TWENTY 6IXERS

DELIVERABLE 1

Stefan Mitic
Kevin Bato
Chia Anamekwe
Zongda Wang
Byron Leung

TABLE OF CONTENTS

MATPLOTLIB ARCHITECTURE

Architecture Overview

Matplotlib uses object-oriented design patterns that support reusable code and association between objects. Figure is the object that has all the elements for each graphic in Matplotlib. There are three main layers that make up the stack in order to implement Figures, these layers include the Backend layer, the Artist layer, and the Scripting layer. The frontend, known as the Artist layer, uses modules that are tightly coupled amongst each other specifically using subclass coupling within object-oriented programming. The use of coupling results in a ripple effect when a module is changed and coupling requires more effort to reuse and test particular modules because dependent modules must be included.

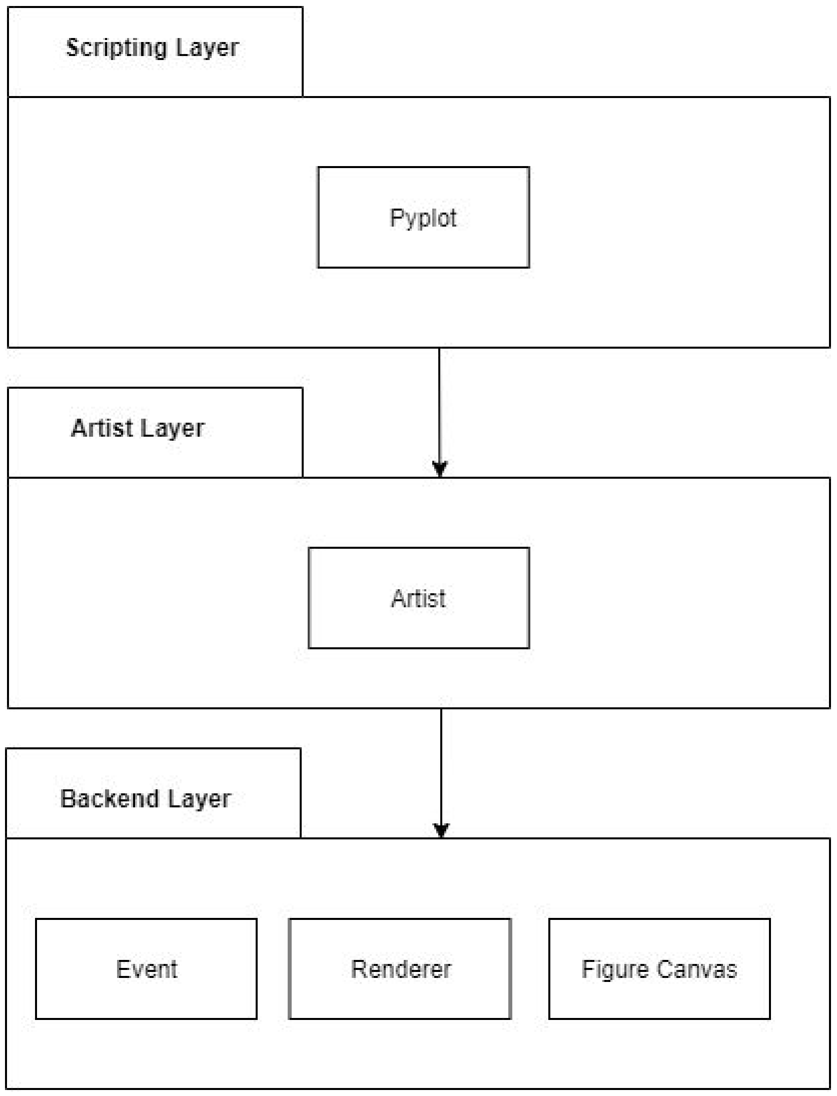


Figure: Overall Architecture

Backend Layer

The backend layer has the implementation of the following abstract interface classes: FigureCanvas, Renderer, and Event. The abstract base classes for the listed objects can be found under backend_bases. The concrete implementation for the derived classes is found in their dedicated modules. The FigureCanvas class takes in a figure object and creates the canvas the figure would render into. The Renderer class handles the drawing and rendering operations by using a drawing interface in order to draw ink onto the canvas. The Event class is the base class for all of the Matplotlib event handling. KeyEvent and MouseEvent are derived from Event and store the metadata of keys pressed and the locations in pixels. Events are primarily handled in callbacks allowing the user the ability to interact with their figure and data.

Artist Layer

The artist layer is the core layer of matplotlib as it provides the objects needed to be rendered from the backend. The Artist class is an abstract class that has shared attributes and methods that every artist instances share which includes: a direct translation of an artist coordinate system to the canvas coordinate system, visibility, clip box (drawing region), labels, and an interface to handle all user interactions on the artist instance (mouse hover, mouse clicks, etc...).

There are two types of Artist: Primitive artist that represent objects seen on the plot such as Line2D and Patch (Rectangle, Circle, Arrow, etc...) and Composite artist that contains Axis, Tick, Axes and Figure. The Figure class is the most notable artist instance in matplotlib, it is a top level container containing both primitive and composite artist instances. The Axes class is mainly responsible for setting the graphical components of the plot background such as tick marks, axes lines, grid and color. Also, it has methods that create primitive artists to add to the Axes instance.

The coupling between the Artist and Backend layer primarily happens from the draw method of abstract Artist class. The draw method in the Artist class will be overridden by the Artist subclasses as the subclasses themselves will call the appropriate method to render the specific object on canvas or an output file.

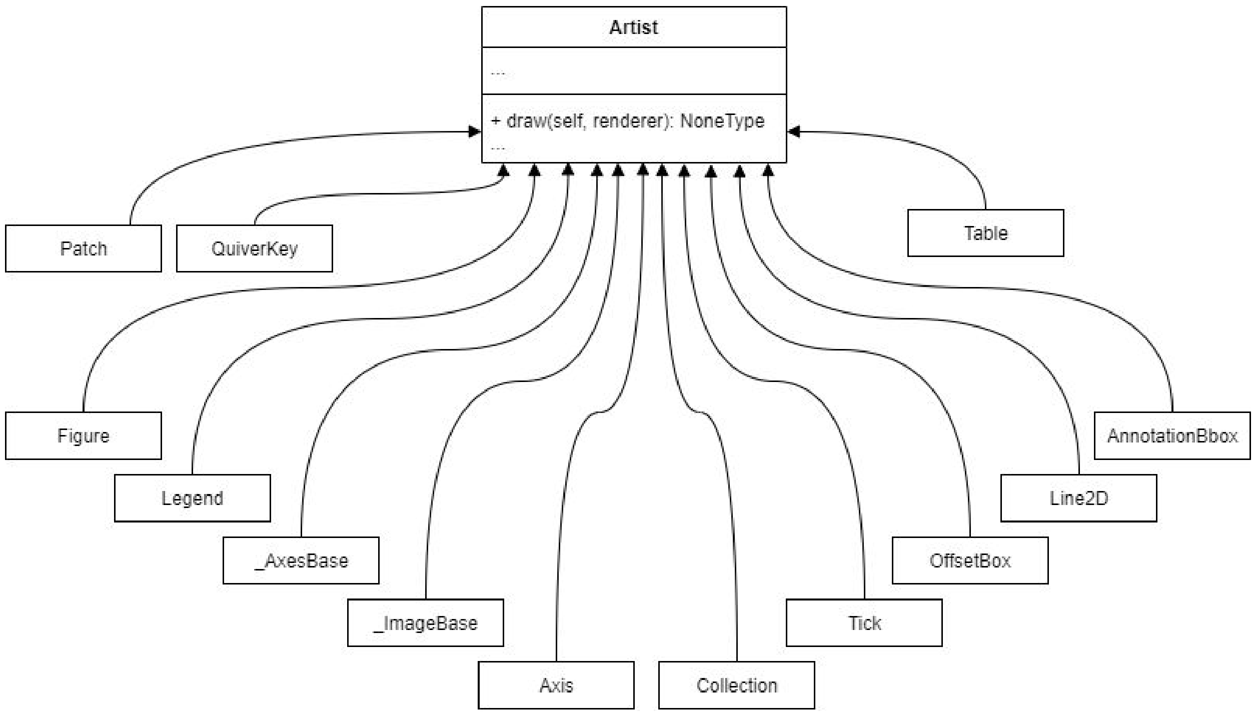


Figure: Artist Layer

Scripting Layer

The Scripting Layer functions as an API having a collection of command style functions that make Matplotlib work like MATLAB. The state-based interface is known as pyplot and it simplifies plotting using procedural programming. Pyplot creates and interacts with objects from the artist layer in order to plot features.

SOFTWARE DEVELOPMENT PROCESSES

Considerations

When deciding which process to adapt for this project, our main considerations were the flexibility of requirements for the project, the project type, team experience and the users.

Flexibility of Requirements

The requirements for the project are clearly defined, understandable, and can be expected not to change. Good development processes for this would be well documented, more rigid models such as waterfall and V-model.

Project Type

Whilst this project involves a complex system, we will only be working on small parts of it, which can be divided into modular tasks, and use a small development team. Based on this, suitable development processes would be agile methodologies.

Team Experience

As the team is made up of less experienced developers working in an unfamiliar environment, it could be argued that agile methodologies are suitable as they allow more flexibility for the project and more room for error, but waterfall, v-model and similar processes are more suited to inexperienced teams.

Users

There will be no user involvement or participation in this project, and no continuous feedback to handle, and so some suitable development processes which don't require user participation would be waterfall, v-shaped, spiral.

Summary

Overall, development processes that may be suitable for this project are: waterfall, v-shaped and agile methodologies. Below are brief summaries of the pros and cons of each of the methods.

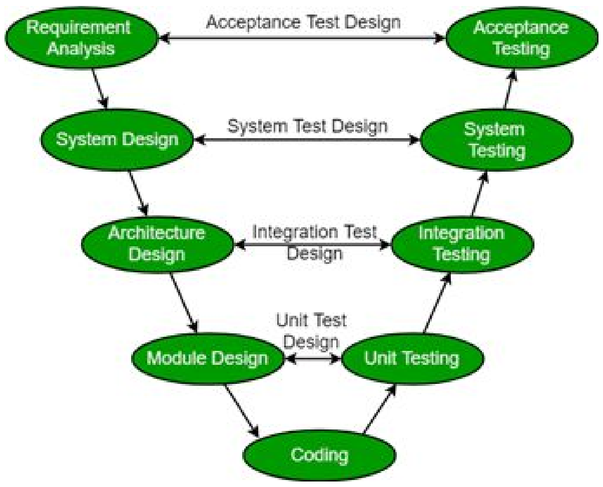
Waterfall – requirements are clearly defined and we know what needs to be done and when, and this is unlikely to change. We also do not have access to user feedback, and waterfall works well in these cases. However, we should consider that this is our first time working with a large codebase, and if errors are made in earlier stages, there is no room to correct this. We could consider adapting this method slightly to allow for reflection and revisions.

V-shaped – again, requirements are clear and unlikely to change, and now there are provisions to deal with errors made earlier due to the reverse manner of the validation stage. It also doesn't require much input from the user. However, there is still a lack of flexibility, and a strong need for a clear plan.

Agile – as we are working with a small, close-knit team on a smaller scale project, an agile methodology would be well suited to the environment. It also allows more flexibility for things going wrong, which is more likely with an inexperienced team. However, due to the large amounts of responsibility on each person and needs for time estimation, agile tends to be more suited for an experienced team.

Chosen Process

For our next deliverable, we have chosen to follow the V-model development process. It can be considered an extension of the waterfall model; stages are executed in a sequential manner, with a corresponding testing phase for each stage.



The two sides of the V can be split into two categories – verification and validation. Verification (left-hand side) involves evaluating the development phase to find whether it meets the specified requirements (i.e. to confirm the product is being built in the right way). Validation (right-hand side) involves evaluating the software upon completion of the development phase (i.e. confirming the right product was built). In this process, the corresponding validation stage is planned in parallel with the verification stage.

Below is a brief description of each stage of the V-model, and any modifications required for our project:

- Requirements analysis – also known as requirements gathering, aim to collect and understand the requirements and expectations of the project. The requirements are already specified for us, and so for our project, this stage will mainly focus on ensuring our understanding of them.
    - Acceptance testing – involves testing the product in a user environment to ensure product built to user satisfaction
- System design – the system as a whole is designed, detailing the hardware and communication setup. As our system is already defined, and we are only working on a very small part of it, this stage will mainly focus on understanding how this small part fits into the rest of the system.
    - System testing – checks the functionality of the entire system and it’s communication with external systems

- Architectural design – breaking the system design down further into modules and the interactions between them. Again, this is already in place and has been analysed as part of this deliverable.
    - Integration testing – involves testing the interactions between modules working correctly.
- Module design – The specific design of the modules defined above are specified. As we are working on bug fixes, we will likely be working within a specific module, which, again is already designed, so our focus here is understanding the design and the specific issue.
    - Unit testing – testing at code level to ensure module functions according to design.
- Coding phase – the coding of the system. Here is where we will implement the code to fix the bug.
    - Our validation phases will mainly ensure that nothing was broken during our changes, and everything still works as it should, along with the performance fix from correcting the bug.

Our main reasons for this decision were its suitability for a short project with clear, un-changing requirements, but also the greater flexibility allowed for managing errors with the various testing stages (as opposed to waterfall). It is also easy to understand and implement, and easy to manage, which is needed in a team of non-senior developers. Furthermore, little input is required from the client, which was an important feature as we do not have access to the client for this project.

References:

https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm