Software Component Time-Domain Visualization

David Kavaler, Sugeerth Murugesan, Vishak Muthukumar

Project Description

It is well-established in the literature that a number of code complexity, process, and social metrics have high correlation with bugs in software. Though this is known and accepted, these metrics can sometimes be difficult to conceptualize for those who are not experts in the field. In addition, predictive models are built on average case behavior, while the ecosystem of a particular project may have fine-grained details that stray from the norm. Also, when considering a system over time, complex dynamics between changing parts makes understanding much more difficult. This project will provide a tool that aids in visualizing metrics and system evolution at varying granularities so e.g., a security analyst may be able to spot potential problems in a system. In addition, we aim to provide further evidence of useful metrics in predicting bugs - specifically security vulnerabilities. To show this, we will use data gathered from the Apache Software Foundation projects and use an established strategy of linking bugs to bug-introducing and bug-fixing commits. We will then overlay mined CVE data with these bugs to examine our precision and recall. All of the information gained from this process will be fed back into the visualization tool, providing a useful look into the projects under study. Our goal is to have this tool operate on any software system that uses Git for version control. We have not changed our project description from our initial proposal.

Work Accomplished

We wrote a proposal and have kept updated PowerPoint slides on weekly progress. In addition, we performed a literature review to search for possible explanatory metrics and existing visualization methods. This literature review was very useful in providing ideas for our approach.

We mined version control data (Git) for the Apache Software Foundation projects and calculated around 40 candidate metrics. We decided to select a handful of these metrics to build a proof-of-concept tool based on prior work. These metrics are measures of churn (number of lines added and deleted), code size (count of lines of code, count of executable lines of code, count of comment lines), object-oriented feature use (total number of functions, total number of local public methods, total number of local private methods), and complexity (ratio of comment lines to code lines, sum of cyclomatic complexity of all nested functions or methods), and are calculated per file and per commit. These metrics provide a view into the evolution of the software system and have been shown to be indicators of potential bugs.

In addition to gathering metrics based on mined version control data, we collected data on bug-introducing and bug-fixing commits based on JIRA issues. The Apache Software Foundation projects use JIRA to track issues within each system. They have a strict policy that requires each JIRA issue to be addressed by a single commit. This allows us to run git blame on the fixing commit to see which lines were changed, allowing us to see which commit (or commits) introduced a particular issue. Using this data, we can track the introduction and fixing of bugs within the system and visually overlay this data on top of other metrics to see their relationship (if any). In addition to bugs mined in this way, we gathered data from the CVE system for Apache Software Foundation projects. Using CVE data as a source adds another dimension in which we can see which bugs mined through the JIRA issue system could have led to security vulnerabilities.

Fig 1. The visualization represents a general timeline of the files that are in the repository. The colors are represented by different developers and each line represents a file. The x-axis is the time line. We can visually infer from the dataset that the repository is most contributed by the author in “golden”.

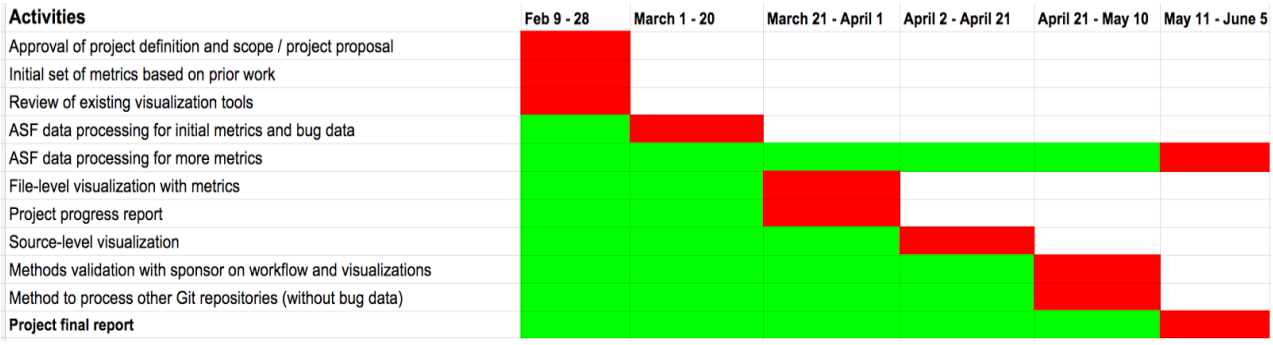
The visualization in Figure 1 is implemented using QPainter class of the Python-Qt framework. The image provides an overall view of the source-code repository with adjustable heatmaps. The colors represent different developers and each horizontal line represents a file. The x-axis is the time line. A few interesting inferences which can be obtained from this are -

1. The most highly represented author is the author in gold
2. Some files have periods of frequent contributions from multiple authors
3. This may be a good tool to compare whether multiple commits from same author or different author on the single file can be a cause of bugs

Work Remaining

While our prototype tool has been applied to a single Apache project, we still need to apply the same tool to other Apache projects. This should not be an issue as the data is formatted in the same way for all projects, but there could be unforeseen difficulties.
Also, as of now our tool displays colors based on author contributions. Our plan is to have colors represent the metric that the user may be interested in (e.g. churn, file size), and we will extend our tool to handle this accordingly. In addition, we will devise a method of overlaying mined bug data and CVE data on top of our visualization of software metrics to illuminate correlations or potential causal links.
We currently have a single view type implemented: files and their evolution over time. However, there are a myriad of other views that could be used to display our data. We plan on creating multiple methods to display our data, all of which may be useful to the analyst or general user of our tool.

Updated Timeline:



Updated Bibliography

New addition:

[1] Data Visualization: Principles and Practice, Second Edition, *Alexandru C. Telea*

Project Difficulties

1. CVE data processing - CVE data is available as raw strings, processing that data to overlay in the visualization is very hard. Mitigation - Still exploring.
2. Figuring out the most intuitive way to represent the transition of a software code in its lifetime. Mitigation - Literature review of existing similar visualizations
3. Making the visualization as general as possible to visualize the change in software engineering metrics for a software in course of its lifetime.