Final Project Proposal ECS 272

Suchita Mukherjee

24th February, 2020

1 Original Publication

The paper proposed for implementation is ”StarGate: A Unified, Interactive Visualization of Software Projects” by Michael Ogawa and Kwan-Liu Ma, as accessed from
https://ieeexplore.ieee.org/document/4475476

2 Summary

The paper by Ogawa and Ma has as it's central goal to propose a visualization technique to help users understand the complex interactions between developers and software repositories while also portraying the evolution of the software project over time. Since most software projects having multiple developers use version control systems to keep track of code and allow simultaneous editing, the data from these systems can provide important details and also pose a challenge for visualization. The authors present their system, StarGate which visualizes code the code repository and the social media for developers. According to the authors, this system is capable of benefiting different users in specific ways. It can help newcomers to a project understand which developers work on which aspects of the code and thus help them to seek help from relevant groups. Project managers can quickly gain an overview of the development process and the communications within the team while software engineering researchers may utilize this visualization to find the relationships between project organization and source code. StarGate also shifts the focus of software repository visualization from the typical file and source code centric view to an author centric view. The system is designed to work with the version controlled software repository and the project mailing list archive.

2.1 Visual Encoding Design

The base of the visualization is formed by the Gate, which is essentially the ring of directories contained in the repository. The innermost ring of the Gate represents the root of the repository and the levels of the hierarchy grow outward as concentric ring slices. The size of the slices depend on the number of files in the directory.

This visualization puts the developers at the center of attention, by representing them by Stars within the Gate. The size of each star is determined by the number of file modifications a developer has done. The positioning of the stars is done by the areas of the repository that the developer has worked the most on. The positioning allows the user to immediately recognize the developers concentrating on a specific area and the spatial grouping leads to natural clustering. The appropriate position of a star is calculated as the centroid of the surrounding modified files, with the "pull" of each file being weighted by the number of times each file was modified. By using the centroid as a positioning metric, the positions are ensured to always lie within the Gate. The links between the stars represent the communications between them thus showcasing the developer social network.

The outer ring of Stardust represents the files that the selected authors have modified. Each file's edit history is placed along a thin line extending from its place in the directory hierarchy. The line can be thought of as a timeline, with the older events closer to the center and more recent events being present outwards.The color of the dot is the same as the color of the star of the developer who modified it. The stardust helps the user understand the amount of files a developer has modified.

2.2 Interaction Design

To tackle the labelling issue of clustered stars, a hover effect is added to the stars. On hovering over a star, a popup appears with the name of the developer represented by the star. On right-clicking a star another popup provides an option to assign color to the star. Initially all stars are colored the same as the authors anticipate that the users can utilize color as an important analytic tool. Left-clicking on a star will select that particular star, while a group of stars may be selected by dragging a box around those stars (rubber banding). The selected stars are distinguishable by a visible halo, their developer names are made visible, Their connections are shown in the social network as constellations, their file history is shown as Stardust and during a time lapse animation they leave a visible trail. To aid finding a specific developer, a text-box is provided to search a developer by name while another text-box accepts filenames and highlights all developers who have modified that file. The social network connections between the stars can be viewed by adjusting a slider that controls the opacity of the edges. Initially, the opacity is set to zero unless the user selects any star or adjusts the slider to reveal the entire social network.

The user may click on one of the files in the stardust and it pops up a file details dialog window which contains a table showing file modification history. StarGate also provides Time Travel which allows users to explore the evolution of the repository. With the help of a time slider, the user can change the latest time and even choose to increment the time automatically as an abstraction. The tools also allows users to track movements of selected stars as star trails. The authors evaluated the visualization tool using datasets of Apache web server project and comparing it to PostgreSQL project. The visualization is able to provide the users satisfactory insights about each project and aids in comparing the software development practices of the two projects. The authors recognize that a visual paradigm like StarGate can be adapted to visualize network and hierarchial data of different sorts like collaborations between music artists.

3 Dataset

BugSwarm is the largest data set of its kind, having the information of thousands of real software bugs and their fixes with the ability to grow continuously. Diagnosis and repair of software bugs costs time and money, and has a great impact on the economy, safety and quality of life. Several software engineering subfields are dedicated to finding and repairing defects based on the knowledge of past defects, but these methods need to be trained on realistic up-to-date datasets of defects. Unlike any other curated dataset of software bugs, Bugswarm surpasses its predecessors in reproducibility, scale and realism. It currently has over 3,000 data points mined from 180+ open-source artifacts.
Each data point in BugSwarm is made of a fail-pass pair in the build process of a repository. The data set captures the repository name, the programming language of the project, the test framework used, the build system integrated, the operating system of the server in which the project is deployed, the exceptions responsible for the build issue, whether the build failed completely or passed while throwing an error, the reproducibility of this build-fail pair, the commits and pull requests in this repository, the location of the fix, details about the last reproduction attempt, changes, addition and deletions required for the fix and the time stamps for creation, updation and last reproduce attempt. For the purpose of this project, the dataset snapshot dated December 2019 is being used which contains 3140 datpoints from 186 repositories. Most of the fields are categorical, except for reproducibility which is measured as the ratio of number of successful reproduce attempts to the total number of reproduce attempts.

4 Implementation

This project will be aimed at visualizing the growth of the BugSwarm dataset using the visualization design of StarGate. While StarGate is originally designed keeping in mind the complexities of a single repository, the BugSwarm dataset encompasses a large number of repositories. To tackle the challenges of this adaptation, some tweaks need to be made to the design and some features may be excluded from the final implementation, as described below.

4.1 The Stars

While StarGate put the developers at the center of the visualization, the adapted version is going to show case all the repositories at its center. These open source repositories are truly the heroes of this data set and provide the lifeline of the BugSwarm project. The size of the stars will represent the size of each repository, which will be measured by the number of commits in that repository. While developers are naturally known to communicate and StarGate utilised email archive data in conjuction with the version control system data to form the social network between the stars, the repositories in the BugSwarm dataset do not have any specific networking within themselves. Hence, the linking of the stars is one feature that appears to be unsuited for this dataset. Currently at this stage a netwoking between the stars does not seem apparent, but in due course of the project if such a connection imerges then this feature will be implemented, or justification will be provided for exclusion.

4.2 The Gate

The Gate, which is the base for this visualization, will be formed of concentric layers with each layer representing a feature of the repository. For example, the first layer will be the language of the repositories which is one of the most important aspect of each repository. The next advanced layer can represent test frameworks and build systems which are dependent on the programming languages used. In this manner, exceptions, reproducibility, operating systems can become subsequent layers.

4.3 The StarDust

The star dust section of the visualization will be used to represent the changes, additions and deletions created for the fixes over time. Since there is a time stamp for creation and updation in the repository, the changes to the repository can be placed along a timeline as done in StarGate.

4.4 Interactions

Most of StarGate's interactions will be well-suited and useful for exploring the BugSwarm dataset. The selection of individual star, rubber banding a group of stars, assigning colors to the selection of stars will be implemented. The name of the repository will be shown on hover and the name label can act as a hyperlink to the actual GitHub repository. A text box will be provided to search for a repository, language, exception or patch location. These are usually important features that users may want to select the associated repositories for. The search will help them highlight all concerned repositories together.

The user will be able to assign different colors to the concentric layers of the Gate. Thus, even in this adaptation, color can play an important tool to aid users' analysis and exploration process.On clicking on any point in the star dust, it can open a pop-up window to show the commit history of the repository and provide a hyperlink to that specific pull request on GitHub. With the help of a time-slider, the user will be able to see the evolution of the data set and also view a time-lapse video as more and more repositories are added to the project.

5 Timeline

The initial plan will be to finish visualizing the stars and the gate by the time of the Milestone Report due date. These are the central components of the visualization and will require some critical thinking to explore possible features that can be added to the existing features of StarGate. After the Milestone project for the following weeks, I will be implementing the star dust and time-lapse animation. The Final Project Report will continue to be updated throughout the process as and when I progress so that I can document the process well and provide detailed explanation of the choices I make along the way.

References

[1] M. Ogawa and K. Ma, "StarGate: A Unified, Interactive Visualization of Software Projects," 2008 IEEE Pacific Visualization Symposium, Kyoto, 2008, pp. 191-198.

$https : //ieeexplore.ieee.org/document/4475476$

[2] D. A. Tomassi et al., "BugSwarm: Mining and Continuously Growing a Dataset of Reproducible Failures and Fixes," 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, QC, Canada, 2019, pp. 339-349. doi: 10.1109/ICSE.2019.00048

$https : //ieeexplore.ieee.org/abstract/document/8812141$