Search Parasparam 2020

Parasparam 2020

View All Results ()

V

HOME
FAQS
MESSAGES
NEWS

HOME
FAQS
MESSAGES
NEWS

5
POINTS

Clear Vote

Find Teammates

Submission Tea ı + Add/Edit

R A D V

Details

Points:	5
Votes:	5
Rank:	1
Unique Views:	44
Total Views:	110
Comments:	0
Favorited:	1

Linked Submissions:

No Linked Submissions

Request To Join Team

Subscribe

Hindsights – An analytics based framework for software development (D899)

Team - Rupesh Shantamurty Submitted: Jun 7 2020 Category: Big Data and Analytics

Status: Pending

analytics data science graph mining machine learning sdlc

□ □ □ □

Description | Attachments (3) | Votes

While doing software development a lot of code gets written and modified. Once the product matures engineers realize that some parts of the software should have been architected and designed differently. With the ever changing structure of development teams it is difficult to retain the context of most of the changes done along the way. So the challenge development teams face is in identifying the components of the software that should be undertaken for redesign or change. Another challenge is to also identify the set of developers who would be apt for making these changes. To address these challenges we are proposing a framework in this paper which analyzes the trends in development lifecycle and presents the components that should be taken up for change. The framework also helps to identify the key contributors for specific functionalities of the software. We call this framework Hindsights since it is based on the analysis of the events in the SDLC of a product. This framework will also assist software architects and product managers to make better decisions towards implementation of new features as well. Better decisions save time and expedite the product development lifecycle.

Expertise Required

Agile Software Development | Analytics | Machine Learning | Graph Mining

What problem are you addressing with your project?

We at HPE write good software and that is the reason our software lasts for many years and sometimes lives for decades together. Such matured software undergoes multiple rounds of rearchitecting and redesign over the years. This also leads to increased set of functionalities in the product. With high focus on the quality of products delivered by HPE and the increased set of features it becomes difficult to identify where to make the changes. Hence it becomes critical to decide on selecting the right components that need to be chosen for the redesign and rearchitecting. The challenge is in identifying the components that are the trouble makers and that need change. Another challenge is to have enough backing data and justification for choosing the component that is being taken up for redesign. It becomes even more challenging to identify the set of developers who should be involved in these transformations.

What is the solution you are proposing

In this paper we present a framework which consumes the revision control historical data along with code review inputs from developers and performs trend analytics on the captured data and presents it in a manner that is easily consumed by the decision makers. With this innovative way of presenting the data the decision makers will be able to take decisions relating to identifying the components which have undergone the most changes and hence an obvious targets for redesign.

An interactive dashboard can be generated for identifying various trends like:

- Most altered modules represented as a heat map of the modules modified against set of all modules
- Most frequently changed modules
- Changes to a module requiring changes to many other modules representing loose / tight coupling between modules
- Using graph mining techniques many developer related parameters can be identified like
  - Developers who had the most influence in the terms of the breadth of functionalities that they got involved in.
  - Set of developers who had the most collaborative development of specific functionalities.
  - Developers whose modules underwent least number of modifications

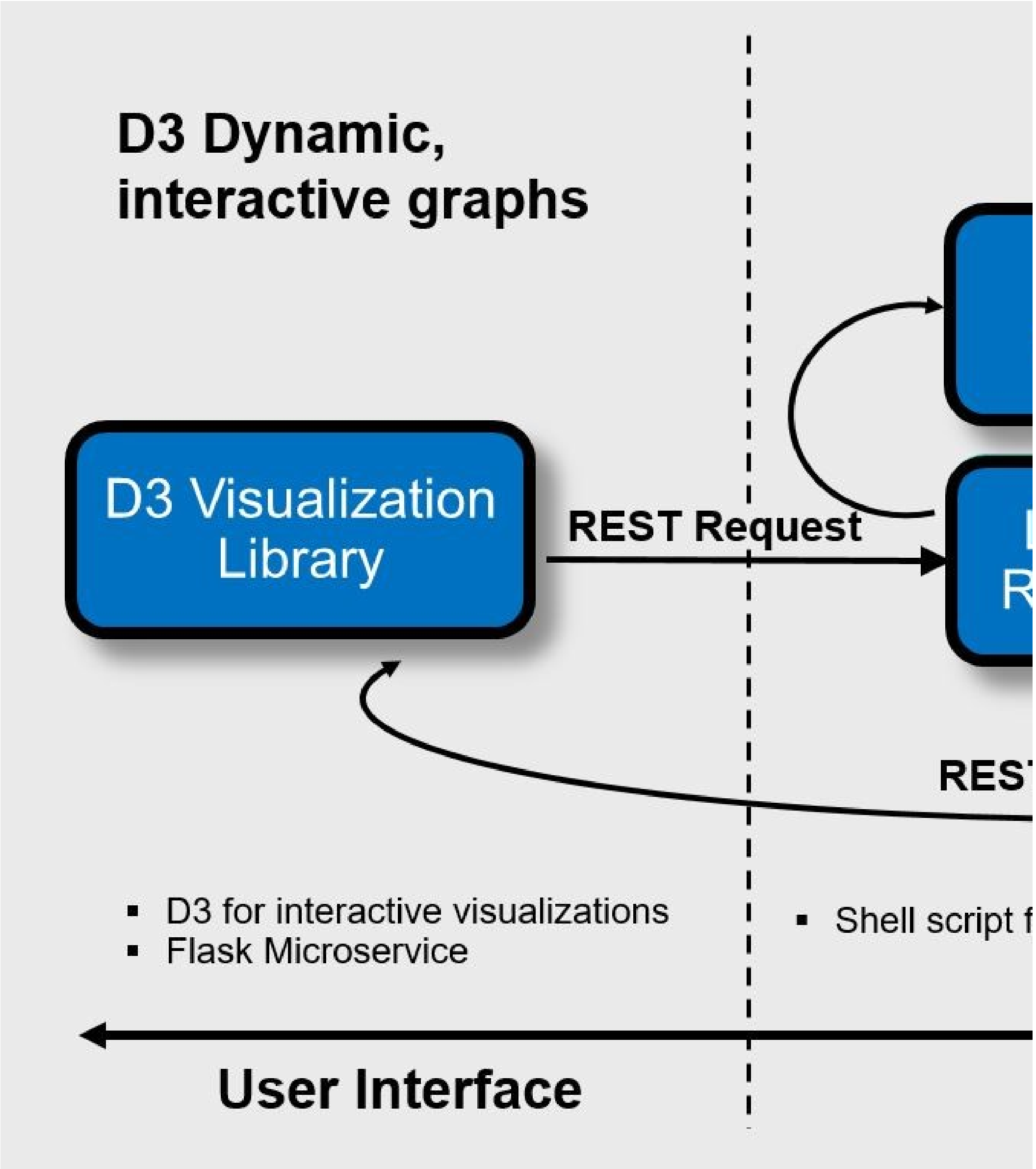

Figure 1. Implementation of Hindsights

Above figure 1 shows the basic design of our framework. Visualization is done using D3[1] and the backend uses a combination of scripts and gitpython[2] library.

The code is generally organized in various folders/packages. The packages generally represent the implementation of a specific functionality. These represent the software functionalities. The dashboard will be interactive in the sense that we are able to zoom in from a root view and drill down to sub levels of code hierarchy and are able to identify the final leaf components that got modified. The final leaf level modification is represented in term of number of lines changed through various commits in the git repository.

For graph mining we are working on utilizing the networkX[3] library to arrive at the various developer related mapping information.

What is the evidence that solution works

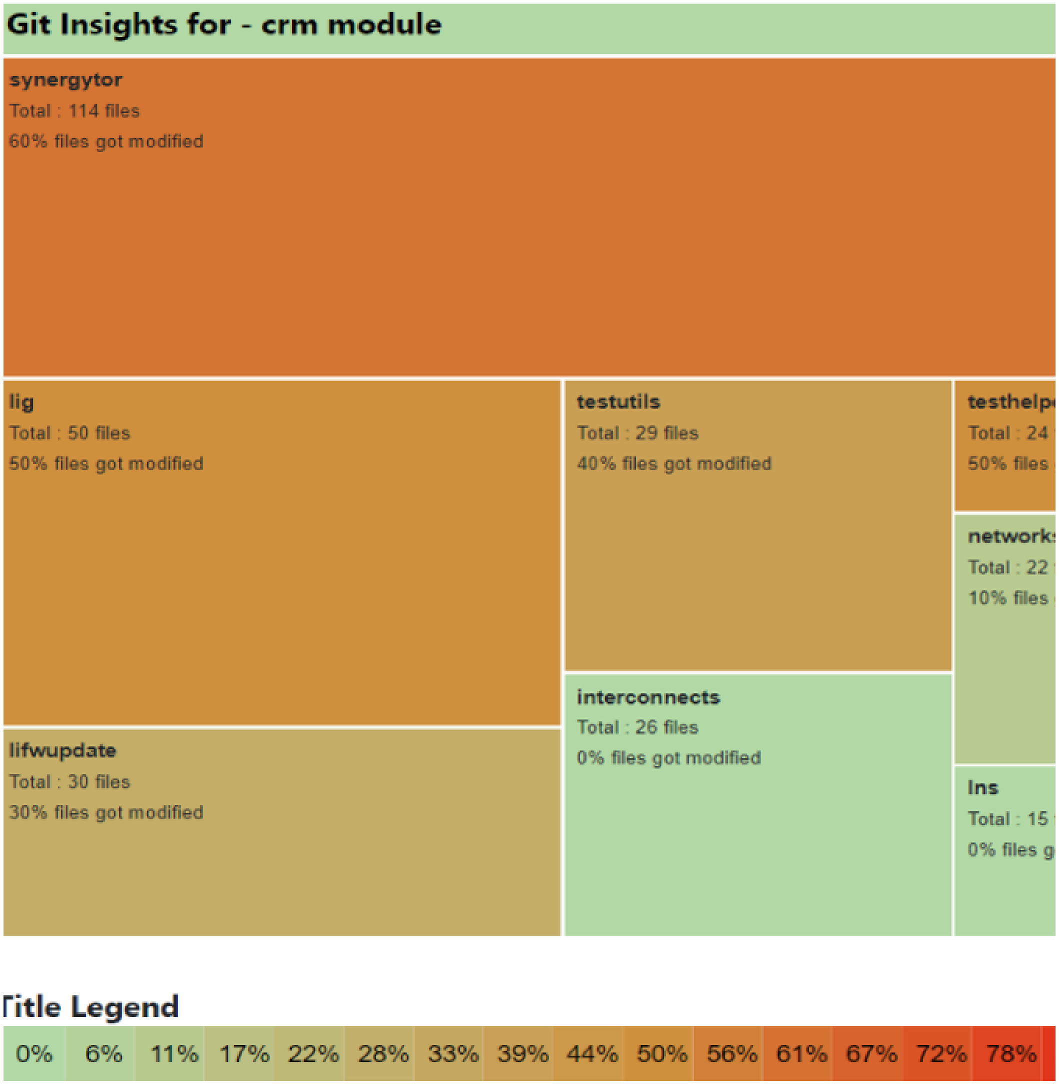

Figure 2. Implementation of Hindsights showing heat map of code changed between two dates

We were able to quickly plug together the various modules as explained in the solution above and present a heat map of the changed module within two given dates. The figure 2 above is a screenshot of the implemented proof of concept. The above dashboard is clickable and that makes it interactive. One can double click and drill down to the file level where the dashboard represents the lines of code changed and the corresponding git commit information.

What are the other competitive approaches ?

We found two instances where people have attempted to solve the problem presented here. One is GitStats[6] – This tool visually presents the activities within a git repository. Our solution is different from GitStats in terms of not only presenting the data in a better interactive manner but also having capability of custom trend analytics. Other prior art is a patent[7] titled Software development automated analytics by Microsoft corporation. Our solution is different from this in terms of a simpler solution also the patent does not mention about the capability to customize the framework based on the business need. Our solution is customizable based on the requirements.

For graph mining we investigated networkX, snap[4] & igraph[5] libraries and found that networkX and snap scaled better for our use cases.

What is the current Status

Prototype ready

What are the next steps

We plan to provide this solution as a service and incorporate capability to generate more dashboard to show

1. Frequently altered modules represented along with the frequency of changes

2. Display modules that got changed together so as to identify the coupling between modules

3. Implement graph mining techniques using networkX and identify the degree distribution and pairwise node similarity so that we can identify developers for the parameters that we have stated in our solution.

We also plan to solicit inputs from the larger development and management community and incorporate more dashboards.

Acknowledgements

References

1. D3 Visualizations https://d3js.org/ For frontend user interface

2. GitPython - https://github.com/gitpython-developers/GitPython For backend implementation

3. NetworkX - https://networkx.github.io/

4. Snap Stanford - http://snap.stanford.edu/

5. Igraph - https://igraph.org/redirect.html

6. GitStats - http://gitstats.sourceforge.net/

7. Software development automated analytics - https://patents.google.com/patent/US8745572B2/

Add Comment

Post Comment

2 Comments

Sort by: Newest

SUBMITTER Q&A

Mayukh Dutta
Jun 8 2020

The treemap/heatmap does not help to decipher the value of the solution. The treemap shows "more the number of files, greater the churn is" which is obvious. Can you please provide some more details that help understanding how analytics is helping here?

Rupesh Shantamurty
Jun 8 2020

The heat map also takes into account the number of times a file is getting modified. We have a mechanism to assign more weights compared to a file which got changed fewer number of times. As stated in paper these "red hot" modules "have undergone the most changes and hence an obvious targets for redesign".