What Makes a Happy Developer?

Damian Kulp and Sean Marty

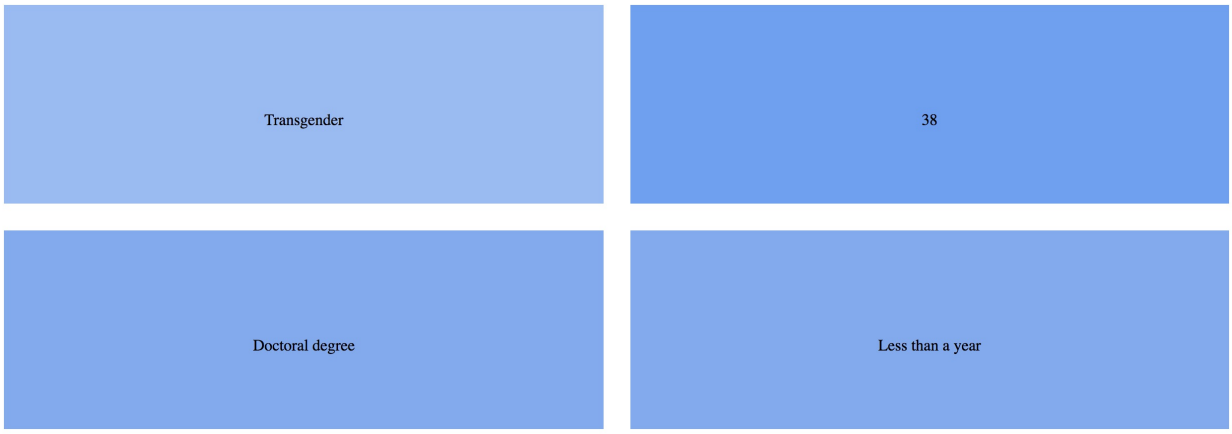

Fig. 1. A view of the landing page of our visualization.

Abstract—This paper contains relevant methods and results of Damian Kulp and Sean Marty's work on visualizing Stack Overflow Developer Survey data to aid graduating computer science seniors in understanding what career choices are more likely to result in high job satisfaction.

1 Introduction

This is the midterm project for CS 458, Info Visualization. We previously turned in a paper outlining our research, client interaction, and database setup for this project. This paper and corresponding visualization application build on the ideas we set up in our first paper. Our task, in a basic sense, was to find an interesting data set and client to create a visualization for. We needed to go through the process of working with our client to establish their needs and how we might help. Now, in this second portion of the midterm project, we made a first attempt at actually implementing our visualization. We decided to work with the data from the Stack Overflow 2017 Developer Survey. The goal was to compile the survey results and create a visualization that would be helpful for graduating computer science seniors looking for their first job. After talking with our client and exploring our available data, we developed a visualization to show the profile of a software developer that lends itself to the highest reported job satisfaction.

2 Method

In this section we will discuss how we created our visualization application, all the way from a raw data dump in CSV format to a functioning web application.

2.1 Tools and Libraries

There were a wide range of tools and libraries used to move from the raw Stack Overflow Developer Survey data to a final visualization. Here we will list the main methods we used, and how we used them.

- *Damian Kulp is an Oregon State University student.*
- *Sean Marty is an Oregon State University student. E-mail: martys@oregonstate.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

- SQL Server import tool: This tool was used to convert the Stack Overflow Developer Survey data CSV from 1 massive file to a single general database table. More information on the latest version of SQL Server can be found at *https://www.microsoft.com/en-us/sql-server*.
- Multiple scripts were used to break the single general database table into a set of normalized tables. This set of tables was selected based both on logical division of information and our specific needs for this visualization.
- Node.js: We used this JavaScript-based runtime environment as the basis for our project. The Node Package Manager (NPM) allows us to install and manage a range of packages within the visualization application. Using NPM, packages can be installed and set up for management with a single command such as "`$ npm install express --save`". More information on Node.js can be found at *https://nodejs.org/en/*.
- Grunt: The Grunt JavaScript Task Runner helped us manage our build sequence and more quickly and easily move from writing code to testing code on the local host. More information on Grunt can be found at *https://gruntjs.com/*.
- Express: We used Express routes to asynchronously serve data from the database to the client on a Node.js server. More information on Express can be found at *https://expressjs.com/*.
- On the client side, we used the JQuery AJAX handler to request data, save the response, and transform the received data into something we can use. More information on the JQuery AJAX API can be found at *https://api.jquery.com/category/ajax/*.
- For writing the client side application code, we kept things very simple. We used JavaScript, HTML, and inline CSS styling to maintain the most compact, readable code for the small scope of this project. For a larger or more complex application, we would

need to build out a more full-featured client side application. Information about JavaScript, HTML, and CSS can be easily found as they are extremely widely used.

- Plot.ly: We used the Plot.ly JavaScript package to quickly and easily render our individual characteristic data into SVG containers for display in the web application. Plot.ly has a wide range of features, but we kept it simple and just used the bar chart functionality. More information on the Plot.ly JavaScript package can be found at *https://plot.ly/nodejs/*.

- Microsoft Azure: We hosted our database as well as our web application on Microsoft Azure. Azure is a computing platform and set of web services that can do a wide range of things. We are currently just using a free version of the service to allow for a quick visualization application proof of concept. More information on Microsoft Azure can be found at *https://azure.microsoft.com/en-us/*.

2.2 Data Flow and Organization

As mentioned above, we started the project with a massive CSV data file that was downloaded from the Stack Overflow Developer Survey web page. We parsed the data into a single table, and began the work of organizing the data into a meaningful relational schema. There were two major ways that we broke the single large table into a usable form:

- First, we trimmed out the parts of the survey result set that were not of use to us. Data that was trimmed either did not pertain to our visualization (such as responses about how effective developers find Stack Overflow's website to be), or was unusable because it was not complete or not in a uniform format.

- Second, we created tables for each type of response that we wanted to be able to visualize, and a unifying reference table to tie those responses together with the common thread being job satisfaction.

Once the database was set up and organized, we needed to pull data related to each category of response we displayed on the visualization. Using JQuery's AJAX handler, we can request a sorted set of results for a survey response that averages job satisfactions for each response and orders those responses for easy identification of the top responses. An example of one of those queries looks like this:

```
select r.hoursPerWeek as HoursPerWeek,
avg(cast(r.jobSatisfaction as Float)) as
JobSatisfaction from Responses as r group by
r.hoursPerWeek order by JobSatisfaction desc
```

Getting our response data in this manner means we can control the amount of data that we are asking for, as well as get the return data in a format that can be easily transfered right into the Plot.ly bar graph format. For now, we are sending one request per response category we display. This might change later on as we complete the rest of the implementation.

3 Result

We implemented a first version of our visualization application that can be viewed in Fig. 2. As per the midterm project requirements, we developed the underlying structure of the visualization and a couple of our survey response pieces, but did not fully complete the visualization. During this development, we saw some of the value that our visualization could provide as well as found new ways to increase the usefulness of our application.

3.1 Data Results

Even with an incomplete implementation of the survey responses in our visualization, a couple of interesting characteristics of the data came out that both are useful in themselves, and of value when looking to improve and expand our visualization.

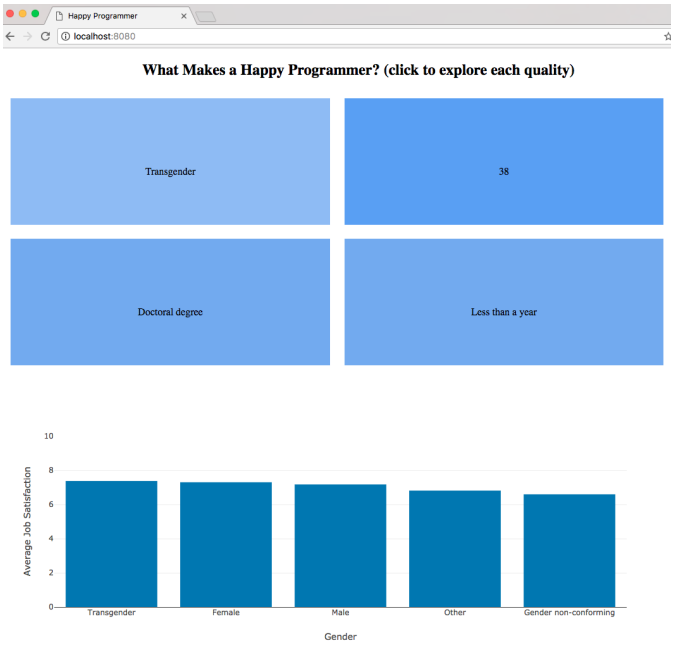

Fig. 2. A view of the whole visualization page in its current state.

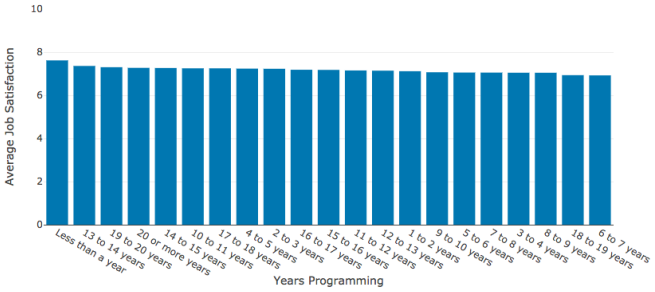

Fig. 3. The graph generated when selecting the Years Programming results section. Almost no correlation with job satisfaction can be seen.

- The most obvious initial result of the visualization is that there are some categories in the survey results that basically don't have any noticeable correlation with job satisfaction. This can be seen in Fig. 3. On the flip side, there are also some categories in the survey which correlate to great variation in job satisfaction, which can be seen in Fig. 4.

- In our visualization, we pull out the survey response that has the highest average job satisfaction associated with it, and display that in the application landing page. In some of our categories, the top and second place responses were separated by very small margins in average job satisfaction, and therefore might not be telling the whole story. There is very little difference in average satisfaction of 7.2/10 and 7.4/10.

- We obviously went into this project with some preconceived ideas about what responses would correspond with the highest job satisfaction. In some cases we were correct, but some of the categories were surprising. For example, it makes some sense that there is a cluster of higher job satisfaction near 40 hours per week, because those developers have full time jobs where they have chosen to work in the field full time and can find consistent work. On the other hand, there are some surprising data points, such as the fact that having a masters degree is worth an increase in average job satisfaction of less than .1/10 over only a primary/elementary school education.

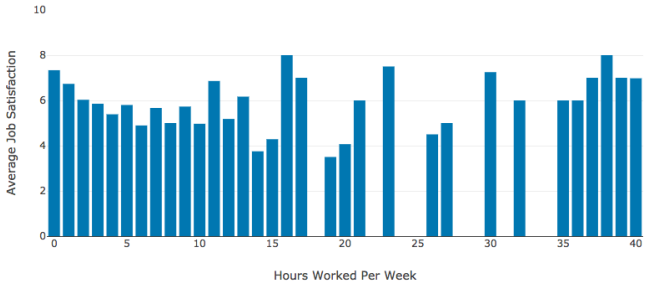

Fig. 4. The graph generated when selecting the Hours Worked Per Week results section. There is a much stronger variance in Job Satisfaction seen here.

3.2 Visualization Application Results

As far as actual rendering our visualization, the idea is there but a more flushed out implementation is required to truly get the final product that we want to see. There are two main sections to our visualization application.

First, when loading up the web page, the user is presented with a collection of colored boxes with characteristics on them. These boxes correspond to the survey results categories that we selected as potentially relevant. Each box displays the response in that category that corresponds to the highest average job satisfaction. Each box is also colored to reflect how high of an average job satisfaction that response correlates to. A ligher colored box might only correspond to a satisfaction level of 6.5, whereas a bright, vibrant box stands out and might correlate to a satisfaction level of over 8.

When a user wants to further explore a response category, they can click on that box and a graph of the compiled results will appear below. This information provides a deeper look into that categories' responses and could later include even more than just a graph. We used a Plot.ly bar graph for this part of the visualization, and it worked great for a simple way to show our data. The graph responds to mouse selection and hovering, and automatically adjusts to the size of the page that the user is viewing it in.

4 Contributions

This midterm project was done with consistent check-ins and communication between Damian and Sean throughout. However, the work was divided based mostly on previous experience and skill.

Damian handled the database work, parsing the Developer Survey data dump into something we could work with. He brought his experience with database management and SQL Server from other jobs to the project. Damian also built up the web application framework to the point where the client application could be served and data could be retrieved from the database.

Sean wrote a good portion of the front-end JavaScript and HTML code to implement the visualization on the web page. Sean also has experience with JavaScript libraries and data parsing/visualization from another job. He also wrote the majority of this midterm report.

5 Conclusion

For being a partial implementation, the visualization we produced has promise. It provides some useful information, and is a strong template for flushing out and viewing a larger number of responses. We learned quite a bit throughout the process of going from idea inception to a functioning visualization application.

5.1 Most Difficult Part

There were multiple hurdles to overcome during the development of this visualization. The hardest problem was definitely parsing the Developer Survey data dump into something we could work with. There were non-normalized responses and inconsistent formatting throughout. On top of that, the file of results was very large, containing over 100 responses for over 50,000 developers. Through the process of parsing that data into a usable format, some small pieces of data were lost. However, this visualization is based on statistical analysis of the entire survey result set, so minor data loss does not change the overall results.

5.2 How to Improve the Project

With every piece of our visualization we developed, more ideas and improvements presented themselves. Here is a short list of ways that we could potentially improve the project, from simple fixes to major functionality upgrades:

- Completing our implementation is the first step in improving our visualization. This application is somewhat valuable as a proof of concept, but it needs many more response categories for it to become truly valuable.
- We need to create a much richer way to distinguish between the relative value and influence of each category. This could be accomplished by making a more obvious, intuitive color map for the response boxes, as well as incorporating changes in the size and/or opacity of the boxes based on other statistical characteristics of the categories.
- The information displayed when a category box is clicked could be increased. Currently we just display a bar graph of the average job satisfaction for each response, but we could link to an entire page with better visuals and a more detailed breakdown of not just averages, but how many responses make up each average and the range for each response.
- Finally, there is more work to be done with the actual database we are drawing from. Some refinement of which categories to expose and how to link them would increase the responsiveness of the visualization as well as allow us to add more interesting information to the details section for each category.

5.3 Other Problems That Can Be Solved

Our visualization provides two values as far as we are concerned. First, it is a great way to get a quick, simple look at survey results for users that do not have the time to dig deep into a bunch of number and graphs. This could be applied to many other sets of survey results, especially in a modern world where most content is consumed in much smaller, shorter chunks than it used to be. Second, our visualization is an interesting way to navigate through the survey results for users that do have time, tying each category back to the simple idea of the "ideal developer". Users can both relate to categories where their own characteristics line up with the survey results, or learn more about areas where they are in the minority compared to the survey results. This personal touch of organizing the visualization for quick comparison to a user's characteristics could be applied to any data set where the data comes from the behavior of humans.

Acknowledgments

The authors wish to thank Jake Jeffreys for providing invaluable input on what he most wants to learn from a visualization of this data set.