Project: Toyplot

Diksha Sharma
V00810784
Assignment 1

Table of content:

Part A: About Toyplot

This section briefly describes what Toyplot is and some of its features.

Toyplot is an open source plotting toolkit for python. The purpose of Toyplot is to develop interactive, animated plots that are unique with capabilities of electronic publishing and support reproducibility. It provides a clean, minimalist interface that mainly focuses on engineers and scientist to work with and love. Toyplot's graphics are completely self-contained and embeddable, without the need for a server. Toyplot has a wide range of color palettes to choose from and sensible default styling that minimizes chart junk and maximizes data ink out of the box. The figures created in Toyplot are highly interactive. Some of the features that are available in Toyplot are different plot types such as bar plots, matrix plots, number-line plots, etc. Toyplot uses standard CSS and HTML markup for styling purposes [1]. Toyplot integrates with Jupyter without the need of any plugins, magics etc. Different interaction types are hyperlinking, exporting figure data to CSV and interactive mouse coordinates. The interactive output formats are embeddable, self-contained HTML. Toyplot also has some dependencies which at minimum Python 2 or 3, custom_inherit, multipledispatch, and numpy.

Part B: Stakeholders

To better understand Toyplot below is the stakeholder table which shows the acquirers, assessors, communicators, developers along with other information. This section is to better understand who the system is for, not what the system does.

Role	Concerns
Acquires	Toyplot/ Sandia National Laboratories
Assessors	https:gitter.im/sandialabs/Toyplot,
Communicators	Timothy M. Shead (tshead@sandia.gov), https:gitter.im/sandialabs/Toyplots
Developers	Sandia National Laboratories
Maintainers	Sandia National Laboratories, 21 watchers on GitHub including Timothy M. Shead
Production Engineers	Python 2 or higher, Jupyter, numpy, six, custom_inherit, multipledispatch.
Suppliers	Jupyter, Python 2 or higher, numpy, six
Support Staff	The workers of Sandia National Laboratories and the watchers present on GitHub.
System Administrators	Sandia National Laboratories
Testers	Sandia National Laboratories and 21 watchers on GitHub
Users	Mainly designed for Scientists and engineers.

Part C: Glossary

This section contains the defining and/or meaning of the frequently used terms in this paper.

Sandia National Laboratories: Multimission laboratory operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration.

Pie Charts: Is a circular statistical graphic which is divided into slices to illustrate numerical proportion.

GitHub: A web-based hosting service for version control using git. Mostly used for computer code.

Jupyter or Jupyter Notebook: It is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

Magics: It is a pre-defined function by IPython that can be called with a command line style syntax.

Custom_inherit: Proper name is custom inheritance. Its classes and custom inspectors.

Multipledispatch: also known as multimethod. It is a feature of some programming language in which a function or method can be dynamically dispatched based on the run-time (dynamic) type or, in the more general case some other attribute, of more than one of its arguments.

Numpy: it is the fundamental package for scientific computing with Python.

Part D: Prior research

This section explains in detail the methods/steps taken in order to understand the project and the process of documenting.

D.1: **Investigation Process:**

As shown in Figure 1

1. Gather all the data for Toyplot, collect every code on the machine, all the datasets. Save all of this information on one's machine.
2. Researched about what is Toyplot, how it works different features available, different graph types that can be made using Toyplot, how the code is written, and different dependencies that the project has.
3. The next step taken was trying the product (Toyplot). In order to see how a user of this open source application would see things the application was installed and used.
4. Understanding the code was the next step. All the files that were used to create the application were available on GitHub. Those files were read to better understand how each feature of the application works and is implemented.

5. The specific information investigated further was the way different graphs and/or chart were implemented in the open source project. This would help in creating/ adding new feature of Pie chart in the system.

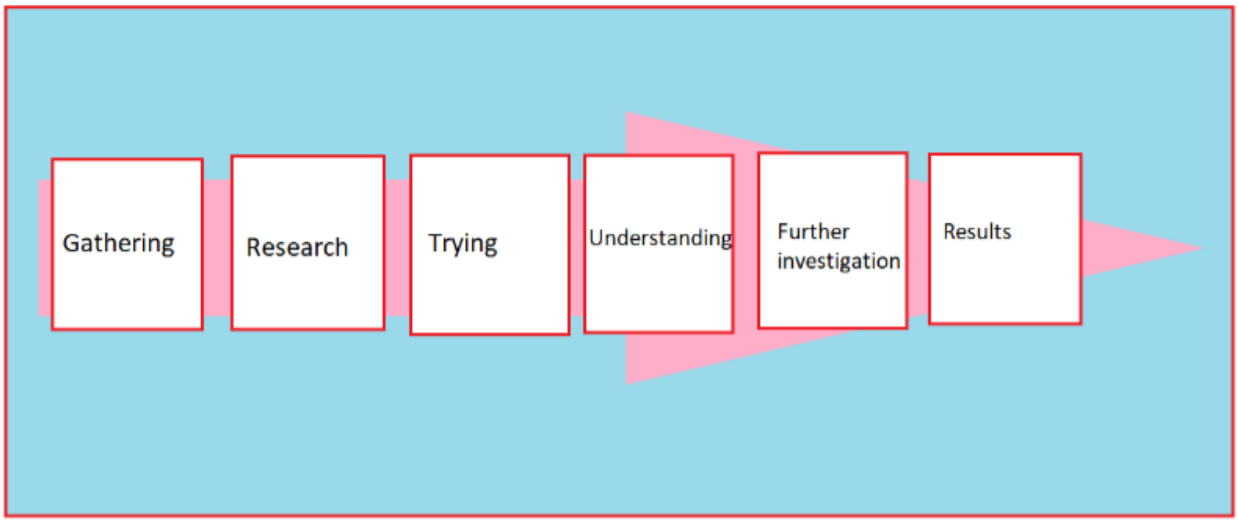

Figure 1: Investigation Process

D.2: Information about the existing application:

1. The way the existing code. Example: the coordinates structure, the text structure.

D.3: Different types:

This section has the different chart types that are already available for the users of Toyplot.

Below are the different chart types that are already available for this application:

- Bar graphs, scatter plot, markers, line graphs/charts, animation charts.

Section E: Use case scenario and Growth scenario

This section defines modifiability and modularity. It also describes the use case scenario and the growth scenario. For each scenario, the purpose to show how well the architecture drivers will support the scenario. This section maps to satisfy business drivers. Each scenario also has supporting reasoning for the choice of response measure.

E.1: Definitions

Modifiability is the ease with which a system can be changed without introducing defects.

Modularity is the degree to which a system or computer program is composed of discrete components such that a system that has a high degree of modularity (i.e. independent system components) would be more easily modified [2]. Modularity is partition of a system into distinct modules representing separate areas of functionality; a classical modifiability technique [3].

E.2: Scenarios

Use case scenario

Aspect	Details
Scenario Name	Adding a different chart type ~ Pie chart
Business Goal	More option of charts for the user/client to choose from.
Quality Attributes	Modularity
Stimulus	User wishes to have Pie charts as one of the option of chart type when working with Toyplot.
Stimulus source	Developer
Response	Added new feature: Pie charts has been included in the open source application. Locates where the existing code has different chart type functions, learn them, modify them without affecting other functionality; test modifications; deploy modification.
Response measure	Rough estimate would be1 person ~ 3 week

Note: The response measure is given with the assumption that the developer working on the application is well experienced with Python, CSS, HTML and other dependency tools.

The response measure would be 1 person for 3 weeks; the breakdown for those 2 weeks is as follows:

1st week – Learning and understanding the existing code, playing around with the different features of the applications. Coming up with different ways of coming up the ideas for the new feature.

2nd week – Researching and implementing the bits and parts of the additional feature to the existing code.

3rd week – Testing the new features on different cases to check and see if it runs and creates the chart according to the user's needs.

Growth scenario

Aspect	Details
Scenario Name	Decrease number of files accessed/ran when creating charts/graphs with Toyplot. Specifically pie charts in this case. For example: instead of accessing 15 different files to gather minor pieces of the application to run, have the features in less number of files.
Business Goal	Create a faster system
Quality Attributes	Modifiability
Stimulus	The program would be going through less number of files to gather content for pie charts
Stimulus source	Developer
Response	Less number of files accessed when the user is making pie charts
Response measure	~ 3 week

Note: The response measure is given with the assumption that the developer working on the application is well experienced with Python, CSS, HTML and other dependency tools.

The response measure is listed as approximately for 3 weeks because of the following reasons:

1. The exiting files would be learnt and understood by the end of week 1.
2. Week 2 would be implementing the new feature to the existing code and checking it see if the features spread over multiple files could be implemented in fewer files.
3. Week 3 would be testing the new features added and the testing if all the other features work.

Section F: Project specific Use case scenario and Growth scenario

This section shows the project specific use case scenario and growth scenario, diagrams and explains the way system works. The section will also answer the questions of if the current design meet the response measure for both use case scenario as well as the growth scenario.

F.1: Use case scenario

A new feature of being able to create Pie charts needs to be added to the current system.

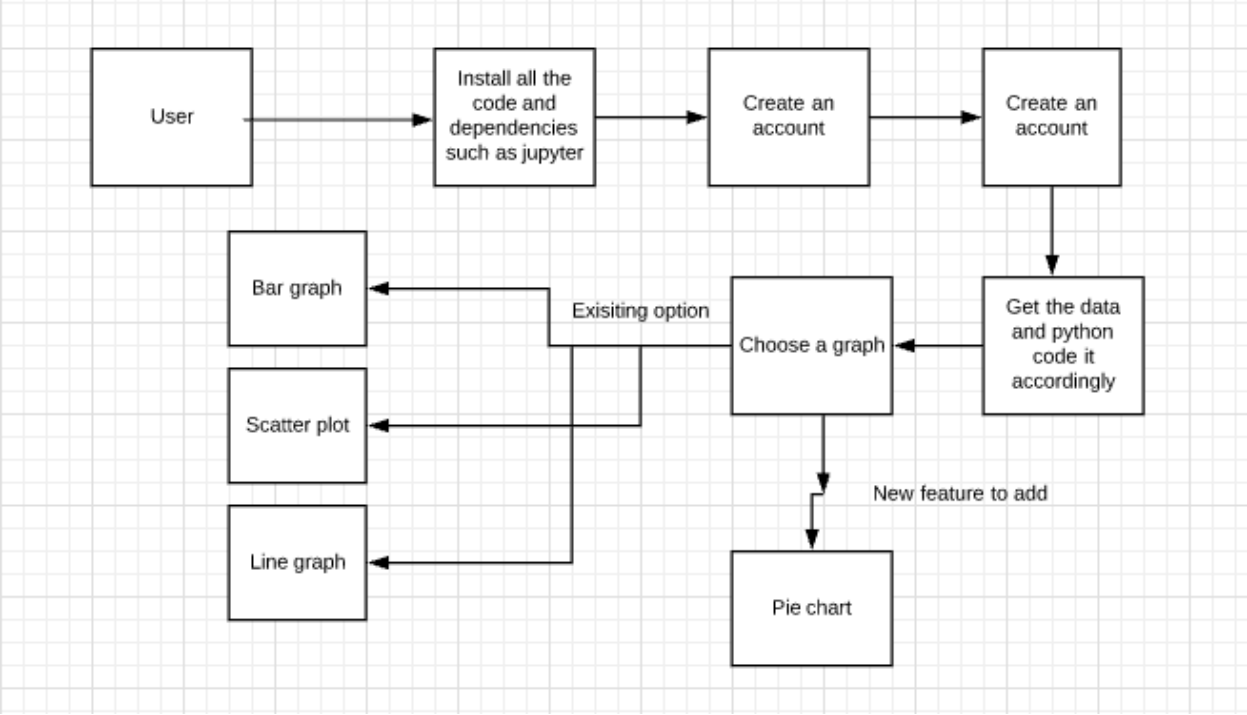

Figure 2: Use case scenario

The current system has many option for a user to graph/chart their data as. For example, there are bar graphs, scatter plot amongst others available for the engineer or the scientist using the application. The open source project is designed in a way that uses multiple features from different files to create a graph/chart that the user desires. There are multiple folders and files on

GitHub that contain information about the graph structure, graph height, graph width along with other features that the user may need. These structures are hardcoded so that the user (scientist or an engineer) would only need to input the data in Jupyter notebook to graph/chart information as desired. These hardcoded files such as format.py and graphs.py are connected to each other in a way that all of the files need to be important for the application to work. The program written is generic enough that once a developer takes their time at understanding how the program is written and the meaning of what certain things are done the way they are.

For the new feature, files under the folder called Features would be used, along with other contents from the GitHub repository. A new python script would be written in order to capture all the necessary content for making pie chart, such as the height and width of the chart, the way text might be displayed on the chart along with other features. The existing application has generic files that contain information about how the width, height of the graph/chart would be displayed on a canvas which makes it easier to add a new feature. One of the most important part of creating a new file would be importing all the right files as seen in figure 3. Since all the files are separated by folders it would be easier for a developer to understand the folder that is needed for them to add new feature.

```
14  import numpy
15  import six
16  import toyplot.coordinates
17  import toyplot.broadcast
18  import toyplot.color
19  import toyplot.config
20  import toyplot.layout
21  import toyplot.style
22  import toyplot.units
23
```

Figure 3: Canvas.py

The open source code is flexible in terms of the way margins and boundaries are defined. Since there are multiple graphs that are created using Toyplot, the developers were thoughtful when writing code for all the python files. They made sure that there is very little to no redundancy and no overlap.

An example of clean written code is the figure 4, it shows that the developers were very detailed with the comments on the entire program.

```
#    if nbconvert_version not in allowed:
#        raise Exception("Unsupported nbconvert version: %s, use one of %s" % (nbconvert_version, ", ".join(allowed)))

    # Some installations of ipython don't properly configure the hooks for Pygments lexers, which leads to missing
    # source code cells when the documentation is built on readthedocs.org.
    import pygments.plugin
    if not list(pygments.plugin.find_plugin_lexers()):
        raise Exception("It appears that ipython isn't configured correctly.  This is a known issue with the stock conda ipython package.

    # Convert the notebook to pure Python, so we can run verify
    # that it runs without any errors.
```

Figure 4: Comments on file build.py

One of the main things that need to be done to make the pie chart option available for user are as follows.

First a python files for pie charts need to be created and imported to canvas in order to define the space available to create charts

```
        if not isinstance(mark, (
                toyplot.mark.BarBoundaries,
                toyplot.mark.BarMagnitudes,
                toyplot.mark.Plot,
                toyplot.mark.Scatterplot,
            )):
            raise ValueError("Cannot set datum style for %s." % type(mark))

```

Figure 5: Pie chart option example

Here we need to state that we are making another type of chart – Pie chart. Therefore, something like "Toyplot.mark.PieChart" would need to be added to the list. The developer could use a lot of the content that is already available. For example: in file text.py at line 290, the need to be compatible with "def compute_position(layout)" is displayed. This is to compute top + bottom + left + right coordinates for line boxes + text boxes, relative to the layout anchor. This code could be directly used with few changes for the Pie chart file.

New feature such as a pie chart can be implemented with not a lot of trouble because most of the styling files are very co-operative although they are hard coded. The classes and subclasses are generic so that even if a new feature is introduced it would be easy for the code to adapt to it. Files such as style.py and color.py are good example of how the code is not specific to one genre of charts.

F.2: Growth scenario

In order to improve the existing system, one of the things that can be done is to decrease the amount of files that are being accessed when creating a graph/chart. An option to do this is to cluster the requirements, features, measurements in folders and overlap some of the information. The exiting code is already in folders divided as test files, docs files, python files as seen in figure 6.

artwork	First iteration of working image visualization tests.
conda.recipe	Replace toyplot.color.near_black with toyplot.color.black, and elimin...
docs	Use Python 3 to build documentation notebooks.
features	Move temperature data into the toyplot.data module.
sandbox	Further simplify the animation API.
tests	Further simplify the animation API.
toyplot	Allow table cell data to be rotated.

Figure 6: Cluster of different files

The way the existing program runs is as shown in figure 7: growth scenario.

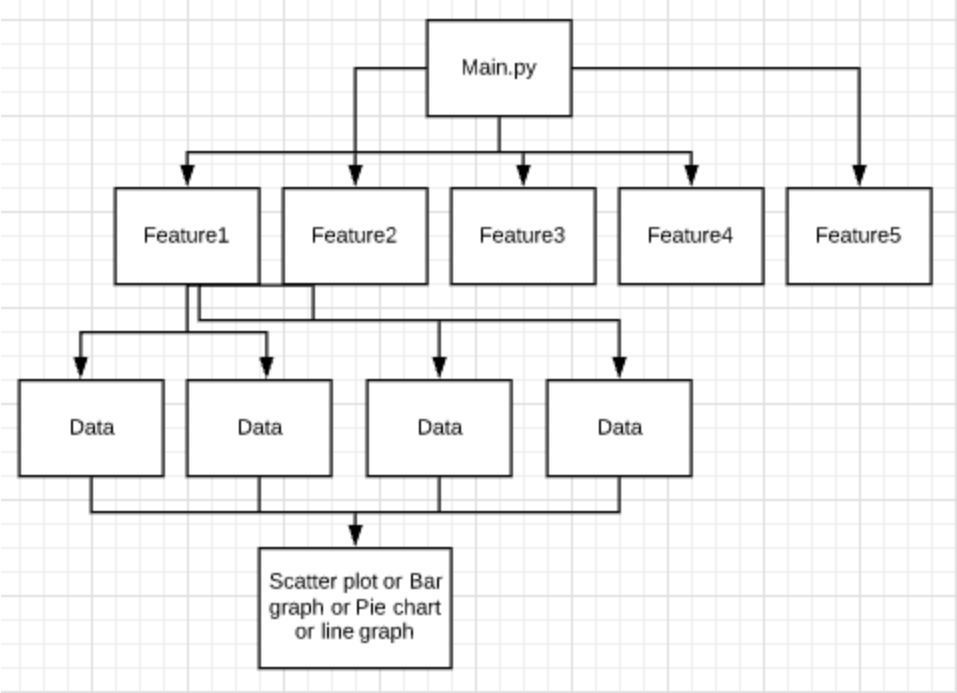

Figure 7: Growth Scenario

The main file is run where multiple features from different files are imported. Then, data is entered by the user and a graph/chart is selected to print out on the canvas. This entire process accesses more than 10 files to print one chart which is a lot of work for an application. One way to grow the application in positive manner is to decrease the number of files that are accessed when printing out a graph on canvas. The next question that comes in the picture is of whether it is current system meets the response measure, to which the answer would be yes. The response measure time for this part is set to be approximately 3 week. The breakdown of which is given in the section above. Since, the existing application clearly labels each feature and most of the times the reasoning is either obvious or given. The clustering of all the features into less amount of files should not be a problem.

References:

[1] Sandia Corporation. Internet: https//toyplot.readthedocs.io, 2014. Accessed: 8 February 2018

[2] K. Adams. (2015). Non-functional requirements in system analysis and design. Available: https://books.google.ca/books?id=U7-pCAAAQBAJ&pg=PA176&lpg=PA176&dq=difference+between+modularity+and+modifiability&source=bl&ots=TEeFUoC4nP&sig=K-jCLzvaMr0QJPYRwOMEeTOEwoE&hl=en&sa=X&ved=0ahUKEwjfu9WxvJjZAhVF62MKHXRPALUQ6AEIODAB#v=onepage&q=difference%20between%20modularity%20and%20modifiability&f=false. Accessed: 8 February 2018.

[3] Internet: http://www.ieee.org.ar/downloads/Barbacci-05-notas1.pdf. 22 January 205. Accessed: 8 February 2018

[4] Toyplot. Internet: https://github.com/sandialabs/toyplot/. 5 February 2018. Accessed: 8 February 2018.