CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: Newcomer-meter for characterizing open-source project dynamics	
Sponsor Information: NORTHERN ARIZONA UNIVERSITY School of Informatics, Computing, and Cyber Systems	Igor Steinmacher, Assistant Professor, SICCS Northern Arizona University igor.steinmacher@nau.edu 928-380-0301

Project Overview:

Open source software (OSS) has become an important driving force in today's software development industry, resulting in many prominent projects that are used extensively through the entire development stack, from kernels to sophisticated end-user applications. Open source software runs on your phone, on your laptop, on your household appliances and cars, and are especially widely used on the servers that provide services over the Internet. The 10th Annual Future of Open Source Survey showed that 65% of the surveyed companies leverage OSS to speed up application development, and 55% leverage OSS for production infrastructure. To imagine the size of the movement, GitHub was hosting more than 67 million repositories, from 1.5 million organizations by the end of 2017. Some notable examples of successful projects include Tensorflow, Angular, Node.js, Kubernetes, Unix, React, Swift, Docker, Linux, Open Office, and Mozilla Firefox. Given this context, it is no surprise that the OSS movement attracts a large, globally distributed community of volunteers. They wish to participate in OSS as their contributions help them gain visibility, benefit society, and even get jobs.

OSS projects usually have a repository openly available in platforms like SourceForge, GitHub, and Bitbucket. The project repositories made available under these platforms provide not only their source code under an open license but also mechanisms to report bugs, request new features, communication means, and tools to better manage the development process. When the developers want to engage with some project, they usually go to the project repository, look for an open task (bug report or feature request), and start interacting with the code to complete the task, and the community to discuss the implementation, get feedback, and get their code reviewed. Newcomers to the project often rely on information available in the project repositories and on the answers from experienced developers to successfully contribute to the project.

Although OSS project development has been extremely successful, there remain many aspects of OSS development that are inefficient or prone to breakdown. In particular, motivating newcomers to join and retaining them in the project over time stubbornly remains a perennial weakness of OSS development. While joining OSS projects, newcomers face many difficulties entering OSS, and are left to learn on their own. They are akin to explorers who must orient themselves in a hostile landscape. In previous research, we have learned that newcomers face different kinds of barriers, including technical hurdles, interpersonal issues (including those related to community reception), and documentation problems. The lack of clear information or indications regarding these issues makes it difficult for newcomers to recognize how welcoming a particular project might be to newcomers...which can have a discouraging effect.

In order to develop tools to improve the experience of new developers on OSS development platforms, we need to better understand the internal social and communicative dynamics of the OSS teams. In our research group, we study developer behavior within open-source projects, analyzing how developers interact and contribute to these projects. Our main research line focuses on exploring ways to support newcomers when they are joining a new project. Our research methodology is generally driven by big datasets available from code repositories that allow us to analyze previous developers' interactions with the platforms including the code and documents they contribute, the messages exchanged, and traces of the process that they follow. By doing that, it is possible, for example, to verify how new developers are answered, which kind of contributions are usually submitted by developers, and temporally analyze the rate of newcomers joining a given project.

By analyzing the available data, we have already learned that newcomers need to be motivated to contribute to a project, and that the onboarding process is full of barriers that can demotivate even experienced developers. To support the newcomers during their first steps, we built FLOSScoach (www.flosscoach.com), a portal that proposes a way to better organize the documentation to better orient the new members. In the portal, newcomers find information on the skills needed to contribute to a project, a step-by-step contribution flow, the location of features (such as source code repository, issue tracker, and mailing list), a list of newcomer-friendly tasks (when provided by the project), and tips on how to interact with the community. This portal main contribution is related to knowledge management, since it presents a way to organize the information. FLOSScoach was experimented and proved to better orient and provide the right information for newcomers during their first steps.

A major current limitation, however, is that the existing portal is manually fed, and contains only already-existing information provided by project members. Moreover, the current version of FLOSScoach does not show current, up-to-date information about how welcoming a project is, or generate knowledge from the data available at the project repository. It would be interesting to have a toolset including automatically generated and updated information about how newcomer-friendly a project is, relying on data collected from repositories, which would enable newcomers to compare the projects when deciding about contributing.

To offer more up-to-date awareness support for OSS newcomers, what is needed is a novel software tool that will take as input data collected from GitHub repositories, including temporal graphs of newcomers' onboarding rate, update rate of documentation, and implementation of recommended community standards as per the opensource guide (http://opensource.guide). This data will be used to generate a set of visual indicators that shows how the project adhere to community welcoming standards, and interactive graphs enabling the analysis of the newcomers' influx and retention for a given project. In particular, we envision a web application, running on a Linux-based server that includes the following capabilities:

- given a GitHub project URL, collect all the initial set of data for a given project and create the indicators, including (for example):
    - percentage of accepted newcomers' pull requests (acceptance rate)
    - estimated processing time of newcomers' pull requests (mean time from opening to closing pull requests)
    - monthly influx rate of newcomers (newcomers/month)
    - retention rate (percentage of newcomers who stay for longer than a month)
    - availability and update rate of guidelines, code of conduct, and documentation files
- compare projects according to the information generated;
- generate interactive graphics based on historical data, enabling:
    - visualizing the history of newcomers/month

    - visualizing the history of newcomers' acceptance rate/month (accepted pull requests/all newcomers' pull requests)
- scheduling incremental collections (configurable) to update the information above;
- providing graphics and visual summaries via API, so projects and other tools can embed it.

This tool can be useful for those who care about new developers' onboarding, and to the open source ecosystem, since community members can use this as a starting point to make their projects more welcoming, or even by using the indicators provided as "badges" for the projects. I would also highlight the importance of these indicators for research, helping to further understand the behavior in open source projects.

Knowledge, skills, and expertise required for this project:

The team will need to know or learn how to:

- collect and deal with json data;
- program using python is desirable;
- setup the project in a Linux-based web server;
- use document databased (e.g. MongoDB). This would facilitate dealing with the data collected, although relational database would be OK;
- choose and use web front-end frameworks to create interactive graphics.

Equipment Requirements:

- There should be no equipment or software required other than a development platform and software/tools freely available online.

Software and other Deliverables:

- A fully-functioning web application, as outlined above, installed and tested on a platform of the client's choice.
- A "system administrators" manual that details step-by-step how the system can be installed on a platform of the client's choice, as well as how to perform basic configuration and maintenance.
- As-built report, that carefully documents requirements, design decisions, and implementation details. Should allow future team to easily pick up where left off.
- Complete professionally-documented codebase, delivered both as a repository in GitHub. There should be special attention to the API usage documentation, which will be used by third-party.