6. Mining Software Repositories
(Last Update: 2024-03-06)

Many companies, as well as open-source projects, use Version Control Systems (VCS) like CVS, SubVersion, ClearCase and Git to record the history of source-code files. These VCS record the whole development and the corresponding change activities applied on the source code. Furthermore, these versioning systems are often associated with change management systems which allow the users to report bugs and request new features. Popular change management systems include repositories like Bugzilla and JIRA. These systems record the entire evolution of a software system.

The highly active Mining Software Repositories (see MSR conference) community shows that the evolutionary information implicitly stored in these systems may provide valuable insight into how the software is being developed. The evolution of a source-code file over time may explain better why previous design choices have been made in the file. The first step to analyse the evolutionary information of a software system consists of extracting the necessary information from the version control system or change management system.

Some interesting studies include:

- Who Should Fix This Bug?
- Don't Program on Fridays!
- Linking E-Mails and Source Code Artifacts
- Predicting Faults from Cached History
- and many more...

This lab session serves as a first contact with this thriving domain. As such you will learn about the kind of knowledge that can be extracted from a software repository.

Sample Project
pandas-dev/pandas

Materials & Tools

1. Git
2. PyCharm IDE
3. CodeScene
4. Gource.

Task 1: Query the Versioning System

First, we will use simple queries applied on the version control system to identify interesting knowledge about the software system at hand. Clone the repository and answer the questions using the Git log command.

1. Can you recognize any reference to the project pandas?
2. What valuable information can you extract from this Git log file about the evolution of the software system?
3. How can you use the information implicitly stored in the Git log to tackle following problems:
    - Read all the code in one hour.
    - Chat with the maintainers.
    - Study the exceptional entities.
4. Combine the Git log and grep commands to extract the following information:
    - Number of commits of **Matthew Roeschke**
    - Number of times that the file **series.py** has been modified.

Task 2: First Contact Visualizations

The previous analysis highlights that the repository may provide relevant information related to the evolution of the system. However, writing command line queries to understand a software system might be quite daunting, especially during your first contact with a system. Therefore, in this section we shall use PyCharm's Git plugin to visualize the system.

Open pandas within PyCharm, locate the Git plugin window and use it to answer the following questions.

1. Which files were recently changed by **Patrick Hoefler**?
2. Which developers can you contact for more information about the **TestTypes** class?
3. Given the last version of file **array.py**, discover:
    - Which files have been modified along with it
    - Who are the authors of the lines **181**to **200**
    - When was the last time these lines were modified, what were the changes, and who made them?
4. Can you identify the source code files which are unstable, i.e.: files which have been subject to many changes?
5. Can you identify which files are modified to fix bug **#52058**?

Task 3: Another brand of Visualizations

The ability to perform this analysis within your IDE is a plus. However, CodeScene also offers a similar visualisation with its Team Dynamics visualisations that are tailored to a different theme.

1. Under Individuals, can you identify:
    - The main author of **pandas**.
    - The main author of **test_frame_apply.py** and which file it often changes with.
2. Can you identify the largest knowledge risk.
3. Can you answer all the questions in Task 2 using **only** CodeScene?

Task 4: Fancy Visualizations

To complete this section, install Gource.

1. Try to answer the questions put forth in the previous tasks using Gource. How many can you answer?
2. In which use case would you employ Gource for refactoring?
3. Which Refactoring Principles does it help you implement?

Conclusion

These are only a few of the tools able to present a quick overview of the information reported in a software repository. There are many others - often developed by the MSR community. Of course, it is possible to write your own to suit your needs.