GitHub Explore - Discover Open Source Coding

Linya Huang
School of Art and Design
University of Illinois
Urbana-Champaign, Illinois,
United States of America
lhuang56@illinois.edu

Madhushree Sreenivasa
Master of Computer Science
University of Illinois
Urbana-Champaign, Illinois,
United States of America
sreenvs2@illinois.edu

Sharanya Bathey
Master of Computer Science
University of Illinois
Urbana-Champaign, Illinois,
United States of America
bathey2@illinois.edu

Tejala Thippeswamy
Master of Computer Science
University of Illinois
Urbana-Champaign, Illinois,
United States of America
thippes2@illinois.edu

ABSTRACT
The field of technology and programming is always changing and changing fast. This change can occur due to a new invention at an organization or by a user or the discovery of a new programming paradigm.

This paper provides a detailed design of an application built for the analysis and visualization of the latest trends and technologies used by organizations and users in the open source programming world using the most widely used open source hosting platform GitHub.

General Terms
GitHub, Organizations, User

Keywords
GitHub, Organizations, User

1. INTRODUCTION

As a programmer, it is always beneficial to keep track of the latest trends and technologies in the market today. For programmers new and old, open source projects are one of the easiest and fastest ways to learn about new languages and coding practices.

One of the most widely used platforms to host open source repositories today is GitHub. It is one of the fastest and ever growing platforms for hosting applications of users and organizations built through the Git version control framework. It provides a platform to find new and trending projects of users or organizations and allows one to start learning and contributing to those projects.

Most big organizations, such as Facebook, Google, LinkedIn, Amazon, etc., host their open source projects using GitHub. Many of the famous projects, such as the Linux kernal, d3, etc., are repositories now hosted, maintained and contributed to over Github.

The purpose of this project, as part of the course on Social Visualization at the University of Illinois at Urbana-Champaign, was to build an application to that uses GitHub as the source to analyze and visualize patterns and trends in the programming world along with allowing users to explore projects in languages, users or organizations of interest directly.

The remainder of this paper shows the background work, design, implementations and conclusions of this project.

2. BACKGROUND AND RELATED WORK

Social Visualization is about visualizing data found in social spaces. One of the most challenging aspects of designing a project is that of finding a data source in a short span of time. The most obvious choices for social data would be Facebook or Twitter.

The search for data and ways to materialize our project lead us to Facebook, Twitter, Spotify, Echo Nest, Nike+, data.gov and finally GitHub.

The first of our project ideas was to analyze the details of Nike+ and health statistics from data.gov. Obesity has become one of the leading health issues in the United States of America. We planned on using user activity details from Nike+, such as running, cycling, etc., users fitness goals and determine the trends across all the states of the USA. We wanted to compare with the health statistics from data.gov over the last 5 years.

The second was to create a mind map of a person through

their social network. Today, social networking sites have become the means of showing personality and interests. We wanted to use information from a persons Facebook and Twitter interests, page likes, following details, etc., to create a mind map of the person.

The third was to chart the music interests and trends of people and their friends across the world. The idea was to show how a person's music interests are trending across the globe. Whether their music interests match with those of their friends and also determine patterns across the globe.

The fourth and last idea was to analyze programming trends across organizations and the open source community. Many open source repositories are hosted either git or subversion version control framework. GitHub proved to be the best source to collate a lot of information about open source repositories. They have a rich set of API's avalilable which provide information about code repositories, programming languages used, followers, etc.

As programmers, learning about programming trends and open source software is the idea that seemed most interesting and useful. We then used the GitHub API calls through a browser plugin called Postman to analyze the different types of data that GitHub makes available to public. We used this data to design different types of visualizations for analysis and determined which would be best.

3. DATA ANALYSIS

GitHub provides API services through REST. All the GitHub API's can be found at https://developer.github.com/v3/

GitHub was founded in 2008 and the API provides all the information from then. The main idea was to visualize the increase/decrease in programming language use over time, to see the language change trends in organizations and to visualize a users usage patterns.

To visualize the trends in programming languages the API that needed to be called api.github.com/search/repositories with the language as a query parameter. This api provided all the information of repositories in that language from the year 2008.

Analyzing the organization and user details and repositories was done through 3 API calls. The first one was to fetch the basic details, such as profile information, profile picture, etc., from the api.github.com/orgs/:org for an organization and api.github.com/users/:user for a user.

We could then gather the details of public repositories through the API calls api.github.com/users/:user/repos for a user and api.github.com/orgs/:org/repos for an organization. This API call, lists all the repositories and gives you further API calls to see the details of each repository.

The language details across repositories was gathered using the api.github.com/repos/:user/:repo/languages for a user and api.github.com/repos/:org/:repo/languages for an organization.

Figure 1: Main page for the application.

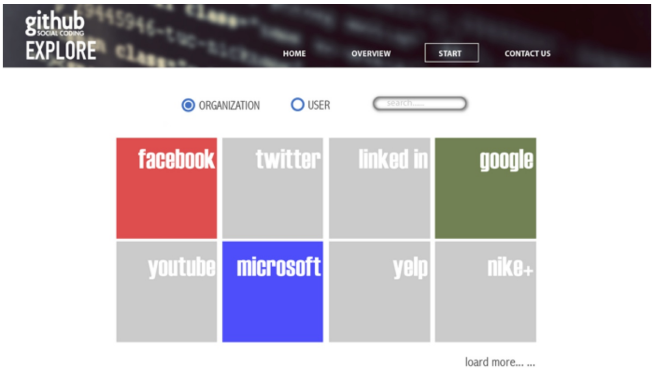

Figure 2: Main page for the application.

4. DESIGN

4.1 Overview

The initial design, shown in figure 1, started with a view to choose one of three things, organizations, users or languages.

We then decided to provide a view with some default options for the user to start off with as shown in figure 2.

4.2 Language View

In the view for a language we wanted to show:

1. The shift in language use over the years.
2. Which languages are currently being used the most.

Figure 3 shows a detailed view of different languages. The vertical axis depicting the languages. The horizontal axis is divided into 7 equal segments, one segment for each year from 2008 to 2014. The length of the bar within each year represents the use of the language during that particular year. This number/usage is based on the number of repositories in that language during that year across all public repositories of users and organizations in GitHub.

At the end of the whole design process we chose to drop this view due to time contraints and determined that the

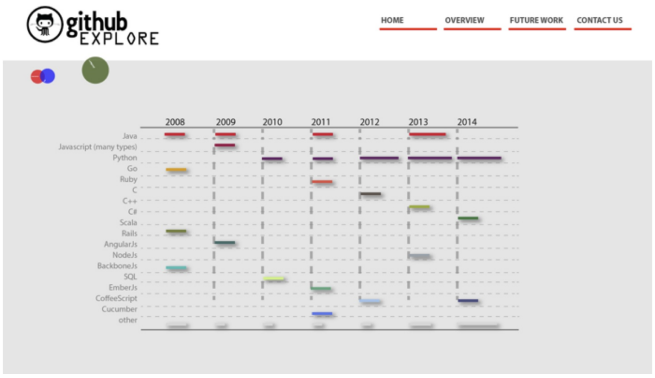

Figure 3: Language patterns view over 7 years.

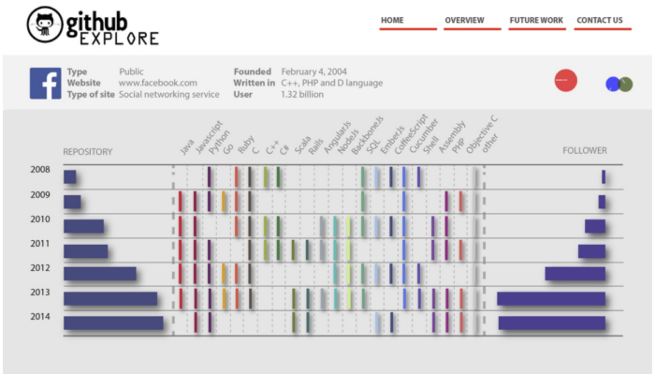

Figure 4: View 1.

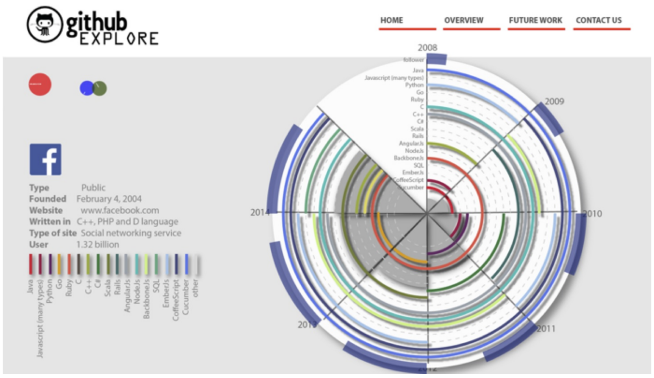

Figure 5: View 2.

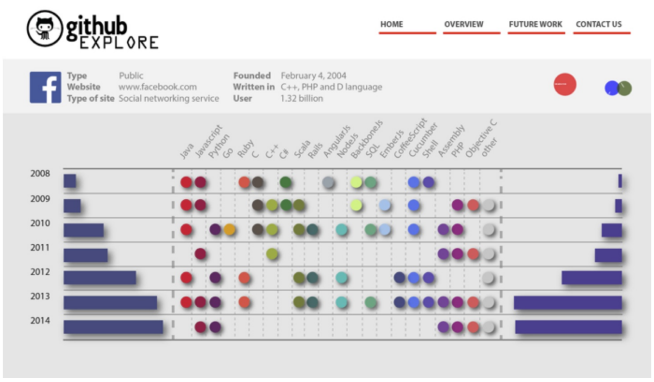

Figure 6: View 3.

user/organization view might help reveal more patterns than the language patterns analysis.

4.3 Organization and User View

In the view for an organization we wanted to show three patterns:

1. Visualize major companies using GitHub and how they have progressed over the years.
2. Most popular repository/project within the company.
3. The transition of programming languages in a company over time.

We had three designs to depict these three aspects. These are shown in Figures 4, 5 and 6.

Figure 4 shows the rectangle divided into 7 equal horizontal sections, one for each year starting from 2008 up until 2014. For each year, the number of repositories are shown on the left and the number of followers on the right as blue rectangular bars, the length of the bars indicating the number. The rectangular bars in between show the distribution of languages used in those repositories.

Figure 5 shows the circle divided into 7 equal regions, each pie representing one year beginning from 2008 up to the year 2014. The number of repositories are represented as the outer most thick blue arcs along the circumference. The inner shaded grey pies show the number of followers, the radius of the pie indicating the number. The arcs along circumference of all the inner sections represent the language distribution in that year.

Figure 6 shows the rectangle divided into 7 equal horizontal sections, one for each year starting from 2008 up until 2014. For each year, the number of repositories are shown on the left and the number of followers on the right as blue rectangular bars, the length of the bars indicating the number. The circles on the vertical axis show the distribution of languages used in those repositories.

One thing common across all the three views is that of the profile information of the organization. It displays the default icon of the organization as shown on GitHub, the day they joined GitHub, the GitHub organization link, the number of followers, the most widely used languages and if possible the type of organization they are.

In the view for an user we wanted to show three patterns:

1. Show a user's activity from the time they joined GitHub.
2. View the change in language usage in their repositories over time.

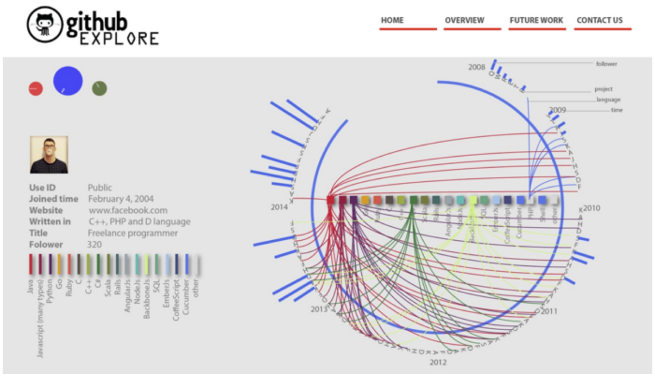

Figure 7: View 3.

3. View their followers/contributors over time.

A Single design was made for the user, which is as shown in figure 7.

The design was represented as a circle divided in to 7 equal regions, one for each year from 2008 to 2014. The alphabets along the outermost circumference arc representing the repositories. The length of the rectangular bars jutting out of the circumference for each repository indicating the number of followers. The square boxes in the center display the different languages. An arc or line goes out from each language square to repositories which have lines of code in that language. These arcs are in the color of the language as displayed through the legend on the left.

The left top corner displays the profile information of the user. It displays the default icon of the user as shown on GitHub, the day they joined GitHub, the GitHub user link, the number of followers, the most widely used languages and if possible the job description of the user.

One very important aspect we realized during the design and analysis of user and organization views is that though GitHub differentiates between the two they are essentially the same and what we wanted to analyze along these two categories is also the same.

We then reviewed our designs to have a single view for both users and organizations, the page to analyze for either is as shown in Figure 2. The views then designed

As pointed out by our Professor Karrie Karahalios, there are many possible problems with circular visualizations though they look very appealing to the human eye. One of the main problem in Figure 8 is that even thought each ring within the circle represents one year, the outer most ring looks to be more important than the inner rings. The problem with Figure 9 is also similar, the languages represented by the outermost circle seem to be more important than the languages represented through the inner rings. Also, there is a single view representing too much information for any user of the application to process. The design using Figure 10 was finalized. Design of the detailed view for a language was changed in order to show it through a representation

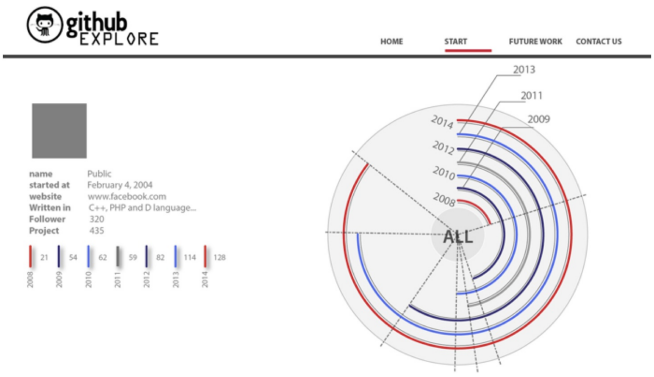

Figure 8: Overview page for a user or organization.

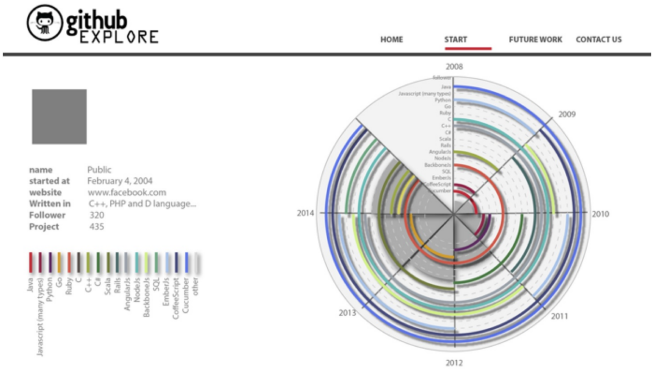

Figure 9: Details page for a user or organization.

Figure 10: Detailed view for any year for a user or organization.

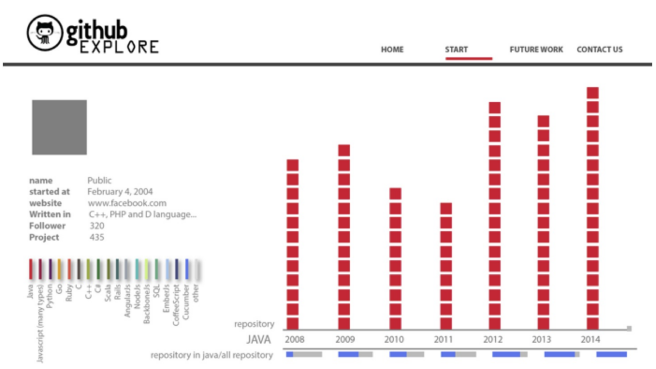

Figure 11: Detailed view for any language for a user or organization.

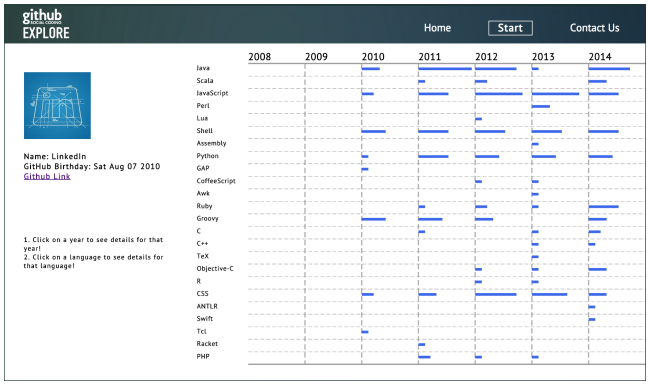

Figure 12: Final application overview.

different from the one found in Figure 10.

The final design for the application can be found in the implementation section of this paper.

5. IMPLEMENTATION

5.1 Overview

We setup a full stack web application to dynamically access and visualize data on GitHub by using the Python Flask framework in the backend and d3, JQuery and Javascript calls on the frontend.

Experimentation was done with several frameworks including Django and AngularJS. But, the final decision was to use Python Flask and AJAX based JQuery calls for data access through the API. This was mainly because the application was quite simple and most of the interactions would be simple HTTP GET operations. Using frameworks like AngularJS over complicated very simple things.

5.2 Python Flask REST Framework

Python Flask module was used to construct a REST API based service that would fetch the data from the GitHub API, parse and serve the data for different visualizations.

When a request is made for a particular user/organization, all the data required to display any part of the visualization in the application is fetched from the GitHub API dynamically and dumped into a JSON file. The data is fetched by making multiple requests to the GitHub API calls mentioned in section 3. Pagination was also taken into account in order to fetch all the results. This JSON file is loaded into a python script as a hash object and parsed to access the required information.

One of the most challenging aspects about processing the data was the normalization done for all the views for each of the visualizations. The aspects for normalization will be explained in more detail in the next section.

5.3 User Interface

The main page of the application is as shown in Figure 2. 8 default options are provided of the most famous organizations. A search bar allows for the user to enter a user or organization name.

One important design choice made during the implementation is that we considered differentiating with color only the top known languages. When we initially designed the application, one thing that we did not anticipate was the sheer number of languages that a single organization can code in or the number of repositories an organization can have in a single year.

Once a user or organization is entered, the default overview page is as shown in Figure 12. The data fetched for the yearly view needed to be normalized in order to fit on the screen with 80 pixel width for each year. The total number of repositories in that language for that year was scaled down to 80 and then displayed. Hovering over the bar will display the number of repositories in that year. The language on the y axis on the left and the year on the x axis at the top are both click able which would allow to navigate to the detailed view for the year or language.

Once a user clicks on any year the application translates to the detailed view for a year which is as shown in Figure 13 Each color corresponds to a particular programming language the legend for which is shown on the left. The length of the red rectangular bars on the bottom indicating the number of followers. The colored bars between the top most horizontal line and the the bottom followers section indicate programming languages used for each repository.

Normalization was done across the x-axis and the y-axis. Across the x-axis the size of each repository (number of lines of code) was normalized using min-max normalization to be between 5000 and 10000. The repositories were then normalized along the y-axis by scaling the number of lines in a particular programming language to the number of lines the repository was scaled down to.

Hovering over the follower bars shows the number of followers. Clicking on the dots along the x-axis shows the details of the repository, a link to find the more information about the repository. Transitions are visible through the application. For some companies like Google and Apache the number of

Figure 13: Final application detailed year view.

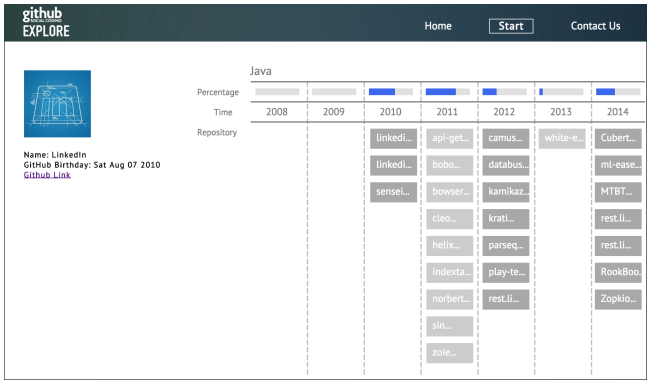

Figure 14: Final application detailed language view.

Figure 15: Detailed year view for 2013 for Microsoft.

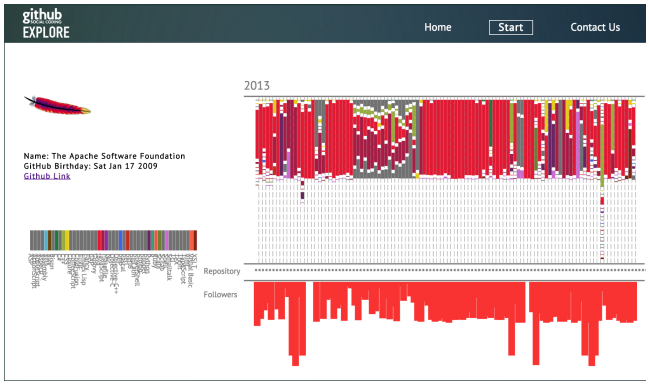

Figure 16: Detailed year view for 2013 for Apache.

repositories would make the visualization very cluttered.

In Figure 12 clicking on a language will display the detailed language view as shown in Figure 14.

The detailed language view shows the repositories in that language during each year. The rectangular grey sections with the blue portions display the percentage of repositories in that language during that year. Clicking on the repository will show the details of the repository and a link to explore the repository.

6. ANALYSIS AND PATTERNS FOUND

6.1 Organizations

We analyzed and visualized many organizations and users. In this section we will discuss some of those patterns.

We will first discuss the patterns found at LinkedIn. The overview for LinkedIn shows many repositories in JavaScript, Java and CSS. This indicates almost the full stack of development technologies that LinkedIn uses for their applications. LinkedIn started with open source repositories on GitHub from the year 2010. They moved from Java to JavaScript to Python from 2010 to 2014. Also, the number of repositories in Java during 2011 and 2012 were much higher than compared to 2013 and 2014. The results for LinkedIn are as shown in Figures 12, 13 and 14.

As mentioned before the colors used were for some of the most used languages. One of the features that GitHub provides is to show the most trending languages right now. We compared with our analysis for different organizations. For Microsoft, which started out on GitHub only in 2013, most of the repositories are in C# which is currently not trending on GitHub. This can be seen in Figure 16.

Another interesting pattern we found for companies like Apache, Google and Mozilla are that they have many open source applications. This aligns with the fact that Apache and Mozilla are open source foundations. Google is another company contributing heavily to open source software. A glimpse of what could be found at Apache is shown in Figure 17. From the figure you will notice that they mainly code in Java.

6.2 Users

Mbostock is another famous GitHub user. He is the inventor of D3. The analysis of his GitHub repositories ends in a lot of open source repositories in JavaScript. One of the patterns can be found in Figure 15.

Another very famous personality in the open source world is Linus Trovalds. His main contribution is the linux kernal. It is available on GitHub and as seen through Figure 18 all mostly in C. This was done in the year 2011.

Mitchell Hashimoto is one of GitHub's most active members.

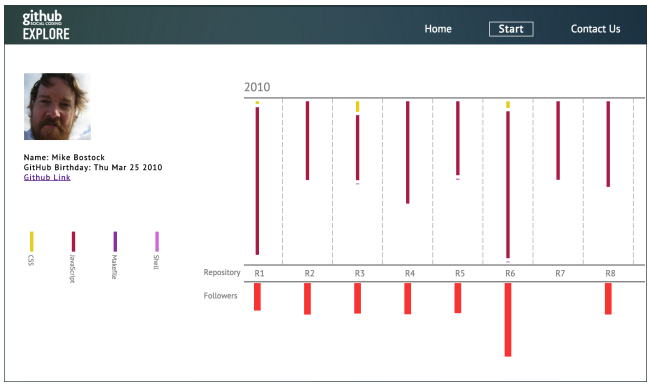

Figure 17: Detailed year view for Mike Bostock.

Figure 18: Detailed year view for Linus Trovalds.

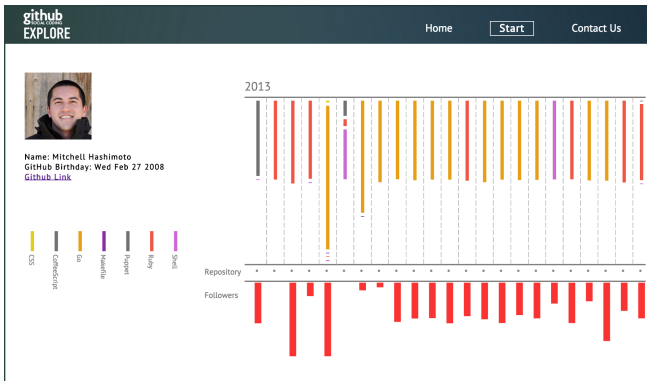

Figure 19: Detailed year view for Mitchell Hashimoto.

His contributions for the year 2013 can be seen in Figure 19. As you can see from this figure most of the contribution is in the new programming language Go. His coding has seen a transition from Ruby to Go, a lot of his earlier contributions were in Ruby.

The differences between users and organizations can be seen very clearly, even though the users are as famous as the organizations. Users GitHub accounts are generally sparse revolving mainly around the one main project that was a major success. But, for organizations there are many repositories each year garnering different amounts of interest. Trends can also be seen where there has been quite some change where repositories have moved from being in Java, C or C# to being in JavaScript, Ruby and Python which have become more famous in recent times.

7. CONCLUSIONS

As seen, there are many differentiating patterns between organizations and users. The trends within an organization can be seen clearly. The move for users from one language to another can also be seen clearly.

8. FUTURE WORK

First and foremost, the sheer number of repositories and languages found in a particular organization makes the visualization extremely cluttered. A way to improve this would be to provide an scrollable visualization. We can also group together programming languages into different languages such as scripting languages, interpreted versus compiled languages, etc. Though we cannot comment without exploring, if the legibility and purpose of the visualization will be lost by doing these changes.

Secondly, a visualization to compare two or more users and organizations can also be provided.

Thirdly, many open source repositories still exist on other version control systems like subversion and hg. Right now there is not platform such as GitHub which collates these version controlled repositories like. It would be a good exercise to analyze there repositories as well.

9. ACKNOWLEDGMENTS

We would like to thank Professor Karrie Karahalios and teaching assistants John Lee and Hyde Kong for the detailed feedback about the visualizations. This application would not have been possible without their insights into understanding and display of visualizations.

10. REFERENCES

1. Andy Cockburn , Amy Karlson , Benjamin B. Bederson, A review of overview+detail, zooming, and focus+context interfaces, ACM Computing Surveys (CSUR), v.41 n.1, p.1-31, December 2008

2. Maureen Stone, Choosing Colors for Visualization

3. Linton C. Freeman, Visualizing Social Networks

4. Karrie G. Karahalios, Tony Bergstrom, Social Mirrors as Social Signals: Transforming Audio into Graphics, IEEE, Computer Graphics and Applications, Vol. 29, No. 5 Sept./Oct. 2009