GITHUB REPO EXPLORER

Sushant Chaudhari (sushant1@umbc.edu), Harsh Vashistha (harsh5@umbc.edu), Sushant Athley (su7@umbc.edu)

Abstract—Code repositories on Github.com contain metadata which describes its collaborators, commit activities and code additions/deletions performed by the collaborators over the lifetime of the project. The repository also contains issues and events which are related to those issues. Github.com comes with basic visualizations out-of-the-box for every repository which shows some basic patterns in this data. However, the visualization is distributed over places and we cannot compare the actions performed by different users. Currently the visualizations on GitHub page show data for different collaborators separately. This does not help us to identify what actions were performed by each user. It does show the additions and deletions in a week but provides no information about who performed those operations. So, in this project we came up with a design that summarizes all this data in a single visualization which can help us to identify the top collaborators in any repository and identify the actions performed by them. Our main aim here is to compare the top collaborators in any repository which is based on the commits, additions, deletions and issues solved by them. We also want to know a user's credibility based on what other repositories is the user collaborating. So, we show other repositories that are linked to the respective collaborator. The size of the shapes that represent these repositories are different as they are directly proportional to the number of 'star' and 'watch' for a repository.

Index Terms— Hierarchical edge bundling, collaborators, circle view, exploratory data analysis, circle segments technique

INTRODUCTION

GitHub.com is a code hosting platform for version control and collaboration. It lets you and others work together on projects remotely from anywhere in the world. It generates a huge amount of content related or real time data. The analysis of this kind of data is an important and challenging task, since analysts are interested in patterns, correlations or exceptions in the data, to identify bottlenecks, critical process states or any other interesting information hidden in the data.

In real time data streams like GitHub, it is important to analyze the data at a fast pace, usually on the fly, and present the results in a meaningful and intuitive way to enable the user to quickly identify the important information, and to react on critical process states or alarming incidents. Interfaces that involve human-computer interaction based on visual representation and analysis of large data sets has evolved rapidly in recent years. Presenting data in an interactive, visual form often fosters new insights, encouraging the formation and validation of new hypotheses to the result of better problem solving and gaining deeper domain knowledge. One of the problems with these data sets is the time dependency. This makes it more difficult to visualize and interact with such time dependent datasets. In this project, we present an approach to the problem of analyzing multidimensional time-referenced data sets called Circle View [1]. Our approach combines the popular visualization technique pie chart with a novel arrangement of the time events on circle segments. The independent variable is plotted around a circle in either a clockwise direction or alternatively in a counterclockwise direction. We have further divided the segments into concentric rings. The dependent variable which is usually a statistical value is visualized as a circle segment whose area is proportional to the magnitude of the statistical value [1].

The basic idea is to display the GitHub data dimensions as segments of a circle, and each segment is then divided into several sub segments to visualize the distribution and changes over the time.

Dividing the circle into concentric circles helps us to add more dimensions to the visualization. More space can be utilized and the data can be summarized in small space. The circle is partitioned into n segments, each representing one of the collaborators from the GitHub repository. Circle View displays enables both local detail and global context in a single continuous view. Circle View displays support detailed inspection tasks and, at the same time, help users to find interesting time event patterns. We show, that Circle View provide a unique possibility for exploring large time referenced data sets in a meaningful, intuitive way, not offered by other approaches. The circle view visualization approach takes advantage of both the computational and graphics display power of today's computers, and the flexibility, creativity, and general knowledge of human data analysts.

Rationale for the project (Why is this topic/work important?)

This project will help to address the following issues:

It will help us to view all data in a summarized manner in one place rather than the distributed visualizations on the GitHub website shown under different tabs. Users must go through different webpages to compare the collaborators and study the commit patterns. As the visualizations are not in a single space, it becomes difficult to compare the data.

This visualization helps users to view data of 30 weeks at the same instance. This creates a summary of such dataset that is distributed over a long-time which is easy to be compared as the users get a static image of the different types of data.

It will help us to visualize the relationship between different collaborators that have worked on the same issue. As of now the issues and events related to those issues are not represented using any form of visualization on GitHub. They are written in text format which is not a convenient way to represent such dataset. So, this project will certainly help to address this problem.

Related work

Work has been done in the past on Circle View - A New Approach for Visualizing Time-related Multidimensional Data Sets by Daniel A. Keim, Jorn Schneidewind, Mike Sips at University of Konstanz, Germany. The circle view design is motivated from three different disciplines which are mainly for designing effective visualization interfaces.

The Circle View interface enables the possibility to present real time data as well as historical data in a single intuitive visualization. To enhance the data exploration, the Circle View interface involves the human in the exploration process. Since Circle View is an approach for analyzing fast changing time dependent data streams, it is important to keep the visualization simple to avoid confusing the data analyst with too much information [1]. The data analysts can then select some interesting pattern or subsets for further investigation of the details or internal structure of the aggregated or summarized data based on personal interest.

Recursive pattern techniques [2] is one of the several pixel oriented techniques that can be used to visualize this type of time dependent

datasets. It is based on a generic recursive schema and aims at representing datasets having a natural order depending on one attribute. Time series data is one such example. With parameters for each recursive schema, recursive pattern techniques allow the user to control the semantically meaningful substructures which determine the arrangement of the attribute values. It visualizes each attribute in a separate sub window. Inside the sub window, each attribute value is represented by one colored pixel with the color reflecting the attribute value. The order of the objects is reflected by the same arrangement of pixels in each sub window such that it enable the user to relate attribute values of different attributes but at the same positions. The recursive pattern technique is based on a back and forth arrangement of the data attributes.

GOALS, RESEARCH QUESTIONS, HYPOTHESIS

Our main goal is to develop an interactive application that could help users to visualize GitHub's repository metadata in a summarized manner. This project introduces a new approach for visualizing multi-dimensional time-referenced data sets, called Circle View. The Circle View technique is a combination of hierarchical visualization techniques, such as tree maps, and circular layout techniques such as Pie Charts and Circle Segments [1]. The most important task is to compare the GitHub data changing its characteristics over time to identify patterns, exceptions and similarities in the data.

We also studied the circle segments technique from the Visual Data Mining with Pixel-oriented Visualization Techniques by Mihael Ankerst. The circle segments technique has been proposed for visualizing large high-dimensional datasets. The idea is not to represent different attributes in sub windows any more, instead the whole dataset is represented by a circle which is divided into segments, one for each attribute. Within the segments each attribute value is again visualized by a single colored pixel [2]. Overall the circle segments technique enhances the visual comparison of values. Users that are involved in a highly interactive exploration process based on different pixel-oriented techniques have reported the circle segments technique to be less tedious than other techniques since they have a "visual anchor point" in the center. However, it takes a more extensive usability test to draw conclusions about advantages or biases related to data. Therefore, circle view technique has adopted these features from the circle segments technique.

Circle view is an easy to understand form of visualization as it provides a simple design that most of the users that are not from the field of analytics can easily understand. To achieve this goal, Circle View provides an intuitive and easy to understand visualization interface to enable the user to swiftly acquire the information he needs. This is an important feature for fast changing visualization caused by time related data streams [1]. Circle View supports the visualization of the changing characteristics over time, to allow the user to observe the changes in the data. Additionally, it provides drill down mechanism and user interaction depending on the user demands for an effective exploratory data analysis [1]. There is also the capability of exploring correlations and exceptions in the data by using similarity and ordering algorithms.

Description of the implementation and environment(s)

For this visualization, we chose to use the circle view technique for our first prototype.

In this initial prototype**,** we implemented a basic version of the circle view. The circle was divided into several segments which represented the collaborators in that repository. We divided the circle into 52 concentric ring, each ring representing a week of the year. The most recent weeks were towards the outer ring as they are of more interest to the user. Each week displayed the number of commits made by the collaborator. But to make the problem more interesting we thought of including more dimensions to the visualizations. So, the initial prototype only showed the commit history of collaborators and no other data. In the final prototype, we decided to include other dimensions of GitHub data like the addition and deletion, along with the events related to issues in the visualization.

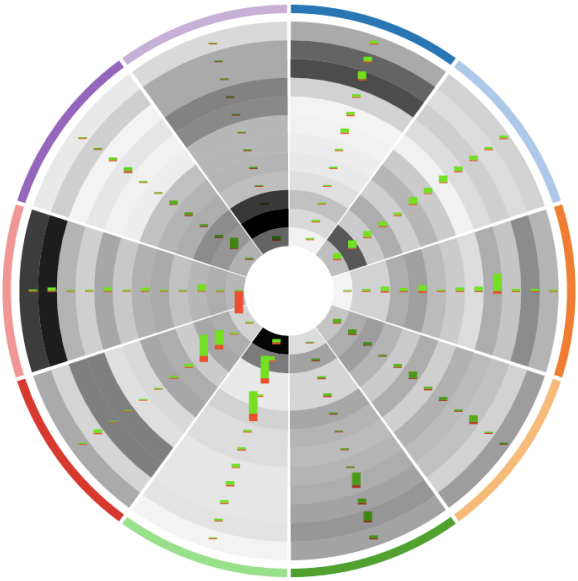

Fig (1) First prototype of GitHub Repo Explorer

For the next iteration in our project we modified the design implemented in first prototype. Fig. (2) shows the circle view implementation for the GitHub Repository explorer. As seen in the fig. (2) the circle is divided into 20 segments. Each of the segment represents a collaborator. The number of segments can be changed by the user. Here it displays only the top 20 collaborators in the repository. The collaborator that is chosen for comparison appears on the north side of the circle. The collaborators are ordered based on their commit history from right to left in the visualization. The outermost ring shows the commit pattern over the weeks. The ring just inside it is used to show the additions and deletions performed by a collaborator. The green bars that are above the axis represent the additions whereas the red bars below the axis show the deletions. This type of representation is very useful to show two data attributes on a single axis, which saves a lot of space. The innermost ring in the circle show the activities on issues in green and interaction on issues with fellow contributors in blue. The green bars represent the issues that are solved by the collaborator alone without any relation to another collaborator. Hierarchical edge bundling is present in the interaction links with opacity gradient displaying directions. Links are from assigner to assignees and from commenter to mentioned person. So, the darker side of the edge is the assigner and the lighter side of the edge is the assignee.

The circle view visualization can be scaled further to include maximum of 100 collaborators. Fig (3) represents a view where 100 collaborators are selected in the top collaborators from the repository over a period of 30 weeks of activity. The segments shrink dynamically to show more number of collaborators.

Circle view is not a good way to compare more number of collaborators as the visualization shrinks in size and does not effectively show the data. This type of visualization is suitable to compare only few of the top collaborators in a repository. As seen in the fig (4) only four collaborators are selected for comparison. It clearly represents all the dimensions and is very effective for

comparing the data. Here the *tenserflower-gardener* is chosen from the collaborators list to be compared with the other top 3 collaborators. It can be inferred from the visualization that *tenserflower-gardener* is the top collaborator as the number of commits is high and is a linear graph on the timeline. Also, the addition and deletions performed by this collaborator is very high as seen from the dominant green and red regions of the graph in the inner ring of the circle. More number of issues were handled by the '*vrv*' collaborator as there are more edges directed from this collaborator. In this way users, can easily analyze the data for any repository using the circle view visualization.

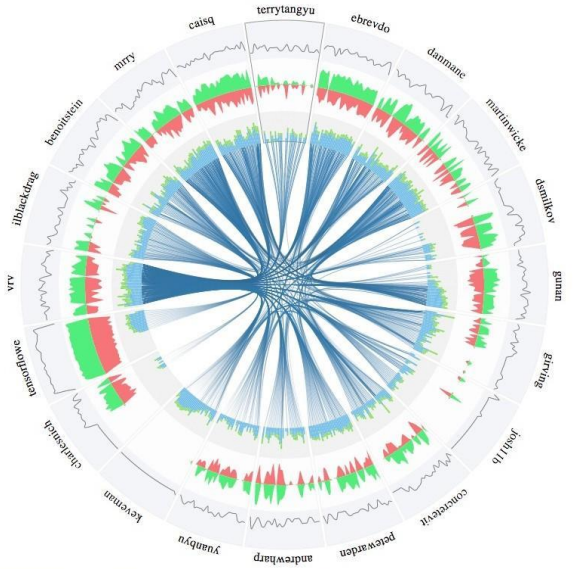

Fig (2) Circle view showing 20 collaborators

Design method:
Data abstraction:

- GitHub provides API to download data of open source repositories. So, we have downloaded the data using the GitHub API's and then used it as a source.
- Data type is static. Already collected and aggregated because of rate limiting by GitHub on public APIs.
- For this project, we got data of the collaborators that are contributing in different repositories of the GitHub.
- Data that tells the number of commits performed by each collaborator since the time they started contributing in a repository.
- Number of additions and deletions performed by the collaborators.
- Star and watch data for open source repositories.
- Data which represents issues and events related to those issues.

Task abstraction:

- One of our primary tasks here is to **Identify** the top collaborators in a repository.
- Another important task is to **Compare** the commit patterns for different collaborators
- Concentric segments **summarize** the data for commits, code additions, deletions and events related to an issue
- User can **search** by repository name and select the number of weeks from the search option given in the right side of the screen
- The hierarchical edges between the collaborators **produce** relationship between different events in an issue.
- The visualization **consumes** new data and modifies itself to show the newly added data. Every time a user changes the number of collaborators to be displayed, the visualization scales itself to show the newly consumed data.

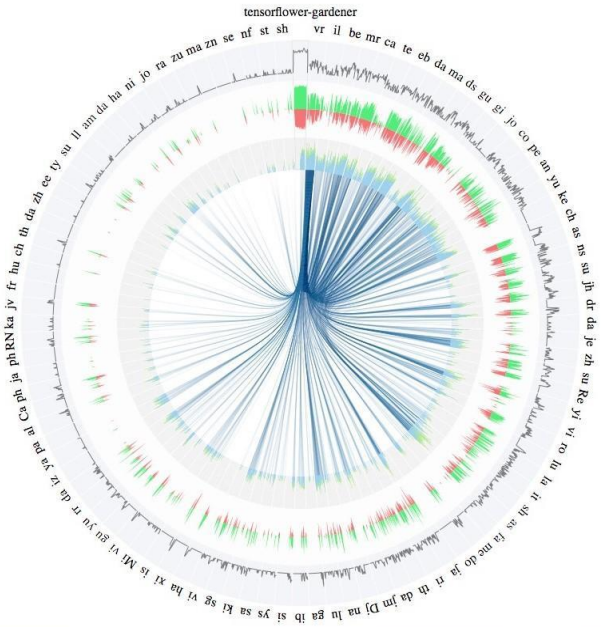

Fig (3) Max 100 collaborators selected for comparison

DESIGN CONSIDERATIONS

Circle view covers many aspects of design like the top-level view of any dataset. It highlights the important information in the repository. Circle view also provides some form of interactivity to the users. Another important aspect of circle view design is that it provides the details on demand and drill down. It can effectively display the continuous data stream displays in the GitHub dataset. Circle view focuses on the intuitiveness of the visualization method and does comparison of data stream attributes on a continuous timescale.

Another technique called the spiral visualization technique [1] for visualizing serial periodic data was studied, which avoids over plotting. But the spiral visualization technique has some shortcomings like in real world data analysis applications time related data sets are not necessarily periodic, and it is difficult to identify non-periodic correlations in such visualizations. So, this technique cannot be used for exploring the GitHub repositories.

Pixel bar charts [1] can be used for visualizing serial periodic data. It visualizes each data item without aggregation and allows the ordering of the data items belonging more than one attribute. But this approach does not address the highly dynamic properties of time-related data sets. Humans find it difficult to see multidimensional patterns and time correlations in these visualizations. In highly dynamic data sets, it

would be very useful to show the actual time pattern in more detail and, at the same time, to keep the history over all time events [1]. Techniques supporting clustering properties such as space-filling curves and recursive pattern [1] are only useful for data sets with 2D or 3D non-temporal semantics. It is not clear, how to visualize multi-dimensional time-related data sets. The problem here is the simple mapping to colored pixels. In high dimensional data sets, it would be very useful to see all dimensions in a single visualization. So, after comparing all these techniques we decided to go ahead with the circle view technique while retaining the useful features of other techniques.

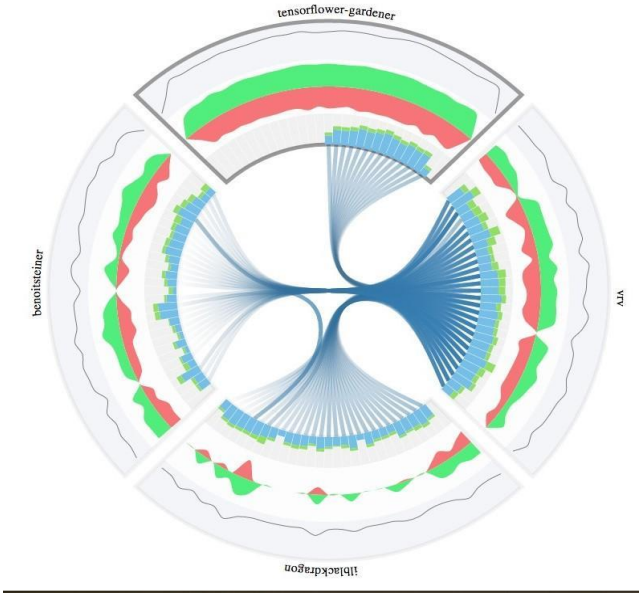

Fig (4) 4 collaborators selected for comparison

Implementation Methods

List of hardware and software you used to complete your implementation (displays, input devices, toolkits, languages, etc.)

Languages used for constructing the visualization are HTML, CSS, JavaScript and D3 library. We used the GitHub API to download the data. Data used was first collected and aggregated because of rate limiting by GitHub on public APIs. Sublime text was used as the editor and Google Chrome browser was used to run the HTML files on the server.

Result and analysis of results

As discussed in the above implementation part of the visualization the circle view visualization for the GitHub repositories is an effective way to compare continuous time series data in a summarized way. The circle view visualization takes very less space and represents mo dimensions in the small space. The hierarchical edges show the relationship between the collaborators working on the same issues in a simplified manner. This type of exploratory data Analysis often follows a three-step process [1]: Overview first, zoom and filter, and then details-on-demand. So basically, the analyst first obtains an overview. The user can thus use the search menu to select the repository and get details of the top collaborators. This may reveal potentially interesting patterns or certain subsets of the data that deserve further investigation. The analyst then focuses on one or more of these, inspecting the details of the data. Our Circle View provides an overview of the time related data set using a familiar and intuitive visualization. In this overview, the data analyst can identify interesting patterns or groups in the data. To focus on one or more of them, the data analyst can choose different time scaling functions to view the time events or phenomena in more detail. To get detailed access to the data, the data analyst can influence the layout on the circle segment. The goal was to get a reasonable clustering of all the data dimensions from the GitHub dataset and we succeeded in achieving our goal by implementing the circle view technique.

Legend plays a vital role here as it displays the selected collaborator in the right window which gives a description of all the collaborator activities as seen in the Fig (5). As seen in the Fig (5) the legend gives information about the contributor ID, number of commits, additions deletions, the single actor and multiple actor events involved.

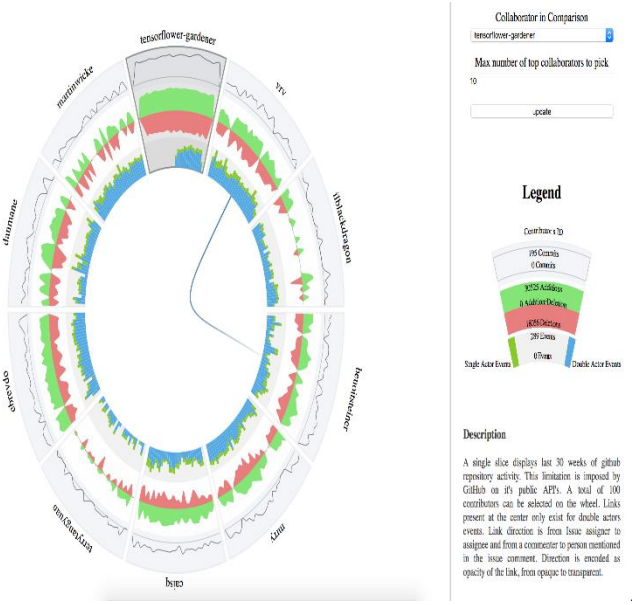

Fig (5) Circle view visualization with legend

Conclusion and future work

To conclude we have designed a GitHub repository explorer system for analysts to study and analyze interesting patterns in the repository. It helps to identify the top collaborators in the repository and study their commit patterns over a period along with the addition and deletion performed by them on the repository. We have combined the features of hierarchical visualization techniques, such as tree maps, and circular layout techniques such as Pie Charts and Circle Segments into the circle view visualization. Our application helps us to compare continuous data changing their characteristics over time to identify patterns, exceptions and similarities in the data.

Future work will include adding more dimensions to the circle view design. We can also dynamically change or expand the radius of the circle when the number of collaborators chosen for comparison increases. Currently the visualization does not give information on

what types of files were modified by the collaborators. So, in future we would like to address this issue.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Jian Chen for her thorough and timely feedback throughout the project timeline.

REFERENCES

[1] Daniel A. Keim, Jorn Schneidewind, Mike Sips, CircleView - A New Approach for Visualizing Time-related Multidimensional Data Sets

[2] Mihael Ankerst, Visual Data Mining with Pixel-oriented Visualization Techniques

[3] http://userpages.umbc.edu/~garba1

[4] ANKERST, M., KEIM, D. A., AND KRIEGEL, H.-P. Recursive pattern: A technique for visualizing very large amounts of data. In Proc. Visualization '95, Atlanta, GA (1995), pp. 279–286.

[5] ANKERST, M., KEIM, D. A., AND KRIEGEL, H.-P. Circle segments: A technique for visually exploring large multidimensional data sets. In Visualization '96, Hot Topic Session, San Francisco, CA (1996).

[6] CARLIS, J. V., AND KONSTAN, J. A. Interactive visualization of serial periodic data. In ACM Symposium on User Interface Software and Technology, San Francisco, CA (1998), pp. 29–38.

[7] I. POPIVANOV, R. M. Similarity search over time series data using wavelets. In ICDE 2002 (2002). [5] KEIM, D. A., HAO, M. C., DAYAL, U., AND HSU, M. Pixel bar charts: A visualization technique for very large multi-attribute data sets. Visualization, San Diego 2001, extended version in: IEEE Transactions on Visualization and Computer Graphics 7, 2002 (2002).

[8] SHNEIDERMAN, B. Tree visualization with tree-maps: 2-d space-filling approach. ACM Transactions on Graphics (TOG) 11, 1 (1992), 92–99.

[9] Y. ZHU, D. S. Statstream: Statistical monitoring of thousands of data streams in real time. In VLDB 2002 (2002), pp. 358–369.

[10] SHNEIDERMAN, B. The eyes have it: A task by data type taxonomy for information visualizations. In Proc. IEEE Visual Languages (1996), pp. 336–343.

[11] Keim D.A., Kriegel H.-P., Ankerst M.: "Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data", Proc. Visualization '95, Atlanta, GA, 1995, pp. 279-286