadir la capacidad de seguir el hilo de la discusión o incluso alguna forma de cuantificar la longitud del mensaje, aunque para ello se deberán encontrar métodos para eliminar las líneas que corresponden a un mensaje original al que se está respondiendo o a las firmas PGP.

IRC

No parece muy claro que otros parámetros interesantes se pueden extraer de las estadísticas del IRC, más allá del número de personas que se congregan en un canal. Sin embargo, también es verdad que la existencia de muchos bots que las generan de forma semiautomática hace que no haya que molestarse mucho en su implementación.

En el caso de las pruebas que se han realizado hasta ahora, la toma de datos no ha sido posible. Muchos canales no quieren tener un bot que les "espíe", por lo que hay que ir con sumo cuidado y pedir permiso en primer lugar.

Herramientas de desarrollo distribuidos

En gran parte, el desarrollo de software libre se basa en unas herramientas que permiten sincronizar el trabajo de los diferentes desarrolladores del proyecto, de forma tal que la distribución geográfica no suponga un problema. Los sistemas de control de versiones y los gestores de erratas, también utilizados ocasionalmente para llevar a cabo tareas de planificación, se han convertido en herramientas imprescindibles para proyectos de software libre grandes, y no tan grandes. Estos sistemas suelen registrar las interacciones con los desarrolladores y, por tanto, una vez que se consiguen estos registros puede dársele seguimiento de manera bastante sencilla a todo el proceso de desarrollo.

Sistema de control de versiones

El uso de un sistema de control de versiones sirve para organizar el desarrollo distribuido, y a veces simultáneo, en proyectos de software libre. El más utilizado en la actualidad por los proyectos de software libre es el CVS. Un análisis de los cambios que se van realizando al repositorio que estos sistemas mantienen, ofrecerá mucha información con respecto a la participación de los desarrolladores, además de la posibilidad de ver si existen ciclos de desarrollos. El estudio de los resultados obtenidos por esta vía se puede extender de manera notable si los datos obtenidos se pueden correlacionar con las inspecciones de código y la actividad en las listas de correo, así como con los datos socio-laborales de los desarrolladores.

El Cvstat2 es una extensión del cvstat de J.Mallet el cual se concibió para poder funcionar de forma distribuida. El objetivo es que junto con la aplicación de extracción de datos, se distribuya un interfaz Web simple e intuitiva por medio de la cual se pueda ver la evolución del proyecto en el CVS. De esta forma, cualquier equipo de desarrollo podrá descargar, instalar y configurar el software al mismo tiempo de medir sus interacciones con el repositorio CVS. Además, estos datos serán exportados, de manera que se descarga el procesamiento de un repositorio central de datos en formato intermedio.

El objetivo de la distribución hace que el software que generará sea lo más fácil de conseguir e instalar. En un principio, la idea es que una vez instalado mediante procesos automáticos, sea la propia aplicación la que se encargue de actualizar sus datos y exportarlos, de forma tal que la manipulación humana sólo se tenga que darse sólo durante los pasos de instalación y configuración.

Por otro lado, la distribución de esta herramienta puede ser una buena forma de promocionar la investigación que se va a llevar a cabo, ya que todo el mundo puede contar con cvstat2 para su propio proyecto y, si exporta sus datos, se sentirá parte de una gran comunidad que aporta para la investigación del fenómeno del software libre.

Los datos que se pueden obtener por medio del cvstat2 son: la fecha del commit que es la acción por la cual un desarrollador sincroniza su versión local con la existente en el repositorio; un archivo modificado, desarrollador, número de versión (CVS), número de líneas añadidas, y número de líneas borradas.

Sistema de seguimiento de errores

En la mayoría de los proyectos grandes de software libre, la existencia de errores críticos hace que la publicación de una versión estable se retrase. Debian y GNOME son dos ejemplos de esto, aunque seguro que hay muchos más. La incidencia de errores críticos es muy importante a la hora de realizar la publicación definitiva en grandes proyectos de software libre. Un ejemplo de muy actual es la segunda versión de la plataforma GNOME. Su publicación definitiva se ha retrasado varias semanas, porque tenía varios errores críticos que no se habían podido corregir.

Los datos que se pueden extraer son: la fecha de apertura de una errata, la catalogación de una errata, el número de las interacciones, la fecha de las interacciones, el autor de las interacciones, y la fecha de cierre de una errata.

Hoy en día, no existe ninguna herramienta que extraiga los datos que se acaban de mencionar. El sistema de control de erratas, BugZilla, cuenta por ahora con la funcionalidad para extraer estadísticas del número de erratas abiertas, cerradas y existentes, pero esos datos son insuficientes para los propósitos deseados en este taller. De cualquier forma, como BugZilla utiliza una base de datos para almacenar los datos, no se puede descartar la idea de pedir una copia, o un acceso directo, para que pudiera ser analizada completamente.

En el caso extremo, se podría crear un parser que tomara de los datos estadísticos de forma automática de las páginas Web, junto con los informes de errata, aunque esto plantea siempre el problema de que un cambio en el HTML de Bugzilla signifique que se debe adaptar el programa que parsea esos datos.

Otras Herramientas

Aparte de las herramientas ya presentadas, existen otro tipo de herramientas que no entran en la clasificación que se ha seguido hasta el momento y que, por tanto, serán presentadas a continuación.

Parser y otras herramientas

El SFparser recorre las páginas Web de todos los proyectos que se hospedan en SourceForge y obtiene información sobre los desarrolladores que están dados de alta, las últimas publicaciones de todas las ramas del proyecto, así como los datos que lo describen y lo ordenan dentro de SourceForge (licencia, lenguaje de programación, destinatarios, estado...).

Este SFparser es un script en Perl muy sencillo que empieza por el proyecto con identificador número 1 y, en el caso de que exista dicho proyecto, se descargan las siguientes páginas HTML: la página principal, la de las listas de correo electrónico, la del CVS y la de la descarga de archivos.

A continuación se presenta la lista de parámetros que devuelve Sfparser:

Nombre del proyecto; Página Web; Descripción; Categorización; Tipo de proyecto; Estado del desarrollo Lenguaje(s) de programación; Lenguaje natural; Licencia(s); Audiencia; Número de desarrolladores; Nombres de usuario; "Cargos" dentro del proyecto; Existencia de CVS; Existencia de listas de correo; Nombre de las listas; Paquetes, incluso de diferentes ramas; Versiones; Fecha de publicación; Tamaño; Descargas; Información sobre la existencia o no de CVS; y la listas de correo puede ser utilizada por cvstat2 y MailListStats para descargarse el repositorio en un caso y los archivos en el otro y proceder

a su análisis como se ha descrito con antes.

Además, al utilizar le mismo software y algunas ligeras modificaciones, se podrían extraer los datos de sitios que utilizan SourceForge con ligeras modificaciones como son BerliOS y Savannah.

Empaquetadores

Codd-find-latest es una aplicación que toma una lista de paquetes, se queda con las últimas publicaciones y desecha las más antiguas. A continuación se presenta el algoritmo que sigue: si existen paquete-1.2.3.tar.gz y paquete-1.3.4.tar.bz2, supondrá que el segundo es más reciente. Es una herramienta muy práctica en el caso de que se estén estudiando los repositorios de código que sólo requieran las últimas versiones de los paquetes. Éste es, por ejemplo, el caso de CODD, ya que al comprobar el código compartido entre varias aplicaciones, se confundiría de manera notable si existieran dos versiones de la misma aplicación.

Otras fuentes de información

A diferencia de los proyectos de software "tradicionales", para el caso del software libre se desconocen los recursos humanos. La situación socio-laboral, económica, geográfica y cultural de los integrantes de los proyectos de software puede ser muy dispar y si ves esto tiene repercusión en la forma en la que un proyecto de software evoluciona.

Datos laborales (y personales) de los desarrolladores

Además, existe una serie de datos que no se podrán extraer de forma automática y que, sin embargo, son muy interesantes para conocer más a fondo la comunidad de software libre. Básicamente, este tipo de datos son datos personales de los desarrolladores.

Datos personales interesantes:
Nombre
Nacionalidad
Fecha de nacimiento
Sexo
Formación

Quizás en algún momento, si este proyecto cuenta con un buen apoyo y los desarrolladores se muestran partidarios sería interesante crear una especie de contador de desarrolladores al estilo LinuxCounter. Hoy en día, esto es una idea poco práctica que se puede descartar.

Por otro lado, los datos de carácter personal, y los datos laborales aportarían una visión muy rica al conjunto. El hecho de poder saber qué desarrolladores se dedican

profesionalmente al desarrollo de un proyecto de software libre, permitirá hacer correlaciones y sacar conclusiones no menos interesantes. Por eso, para entornos reducidos y conocidos, se debería poder contar con una base de datos con los nombres, nombres de usuario, número de horas a la semana y fechas en la que estuvo contratado.

Los resultados que se obtengan pueden ser utilizados para, en conjunto con otras fuentes, se puedan calcular los costes y los tiempos de desarrollo. Una de las primeras acciones que podrían llevarse a cabo es integrar estos resultados en la algoritmia de SLOCcount para ponderar convenientemente el modelo de COCOMO , o incluso realizar un nuevo modelo que se encuentre en mayor conformidad con los desarrollos de software libre.

Datos laborales interesantes:
Nombre
Fechas en las que ha estado/estuvo empleado
Tiempo parcial/completo
Proyecto para el que estuvo empleado

En GNOME, KDE, Apache o incluso Linux se habla de comunidades en las que este tipo de datos se podría conseguir con una fiabilidad bastante aceptable.

Encuestas

También se tiene la posibilidad de referirse a encuestas para los desarrolladores que se han hecho hasta la fecha. Por desgracia, las encuestas tienen la desventaja de que no pueden ser reproducidas. Aún así, los resultados de las encuestas pueden servir de punto de partida para analizar el software libre e interpretar los resultados que las diferentes fuentes y análisis pueden ofrecer.

Algunos aspectos hacen necesaria la realización de encuestas. Es difícil cuantificar la motivación, el estado de ánimo u opinión de los desarrolladores de software libre. También es siempre muy interesante observar que cierto tipo de desarrolladores tienden a agruparse en cierto tipo de proyectos, mientras que otros lo hacen en otros.

Formato de intercambio de información

Primero, se intentará que todas las herramientas utilizadas devuelvan los resultados en un formato intermedio e independiente que permita aglutinar los resultados de las diferentes herramientas de forma sencilla. El formato elegido debe ser muy flexible, ya que puede que en un futuro próximo se le añadan más. Hoy en día, lo mejor sería utilizar un formato XML, ya que cumple con todos los requisitos comentados con antes y además permite la compatibilidad hacia atrás. También se debe tener en cuenta que los conversores de XML a cualquier otro tipo de formato que se desee no serán muy difíciles de

implementar.

Herramientas de conversión a XML

Al partir de aplicaciones de extracción de datos ya existentes que utilizan sus propias formas de almacenamiento de datos, puede ser necesaria la creación de herramientas que conviertan los datos del formato original al formato intermedio e independiente en XML.

Un ejemplo de esto podría ser CODD que, como se ha podido ver con anterioridad, utiliza un formato propio de archivos. Para llevar a cabo la conversión hará falta una aplicación que bien podría llamarse codd2xml. En el caso de SLOCcount, también será necesario una especie de sloc2xml.

Herramientas de conversión a otra cosa

El uso del formato intermedio e independiente en XML plantea muchas ventajas, pero puede no ser el más idóneo para la realización de ciertas tareas. En el caso de que se desee generar un interfaz Web para acceder a los datos, lo más sencillo, y eficiente, es utilizar una base de datos relacional. Por lo tanto, será necesario contar con alguna forma para convertir XML a SQL. De esta forma, habrá que estudiar cómo realizar las optimizaciones, como por ejemplo normalizar las tablas SQL generadas.

De igual forma, muchas de las herramientas de estadística utilizan un formato de entrada muy simple denominado SPSS. Estas herramientas permiten crear correlaciones y diagramas con bastante rapidez, ya que han sido diseñadas para ser utilizadas por sociólogos y psicólogos para el análisis de encuestas. La conversión a SPSS tiene dos vertientes interesantes, se tiene la posibilidad de utilizar este tipo de herramientas y, sobre todo, puede facilitar sobremanera el que científicos de otras ramas puedan obtener los datos en un formato con el que estén familiarizados para proceder a su análisis. Todo lo que sea despertar interés hacia la ingeniería del software libre en otras ciencias es de gran importancia.