Summer 8-14-2015

# Software Metrics and Dashboard

Shilpika Shilpika
sshilpika@luc.edu

George K. Thiruvathukal
*Loyola University Chicago*, gkt@cs.luc.edu

Saulo Aguiar
*Loyola University Chicago*, saguiar@luc.edu

Konstantin Läufer
*Loyola University Chicago*, klaeufer@gmail.com

Nicholas J. Hayward
*Loyola University Chicago*, nhayward@luc.edu

# Software Metrics Dashboard

**Shilpika, Saulo Aguiar, Dr. George K. Thiruvathukal, Dr. Konstantin Läufer, Dr. Nicholas Hayward**

sshilpika@luc.edu, saguiar@luc.edu, gkt@cs.luc.edu, laufer@cs.luc.edu, ancientlives@gmail.com

## About Software Metrics

- Computed from one or more measured values
- A critical tool that provides continuous insight to products and processes
- Helps build reliable software in mission-critical environments.
- The two types of metrics relevant to our work are *Complexity metrics,* intrinsic code properties like code complexity. *In-process metrics,* information that can provide insight into the underlying software development process.

## Approach

1. Evaluate whether CSE teams find the metrics dashboard useful.
2. Evaluate the effect of the metric dashboard on software quality and software process.
3. Implement dashboard based on metrics derived from information collected by the tools (GitHub and Bitbucket) used by projects.
4. We add new metrics as they become necessary.
5. We use Apache Spark, a cluster computing platform which serves as a general purpose engine for large scale data processing.

## Preliminary Results

The metrics dashboard is developed using modern web development methodologies like **Spray** in **Scala** which provides client-server side REST/HTTP support on top of Akka.

The metrics dashboard will continually compute the metric values and expose web services that allows developers to obtain metrics reports various textual and graphical formats.

We use Apache Spark, a cluster computing platform which serves as a general purpose engine for large scale data processing.

## Conclusions

Building appropriate sets of metrics, presented in a useful way, can prove beneficial to CSE software teams, large, small and solo.

Facilitating the production of quality software would be a key component for developing and sustaining CSE software, especially as other mission-critical projects grow to depend on it.

In this project, we take steps to introduce a pragmatic set of metrics into CSE software projects by conducting surveys, building a metrics dashboard, and performing analysis and post-surveys on selected projects.

For interactive exploration of Metrics information and reduction in computation overhead the datasets are partitioned into clusters in a distributed environment which introduces concurrency and independent failures/recovery of partitioned tasks.

## Work Plan

The task requires us to perform the following activities:

1. Assess how metrics are used and which general classes/types of metrics will be useful in CSE (Computational Science and Engineering) projects.
2. Develop a *metrics dashboard* that will work for teams using sites like Github, Bitbucket etc.
3. Assess the effectiveness of the dashboard in terms of project success and developer attitude towards metrics and process.

## Code Base

- https://github.com/LoyolaChicagoMetrics/loyolachicagometrics.github.i
- https://github.com/sshilpika/metrics-dashboard-commit-density
- https://github.com/sshilpika/metrics-dashboard-storage-service


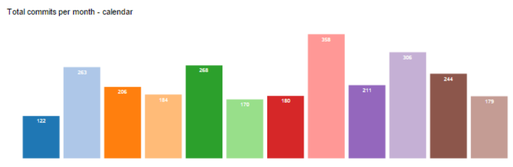


***Figure 1. Total commits per month (django/django) 03/2014 - 03/2015***

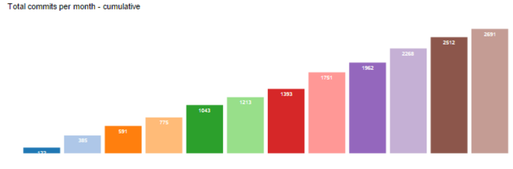


***Figure 2. Total cumulative commits per month (django/django) 03/2014 - 03/2015***

**Defect Density** = Number of Defects / Module Size

Defect density for a given project in a Github repository is calculated as

$\sum$ **Issues** / $\sum$ **KLOC**

$$\sum \textbf{KLOC} = \sum_{f(i)\ \forall\ i=(1\text{-}n)} \sum \mathrm{KLOC}_{(f(i),m(j))} = \sum_{f(i)\ \forall\ i=(1\text{-}N)} (\mathrm{KLOC}_{(f(i),m1)} + \mathrm{KLOC}_{(f(i),m2)} + \mathrm{KLOC}_{(f(i),m3)} + \ldots + \mathrm{KLOC}_{(f(i),m(curr))})$$

| Month (2015) | Defect Density | KLOC | Issues |
|---|---|---|---|
| February | 0 | 0.027 | 0 |
| March | 4.69 | 0.425 | 2 |
| April | 8.68 | 0.576 | 5 |
| May | 12.15 | 0.576 | 7 |
| June | 15.62 | 0.576 | 9 |
| July | 13.88 | 0.576 | 8 |
| August | 8.68 | 0.576 | 5 |

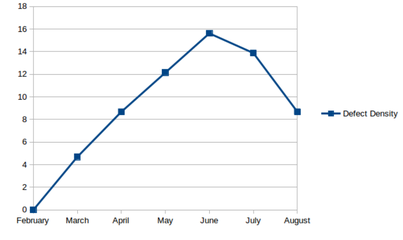


***Figure 4. Defect Density per month (sshilpika/metrics-test) 02/2015 - 08/2015***

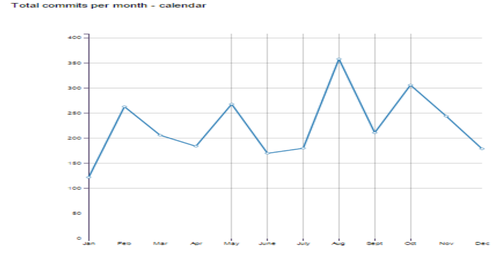


***Figure 3. Total commits per month (django/django) 03/2014- 03/2015***

## Future Work

We will ensure the metrics dashboard is properly instrumented to allow actual usage of the tools to be determined as projects collectively take advantage of them.

We plan to identify a set of metrics that are helpful to our own project (and for dissemination beyond scientific software teams in the future).

We will work toward a plugin framework, so teams can extend the dashboard with additional metrics we have not implemented yet.

Year 2 will end with a formal release of the metrics dashboard and online user manual.

## Bibliography

- Fenton, Norman and Bieman, James *Software Metrics: A Rigorous and Practical Approach*, Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series.
- http://doc.akka.io/docs/akka/snapshot/scala.html, Akka Scala Documentation
- https://github.com/mbostock/d3/wiki, D3 JavaScript library
- http://spray.io/introduction/spray-for-web-development/, Spray toolkit for Scala and Akka

**Acknowledgements:** This material is based upon work supported by the National Science Foundation under Grant No. 1445347.