CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: A skill-matching approach to facilitate developers to choose their next Open Source project	
Sponsor Information: NORTHERN ARIZONA UNIVERSITY School of Informatics, Computing, and Cyber Systems	**Igor Steinmacher**, Assistant Professor, SICCS Northern Arizona University igor.steinmacher@nau.edu 928-523-2775

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 throughout 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, support for communication among developers, 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...but also with 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.

The Problem

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. When first exploring and considering joining OSS projects, newcomers face many difficulties and information gaps that they are largely left on their own to figure out. They are akin to explorers who must orient themselves in a new and unknown space that often has a well-established (and yet invisible) structure, culture and ways of working. 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.

Our recent work on the barriers that newcomers face when onboarding reveals that *finding an appropriate first "icebreaker" task to work on* is one of the most recurrent barriers that eventually prompt newcomers to dropout. This means that the first step towards making a contribution – just finding something good to work on – often poses an insurmountable challenging that eventually discourages newcomers from getting engaged. In particular, the challenge is that newcomers are expected to find a task that they can complete from a plethora of open issues that require different skills and are of varying complexity. A fundamental problem is that the skills that are required for a task are difficult to infer solely from the task description. This may be because few projects annotate tasks in any way to signal those that are appropriate for newcomers (partly because it's unclear how one would even effectively annotate/communicate such information), or because the amount of projects is gigantic, making it impossible to lurk around and find a subset of projects. What is needed is a robust mechanism for clearly but compactly assessing and revealing the gap between a particular newcomer's skills and the skills required for a particular project or task on its work list.

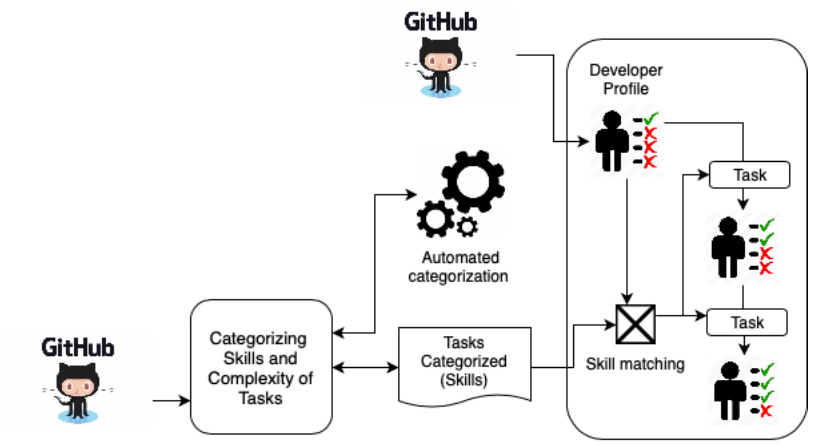
Envisioned Solution

The aim of this proposal is developing a tool that can automatically match the skills extracted from each GitHub user profile with skills that are determined to be required to work on various OSS projects, focusing on open issues identified in each project. The classifications will be made based on existing taxonomies, mining textual data (from code), analyzing previous contributions from the user. For our purposes, we'll define **skills** as the capability of performing an activity; skills can refer to knowledge about the contribution process, programming concepts (e.g., event handling), or about a class, an API, a library, or the domain.

In particular, we envision a web application, running on a Linux-based server that includes the following capabilities:

- given a GitHub project, collect all the initial set of data, including:
  - API's used by a given project (e.g. test API, third part API, etc.);
  - required programming language;
  - labels and tags used (to find easy for newcomers).
- given a GitHub profile, collect:
  - previous contributions (to assess existing skills)
  - projects that are followed or of interested (mining the stars and watches from GitHub)
- provide ways so developers can say what they envision to learn or tackle in future contributions
- create a resumé including developers' previous activities on GitHub
- create the skill matching tool, based on the information collected from the projects and users
- scheduling incremental collections (configurable) to update the information above
- providing graphics and visual summaries via API, so projects and other tools can embed it.

The expected workflow is presented in the figure below.



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 developers that want to find their next task more easily.

This proposal will contribute providing an infrastructure that scaffolds the skill acquisition of newcomers by identifying and recommending open issues such that, these tasks match newcomers' skills and goals. This tool may empower newcomers to become successful contributors and yield tangible benefits, since participation in OSS is now also used for recruitment. The proposed tool can also provide infrastructure for research and education, producing an integrated, open-sourced onboarding environment for use by researchers and other OSS communities.

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.