Titulo

Identificación de “hotspots” usando métodos de Optimización y/o Machine Learning

Responsables

- Abel García Nájera
- Humberto Cervantes Maceda

Perfil deseable

- Conocimiento básico de inteligencia computacional
- Haber llevado el curso de “Administración de proyectos”
- Saber programar

Presentación de contexto e identificación de problemática

Una de las herramientas indispensables a los equipos de desarrollo son los sistemas de control de versiones, pues permiten apoyar al equipo en centralizar y versionar la base de código y evitar conflictos al realizar modificaciones al mismo. Herramientas como Git [1] almacenan información muy valiosa al momento en que los desarrolladores realizan “commits” de los cambios. Un ejemplo de esto se puede apreciar en la entrada siguiente:

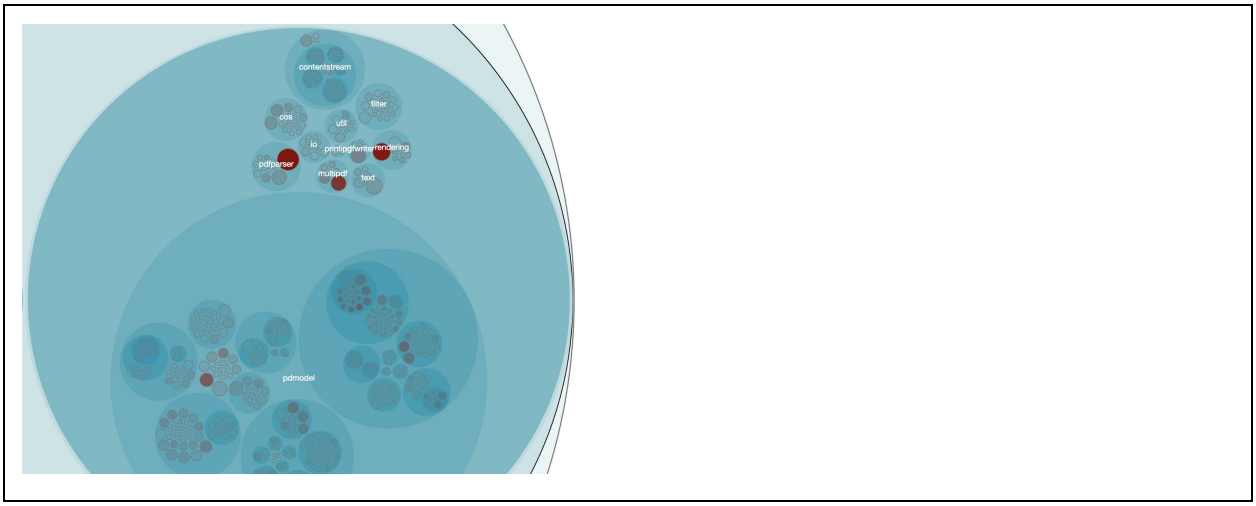
```
commit db1aece429e0fb46054c732546362a83930ac52c
Author: Tilman Hausherr <tilman@apache.org>
Date:   2019-11-17 14:25:16 +0000

    PDFBOX-4071: SonarQube fix: remove exception that won't be thrown

    git-svn-id: https://svn.apache.org/repos/asf/pdfbox/trunk@1869946
13f79535-47bb-0310-9956-ffa450edef68

1       2
examples/src/main/java/org/apache/pdfbox/examples/signature/CreateEmbeddedTimeStamp.java
1       1
examples/src/main/java/org/apache/pdfbox/examples/signature/CreateSignedTimeStamp.java
1       2       examples/src/main/java/org/apache/pdfbox/examples/signature/SigUtils.java
1       1
```

Como se puede observar, git almacena para un commit dado la fecha, un identificador, el autor, un mensaje de resumen, así como la lista de archivos que fueron cambiados y la cantidad de líneas de código que fueron cambiadas en estos archivos. Estas bitácoras pueden ser analizadas y se pueden identificar, por ejemplo, los archivos que cambian más frecuentemente en un proyecto en un periodo dado. Por ejemplo, la visualización siguiente muestra los archivos que han cambiado más frecuentemente en el proyecto [2].



Un problema importante en el desarrollo de software es la acumulación de “deuda técnica”. Esta deuda ocurre por malas decisiones de diseño o erosión del diseño por errores en la implementación. Su existencia resulta en reducción de la productividad del equipo, que debe dedicar más tiempo de lo ideal para hacer cambios [3].

En este proyecto estamos buscando aplicar métodos heurísticos de optimización o de aprendizaje automático (machine learning) para identificar puntos donde se puedan realizar intervenciones para reducir la deuda técnica de los proyectos (“hotspots”). Estos hotspots frecuentemente están relacionados con los archivos que cambian más frecuentemente y que pueden ser identificados mediante el análisis de bitácoras del sistema de control de cambios.

Objetivos generales y específicos

General

Aplicar métodos heurísticos de optimización o de machine learning para identificar puntos de intervención para reducir la deuda técnica.

Especifícos

- **Obj. 1.** Identificar propuestas en la literatura enfocadas en reconocer puntos de reducción de deuda técnica en el código.
- **Obj. 2.** Proponer un método heurístico de optimización o de machine learning que facilite la identificación de hotspots.
- **Obj. 3.** Desarrollar un prototipo de herramienta que implemente dicha técnica.
- **Obj. 4.** Evaluar la efectividad de la herramienta.

Metodología

- **Act. 1.** Realizar un estudio de mapeo sistematizado de la literatura.

Act. 2. Desarrollar y evaluar el método heurístico de optimización o de machine learning de forma iterativa con el fin de refinarlo. Para su evaluación se usarán proyectos de fuente abierta.
Act. 3. Implementar un prototipo de herramienta con el método antes antes mencionado.
Act. 4. Comunicar los resultados.

Calendario

La siguiente tabla presenta un calendario tentativo para el proyecto. Nota: Los meses son meses “efectivos” de trabajo.

	01	02	03	04	05	06	07	08	09	10	11	12
Act1												
Act2												
Act3												
Act4												

Resultados esperados

Se espera que el alumno cumpla con los hitos establecidos para el programa:

- Capítulo 2 de la tesis (Estado del arte) completado al final del 1er trimestre.
- Capítulo 3 de la tesis (Propuesta teórica) completado al final del segundo trimestre.
- Capítulos 4 (Desarrollo de la herramienta) y 5 (Evaluación) completados al final del tercer trimestre.

Referencias

1. Chacon, S., Straub, B., “Pro Git, 2d Edition”, Apress, 2014.
2. Tornhill, A. “Software Design X-Rays - Fix Technical Debt with Behavioral Code Analysis”, The Pragmatic Bookshelf, 2018.
3. Kruchten, P., Nord, R., Ozkaya, I., “Managing Technical Debt, Reducing Friction in Software Development”, Pearson Education 2019.

Infraestructura necesaria y disponible

El participante requiere de una estación de trabajo con acceso a internet, así como el acceso a un servidor que albergue un repositorio de código y de información. A nivel de software, todas las aplicaciones que se utilizarán son gratuitas. Toda esta infraestructura está disponible actualmente.

Lugar de realización

El estudiante dispondrá de máquinas recientemente adquiridas y que están ubicadas actualmente en el laboratorio de sistemas distribuidos.

Información adicional

- El alumno deberá tener juntas semanales de una hora con sus asesores.
- Toda la documentación del proyecto se llevará en Google Docs.