Analyzing HPC software Git repositories to identify and compute software productivity metrics

Kanika Sood

University of Oregon

August 18, 2017

Supervised by Anshu Dubey, Boyana Norris, Rinku Gupta and Lois McInnes
IDEAS mission

Improve software productivity and sustainability for computational science

- **Software Challenges:** Exploit massive on-node concurrency and handle disruptive architectural changes while working toward predictive simulations that couple physics, scales, analytics, and more.

- **Approach:** Collaborate to curate, create, and disseminate software methodologies, processes, and tools that lead to improved scientific software.

Methodologies to improve software quality *and* achieve science goals.

www.ideas-productivity.org

Slide courtesy https://ideas-productivity.org/
Motivation

- Software productivity, reusability → important
- Traditional metrics → give limited insight
- Higher project complexity → difficult to estimate team productivity and project maturity

Gain

- Opportunity to reduce cost and increase scientific output
- Support for future project planning and funding projections
Introduction

Software productivity - the effort, time, and cost for software development, maintenance, and support

- Scientific software is rapidly growing in capabilities, accuracy, performance.
- Software productivity has received insufficient attention.
- We analyze the correlations between project issues and characteristics using traditional metrics.
- We propose new time-dependent metrics that can help quantify productivity.
- The metrics can be used to better understand the trends of software development workflows and provide objective measurements of productivity.
- We demonstrate our approach on ACME, PETSc, MOOSE, YT and SPACK.
Methodology

1. Fetch Github/Bitbucket data
2. Parse Json
3. Categorize issues¹
4. Analyze issues with standard metrics
5. Create metrics and correlate them
6. Test on scientific software projects

¹Github/Bitbucket Issues: A way to keep track of project tasks, enhancements, and bugs
Examples: bug reporting, new feature suggestion
Why is quantifying productivity hard?

Formatting and word replacement can change lines of code (LOC)

Current metrics like LOC are not very useful for quantifying software productivity
**Metrics we use**

- **Traditional metrics**
  - Weekly commits and additions for the project lifetime
  - Number of issues reported
  - Category of issues

- **New metrics**
  - Monthly bug fix rate
  - Monthly feature request rate
  - Correlation of the number of issues with project age
  - Total commits and additions
  - Number of followers and watchers
  - Cumulative bugs and cumulative fixes
  - Number of open and closed issues

**so far...**
Traditional metrics - I

How actively are developers participating in development?

MOOSE

![Developer weekly data for number of deletions and additions for person A](chart1)

![Developer weekly data for number of deletions and additions for person B](chart2)

![Developer weekly data for number of deletions and additions for person C](chart3)

![Developer weekly data for number of commits for person A](chart4)

![Developer weekly data for number of commits for person B](chart5)

![Developer weekly data for number of commits for person C](chart6)
How often is the issue tracker employed by the users?
Traditional metrics - III

What are the most common tags associated with issues?

- Are bugs and issues among the top 10 tags?

Top 10 tags used in ACME:
- Bugs
- Enhancement
  and others…
New metrics - I

How interested is the community and what is the potential for expanding capability?

MOOSE

SPACK

PETSc

*PETSc mainly uses emails for open and closed issues (Next talk)
New metrics - II

- Is a project under-resourced for user support and/or development?

**MOOSE**

Cumulative bugs and cumulative fixes for MOOSE

**SPACK**

Cumulative bugs and cumulative fixes for SPACK

**YT**

Cumulative bugs and cumulative fixes for YT
New metrics - III

Do critical issues get resolved quicker than non-critical issues?

ACME
**New metrics - IV**

- How many bugs and features have been requested over time?

**ACME**

Monthly bug fix and features requested rate for ACME

- **MOOSE**

Monthly bug fix and features requested rate for MOOSE

- **PETSc**

Monthly bug fix and features requested rate for PETSc
New metrics - V

- How much effort is spent in resolving issues?

ACME
Issues in ACME

MOOSE
Issues in MOOSE

SPACK
Issues in SPACK

New metrics - V

- How much effort is spent in resolving issues?
New metrics - VI

- Does the number of issue requests correlate with the different ways in which a project can be used?

* PETSc is excluded because it has issue tracking in the form of emails as well.

Circle diameter: No. of forks
Fork: A copy of repository
Watchers: Active developers or users

Bigger circle: More customized use of project
Smaller circle: More ‘as is’ use of project
Summary

- Analyze the correlations between project issues and characteristics using standard metrics.
- Propose new time-dependent metrics that can help quantify productivity.
- Demonstrate our approach on ACME, PETSc, MOOSE, YT and SPACK.

Future work

- Explore derived (more complex) productivity metrics.
- Integrate information from developers with these analyses.
- Better understand the trends of software development workflows and provide objective productivity measurements.
- Combine Git analysis with email analysis.
Questions ?

Thank you
Outline

- Mission
- Introduction
- Motivation
- Standard metrics
- Metrics we compute
- Methodology
- Results
IDEAS mission

- Move scientific software development toward an approach of building new applications as reusable and scalable software components and libraries using the best available practices.

- Develop and demonstrate new approaches for producing, using and supporting scientific software.

- Establish methodologies that facilitate delivery of software as reusable, interoperable components.
SPACK, PETSc, YT, ACME, MOOSE

- PETSc: Portable Extensible Toolkit for scientific computation is a numerical software library [1] that offers a collection of linear and non-linear solvers and preconditioners for scalable solution of scientific applications. It is one of the most widely used parallel numerical library.


- MOOSE: The Multiphysics Object-Oriented Simulation Environment is a finite-element, multi-physics framework [3]. It aims to make predictive modeling accessible and scalable, for nuclear engineering problems.

- ACME: The accelerated climate modeling for energy project [4] applies advanced climate and Earth system models to solve the most challenging research problems related to climate-change. The goal is to build a modeling system that can be used efficiently on the next generation of computing systems.

- YT: A multi-code analysis toolkit for astrophysical simulation data. It is a community-developed analysis and visualization toolkit for volumetric data
References


