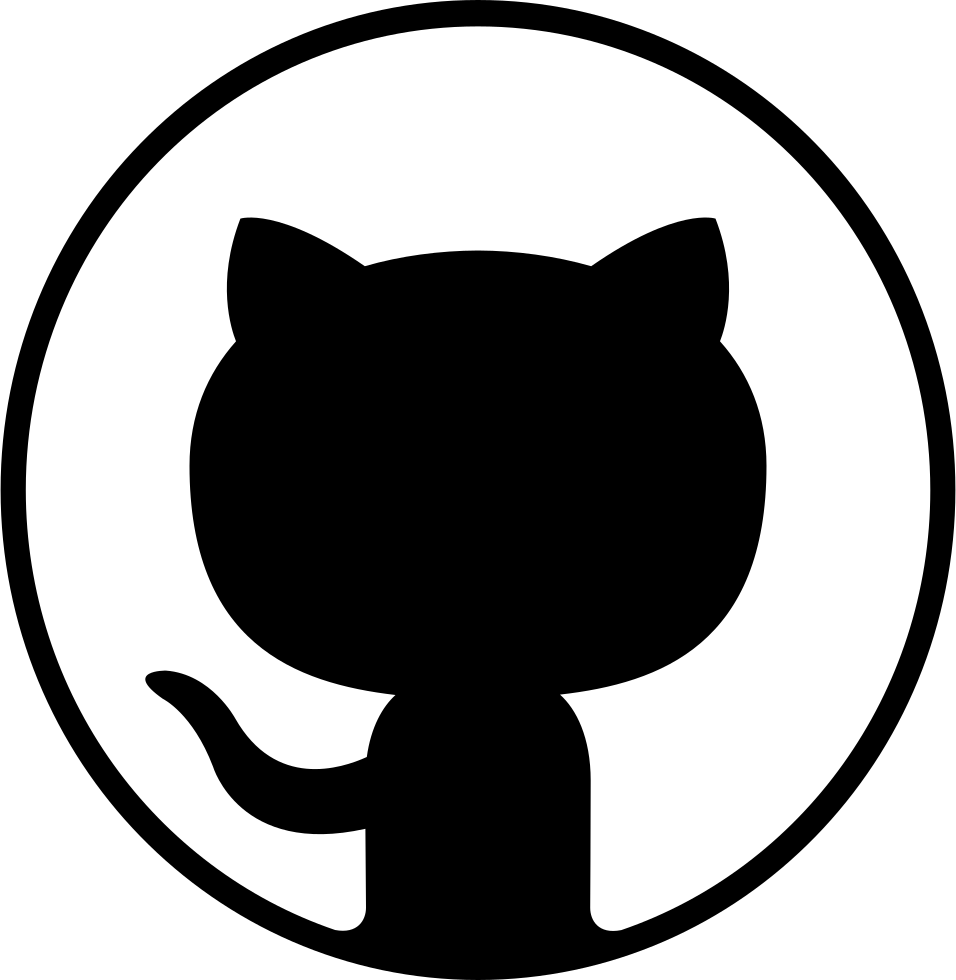
**Sessional :2**

**Git-Hub**

* **Logo:**

****

**Features:**

* https://lh3.googleusercontent.com/l7A22iQfAWoDWP1Jyewd7HiiQyfAkVSIbbzJqHOx8hVl7s8TzrUPlqlxy1iM8BmbQjj0eQ=s50**Collaborative Coding**
* Contribute to projects quickly with automatic environment setup.
* Make sure you see the changes you care about.
* Build community around your code.
* https://lh3.googleusercontent.com/5iKcvbomiyC6UgavZ4dtWJiBljmhjhgUXmO1u-7D61rc-946plVH7H50StRvKGDU-2rtiQ=s50**Automation and CI/CD**
* Automate everything: CI/CD, testing, planning, project management, issue labeling, approvals, on boarding, and more.
* Standardize and scale  
  best practices, security, and compliance across your organization.
* Get started quickly with thousands of actions from partners and the community.
* https://lh3.googleusercontent.com/kNuqj-pf2Mb16MGKx9wRi_o1iGPexDKK7A9Br3z-IKILl8HpMP9VM6oFoJB4YOzB9QKSiQ=s50**Security**
* Secure code as you write it. Automatically review every change to your codebase and identify vulnerabilities before they reach production.
* Understand and address any vulnerabilities in your open source dependencies.
* Automatically detect and deactivate secrets committed to your repos.
* https://lh3.googleusercontent.com/PIJBVwQnW9krq_9LsVzpHodPkLiRgfc738HvdAHPTTd9tL8c_rP6vGOQ-XxSWfjyNPrDvg=s50**Client Apps**
* Take GitHub with you on any connected device.
* Skip the visual UI with a quick, powerful text interface.
* Access GitHub from your OS X or Windows desktop.

https://lh3.googleusercontent.com/qJOYsbgSmtKEpydiKPQWmknNEml0nJaDmzcznLmnl4xLz7Q40-TRA6PcZKSox7Zd5PrBmQ=s51**Project Management**

Keep your team in step and your code at the center of your projects.

Give your team continuously updated information on your progress, priorities and roadmap.

Track what you deliver down to the commit.

**Team Administration**

Simplify access and permissions management across your projects and teams.

Update permissions, add new users as you grow, and give everyone the exact permissions they need.

Sync with Okta and Azure Active Directory.

Title:Analysis of Software Developer Activity on

a Distributed Version Control System

Author: Shu Li\*, Hayato Tsukiji\*, and Kosuke Takano\*\*

Year:2016

* In this paper, for the analysis of various characteristics of software developers, such as their skills and project roles, a method of feature extraction based on those developers’ history of collaborative

development using a distributed version control system has been presented.

* Real Git projects on GitHub were experimentally analyzed, and the

experimental results demonstrate the feasibility of our proposed method**.**

* **Figure :Data available using Git-Hub API**

**Project Name**

**Overview**

**Name of Developer**

**Commit Message**

**URL**

**Name of project Owner**

**HTML 5**

**PHP**

**Programming language and amount in which they are used**

**JS**

**Node.js**

**C**

* Our proposed method allows us to analyze each developer’s characteristics in terms of “strong development areas (M/V/C)”, “contribution”, “initiative”, “support”, and “leadership” in progressing software development on a Git project.

* **Figure: Items that can be analyzed using proposed model**

**(E)Leadership**

**(D)Support**

**(C)Initiative**

**(B)Contribution**

**(A)String development area(M/V/C)**

**Analysis of Developer**

**Analysis of MVC Framework**

**Controller**

**View**

**Model**

* **Example of Analysis result for a Software Development project using Laravel**
* Based on these data, our method extracts developer characteristics with regard to

(A) development areas

(B)contribution

(C) initiative

(D) support

(E) leadership.

Model:

**Analyzer**

**Git Projects on Git-Hub**

Laravel provider

(1)Analysis of MVC Architecture

Ruby On Rail provider

Java Spring 4 provider

Xxx Provider

**Providers for MVC Framework analysis**

(4)Count of number of function calls in each MVC layer

(5)Count of number of configuration files/tools committed

**Functions of analysis of development status in MVC framework**

(3)Count of no. of source codes modified by other developers in each MVC layer

(2)Count of no. of source codes newly committed in each MVC layer

(D)Extraction of support

(E)Extraction of leadership

**Functions for extraction of Development features**

(C)Extraction of initiative

(B)Extraction of contribution

(A)Extraction of development area

Extraction of development areas]

**Filename** **Development area** **Function** **Role**

**Database access**

**Model**

**Database**

**engineer**

**xxx.model**

**xxx.html**

**xxx.control**

**UIOutput**

**Coder**

**Front end**

**engineer**

**Event Handler**

**Controller**

**View**

Conclusion:

By means of experiments using real Git projects on GitHub, method

Has been developed that allows us to extract the characteristics of software developers with regard to “strong development areas (M/V/C)”,“contribution”, “initiative”, “support”, and “leadership”

based on an MVC framework analysis of a Git project.

Paper:2

Title:No Single Metric Captures Productivity

Author: Ciera Jaspan, Caitlin Sadowski

Year:2019

* Earlier attempts have been made to measure productivity based on lines of code (LOC).
* **Why Do People Want to Measure Developer Productivity?**

1. one possible motivation for measuring developer productivity is identifying high/low-performing individuals and teams
2. other motivations include surfacing global trends across a company, rating the effectiveness of different tools or practices, running comparisons for an intervention meant to improve productivity, and highlighting inefficiencies where productivity can be improved.

* **What’s Inherently Wrong with a Single Productivity Metric?**

1. **Productivity Is Broad**

* When we create a metric, we are examining a thin slice of a developer’s overall time and output. Developers engage in a variety of other development tasks beyond just writing code, including providing guidance and reviewing code for other developers, designing systems and features, and managing releases and configuration of software systems
* Even for the narrow case of measuring productivity of developers in terms of code contributions, quantifying the size of such contributions misses critical aspects of code such as quality, or maintainability.

1. **Flattening/Combining Components of a Single Aspect Is Challenging**

* flattening all of these into a single measure along with quantity has limited applicability and risks, reducing the action ability of a metric. Is a developer with few code contributions of very high quality more or less productive than a developer with many contributions but some quality issues?
* flattened metrics may not make intuitive sense and so may be distrusted or misinterpreted. For example, if a variety of factors (e.g., cyclomatic complexity, time to complete, test coverage, size) are compressed into one number representing the productivity impact of a patch.

1. **Confounding Factors**

* Even if we are able to tease out a single metric that holistically covers some aspect of productivity, confounding factors can make the metric meaningless. Take the case of comparing programming languages. It is difficult to measure the productivity of languages in particular because of the number of confounding factors. There is the language itself, the tools, the libraries, the culture, the types of projects, and the types of developers who are attracted to that language.
* There can even be *externalities* that are not captured within a metric. For example, one team might appear to be submitting fewer lines of code than another team. There are many possible causes for such a difference that do not mean the team has lower productivity; perhaps the team is taking more steps to improve quality and therefore has fewer bugs down the road,

**Conclusion:**

* There is no single productivity metric for software engineers.
* Instead, focus on a set of custom metrics targeted to a specific question.