1. Please choose 2 projects from the following open-source software projects in the list below and write a (very) brief report for each project. Your report must explain what is the purpose of the project, its architectural patterns/styles, as well as 3 quality attribute scenarios. The quality attributes must be aligned with the quality attribute advocated/promoted by project developers/maintainers. Please include in your report references to the sources where the relevant information can be found.

* Audacity [<https://www.audacityteam.org/>]
* gpsd [<https://gpsd.gitlab.io/gpsd/>]
* matplotlib [<https://matplotlib.org/>]
* Selenium WebDriver [<https://www.selenium.dev/>]
* Zotonic [<https://zotonic.com/>]

1. Please choose 2 projects from the following open-source software projects in the list below and write a (very) brief report for each project. Your report must explain what is the purpose of the project, its architectural patterns/styles, as well as 3 quality attribute scenarios. The quality attributes must be aligned with the quality attribute advocated/promoted by project developers/maintainers. Please include in your report references to the sources where the relevant information can be found.

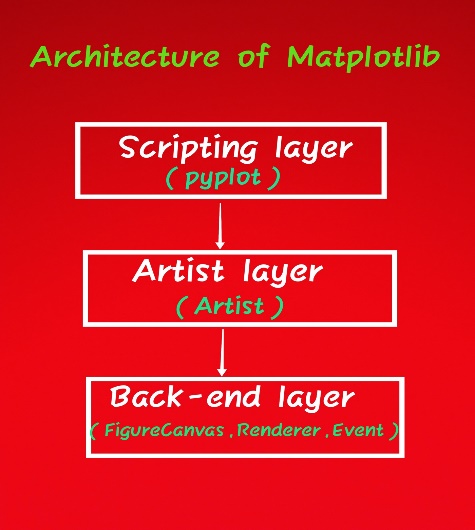
* Jitsi [<https://jitsi.org/>]
* Joomla [<https://www.joomla.org/>]
* Kill Bill [<https://killbill.io/>]
* ONOS [<https://opennetworking.org/onos/>]
* Yesod [<https://www.yesodweb.com/>]



Architectural patterns/styles

Matplotlib มีรูปแบบ สถาปัตยกรรมที่ใช้จะแบ่ง Layer เป็น 3 Layer หลักๆคือ

**Backend Layer, Artist Layer,** **Scripting Layer**



**Backend Layer**— เป็น Layer ที่จัดการงานต่างๆทั้งหมดผ่านการสื่อสารกับชุดเครื่องมือเช่น [wxPython](https://www.wxpython.org/" \t "_blank) หรือภาษาที่ใช้วาดภาพเช่น [PostScript](https://www.comgeeks.net/postscript/index.htm) กล่าวคือ เป็น Layer ที่เกี่ยวกับการวาดรูป โดยจะมีคลาสหลักๆ 3 คลาสคือ

* **FigureCanvas —**ใช้ในการร่างโครงที่จะวาด ตัวอย่าง into.[matplotlib.backend\_bases.FigureCanvasBase](https://matplotlib.org/api/backend_bases_api.html#matplotlib.backend_bases.FigureCanvasBase)
* **Renderer —**ใช้วาดสิ่งที่เราจะวาดขึ้นมา

ตัวอย่าง FigureCanvas.[matplotlib.backend\_bases.RendererBase](https://matplotlib.org/api/backend_bases_api.html#matplotlib.backend_bases.RendererBase)

* **Event —**ใช้ในการรับ input จากผู้ใช้

ตัวอย่าง clicks.[matplotlib.backend\_bases.Event](https://matplotlib.org/api/backend_bases_api.html#matplotlib.backend_bases.RendererBase)

**Artist Layer** — เป็น Layer ที่ผู้ใช้จะสามารถควบคุม-ปรับแต่งภาพว่า จะให้ออกมาในรูปแบบไหน มีลักษณะแบบใดบ้าง ออกมาเป็น Object เช่น ชื่อ, ป้ายกำกับ,เส้น,จุด ,ฯลฯ

โดยจะแบ่งชนิดของ Object เป็น 2 แบบ หลักๆ คือ

* **Primitive:**

Line2D, Rectangle, Circle, text.

* **Composite:**

Axis,Axes,Tick, and figure.

**Scripting Layer** — เป็น Layer ที่มีความซับซ้อนน้อยที่สุดโดยจะเป็นเพียงแค่การเขียน สคริปต์ ซึ่งเป็นคําสั่งสําหรับการสร้างกราฟิกและพล็อตที่รวดเร็วและง่ายดาย

ตัวอย่าง matplotlib.pyplot

**Quality Attribute Scenarios (QAS)**

**Interoperability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | User |
| Stimulus | การใช้งานร่วมกับภาษา Python |
| Artifact | ระบบ |
| Environment | Integration |
| Response | สามารถใช้งานร่วมกันได้ |
| Response Measure | ผลลัพธ์ในการใช้งานร่วมกัน (สำเร็จ,ล้มเหลว) |

**Scalability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | ข้อมูลที่ป้อนเข้ามา |
| Stimulus | ปริมาณข้อมูลที่มีมาก |
| Artifact | ทรัพยากร Ram |
| Environment | Overload Mode |
| Response | ทำให้เกิดการใช้งานของทรัพยากร RAM น้อยที่สุด และ ไม่ใช้เกินความจำเป็นในการรันกราฟต่างๆ |
| Response Measure | วัดระดับการใช้งานพื้นที่ของทรัพยากร RAM, ไม่กระทบกับระบบอื่นๆ |

**Performance**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | User,ข้อมูล |
| Stimulus | ปริมาณข้อมูลที่มีมาก,คำสั่งใช้งานจากผู้ใช้ |
| Artifact | ปริมาณข้อมูลที่จะทำเป็นกราฟ, ทรัพยากร Ramที่รองรับข้อมูลกราฟ,การประมวลผลของระบบ,เวลาที่ใช้ |
| Environment | Normal Mode, Overload Mode |
| Response | ระบบสามารถประมวลผลได้อย่างมีประสิทธิภาพ |
| Response Measure | วัดการทำงาน Latency, ปริมาณ/จำนวนงานที่ผ่านเข้าระบบได้, ความเหลื่อมของเวลา |

Ref:

[Data Visualization with Python — Matplotlib Architecture | by Vin Busquet | DataDrivenInvestor](https://medium.datadriveninvestor.com/data-visualization-with-python-matplotlib-architecture-6b05af533569)

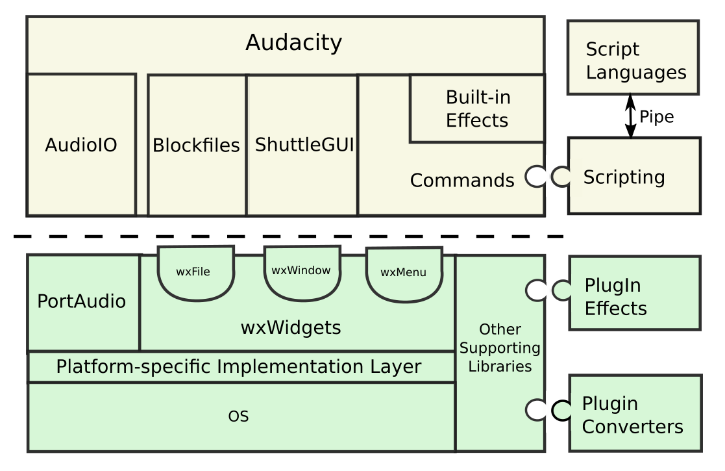
[Architecture of matplotlib. Matplotlib is python library used for… | by Keerti Prajapati | Medium](https://medium.com/@codingpilot25/architecture-of-matplotlib-1a2d44370f5a)

[plt.xxx(), or ax.xxx(), That Is The Question In Matplotlib | by Jun | Towards Data Science](https://towardsdatascience.com/plt-xxx-or-ax-xxx-that-is-the-question-in-matplotlib-8580acf42f44#:~:text=As%20shown%20in%20Figure%201,%2C%20and%20Backend%20layer%20(%20matplotlib.)

[Matplotlib - Wikipedia](https://en.wikipedia.org/wiki/Matplotlib)



Architectural patterns/styles



Audacity มีรูปแบบ สถาปัตยกรรมที่ใช้แบ่งเป็น Layer โดยจะมีหลักๆ ก็คือ

1. **wxWidgets GUI Library**

เป็นส่วนที่ใช้ในการติดต่อกับผู้ใช้งาน

1. **ShuttleGui Layer**

เป็นส่วนที่เอาไว้เป็นตัวกลางการติดต่อระหว่าง Library wxWidgets กับ ตัวของ Audacity และยังมีส่วนช่วยในการลดบรรทัดของกล่องโต้ตอบ

1. **PortAudio Library: Recording and Playback**

เป็น Library เสียงที่ให้ Audacity สามารถเล่นและบันทึกเสียงแบบข้ามแพลตฟอร์ม (Mac, Linux และ Windows)

1. **BlockFiles**

เป็นส่วนที่มีการแบ่งไฟล์เสียงที่มากและยาวออกเป็น BlockFiles จํานวนมาก ซึ่งแต่ละไฟล์อาจมีขนาดประมาณ 1 MB ซึ่งไฟล์จะเป็นนามสกุล .XML เพื่อที่จะทำให้เกิดการใช้ทรัพยากรของ Ram น้อยที่สุด เมื่อมีการเพิ่ม ,แก้ไข หรือ ตัดออก

1. **Scripting**

เป็นส่วนที่มีไว้รองรับการเขียน Scripting ของตัว Audacity ที่รองรับได้หลายภาษา

**Quality Attribute Scenarios (QAS)**

**Usability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | End user |
| Stimulus | เรียนรู้การใช้งาน |
| Artifact | หน้าข้อมูลที่จะทำ, GUI |
| Environment | Runtime |
| Response | สามารถเรียนรู้ได้รวดเร็ว |
| Response Measure | วัดระดับความพึงพอใจของผู้ใช้, ความเร็วในการเรียนรู้การใช้งานของผู้ใช้ |

**Scalability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | ข้อมูล |
| Stimulus | ปริมาณข้อมูลที่มีมาก |
| Artifact | ทรัพยากร Ram |
| Environment | Runtime, Overload Mode |
| Response | ทำให้เกิดการใช้งานของทรัพยากร RAM น้อยที่สุด และ ไม่ใช้เกินความจำเป็น |
| Response Measure | วัดระดับการใช้งานพื้นที่ของทรัพยากร RAM |

**Performance**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | User ,ข้อมูล |
| Stimulus | ปริมาณข้อมูลที่มีมาก |
| Artifact | ปริมาณข้อมูลที่จะทำ, ทรัพยากร Ram |
| Environment | Runtime, Overload Mode |
| Response | ทำให้ผู้ใช้ใช้ได้งานได้อย่างมีประสิทธิภาพ |
| Response Measure | วัดการทำงาน Latency, deadline, ปริมาณ/จำนวนงานที่ผ่านเข้าระบบได้, ความเหลื่อมของเวลา |

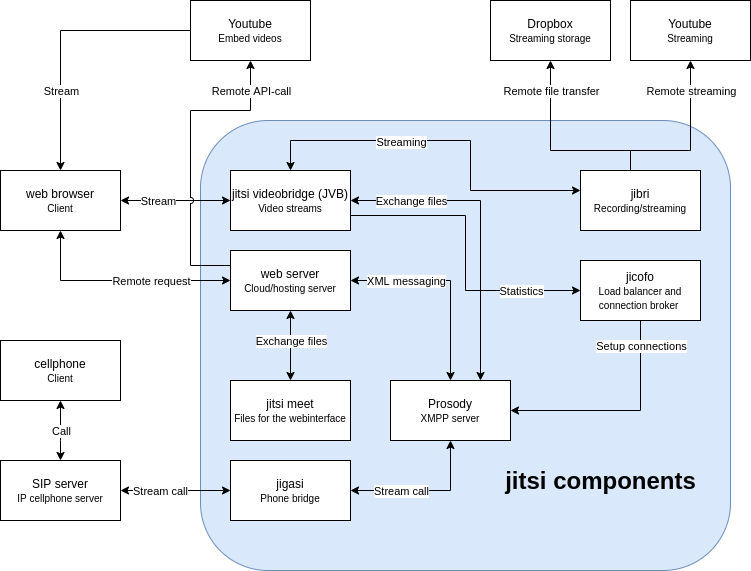
Ref:

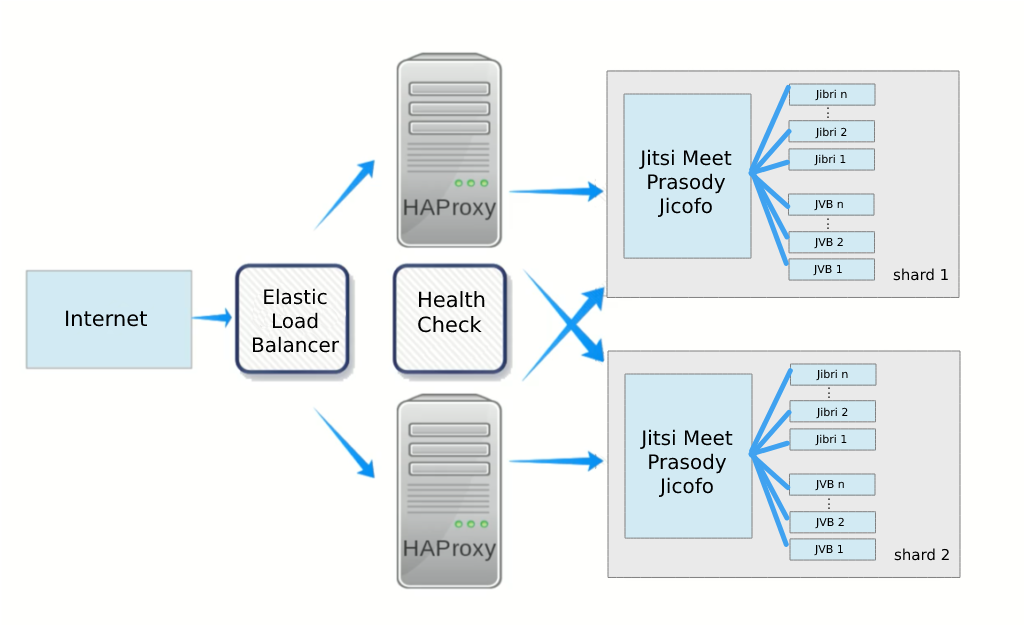
[The Architecture of Open Source Applications: Audacity (aosabook.org)](https://www.aosabook.org/en/audacity.html#fig.aud.2)

[ArchitecturalDesign - Audacity Wiki (audacityteam.org)](https://wiki.audacityteam.org/wiki/ArchitecturalDesign)



Jitsi

Architectural patterns/styles 



Jitsi มีรูปแบบสถาปัตยกรรมที่ใช้คือ แบบ Client-Server โดยจะมี

**Jitsi Meet**

* + เป็นหน้าเว็บไซต์ ทำงานกับ WebRTC ผ่าน JavaScript โดยเรียกใช้ Jitsi Videobridge เป็นโมดูลที่ทำเรื่องคุณภาพของภาพ, การสเกล video conferences. โดยตัวเว็บไซต์ใช้ React และ React Native ในการทำ

**JVB (Jitsi Videobridge)**

* + เป็นตัวรับสตรีมวีดิโอจากผู้สนทนาทุกคน จาก WebRTC มาตัว server

**Jicofo (Jitsi Conference Focus)**

* + เป็นโมดูลของ server-side ที่รับการทำงานจาก Jitsi Meet เพื่อจัดการเรื่อง sessions ต่างๆ ทำหน้าที่เหมือน load balancer ระหว่างผู้สนทนากับ Jitsi Videobridge

**jigasi (Jitsi Gateway to SIP)**

* + โมดูลฝั่ง server-side ที่ทำงานเกี่ยวกับ SIP เพื่อใช้บน Jitsi Meet

**jibri (Jitsi Broadcasting Infrastructure)**

* + เป็นเครื่องมือสำหรับการทำ Video Recording หรือ Streaming Video ที่รับมาจาก Jitsi Meet ในรูปแบบ virtual framebuffer โดยโมดูลนี้จะคอย Capture ภาพและเสียง จากนั้นไป Encode ด้วย ffmpeg ให้อีกที

**Quality Attribute Scenarios (QAS)**

**Interoperability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | User |
| Stimulus | การใช้งานร่วมกับภาษา Java, Video Streaming ออกไปที่ Youtube,การนำมาทำเป็น Self-Hosting ที่ Desktop Browsers กับ Mobile Browsers (windows, iOS,Linux ,Android) |
| Artifact | ระบบ |
| Environment | Integration |
| Response | สามารถใช้งานร่วมกันได้ |
| Response Measure | ผลลัพธ์ในการใช้งานร่วมกัน (สำเร็จ,ล้มเหลว) |

**Availability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | เกิดจากภายใน, ภายนอกระบบ |
| Stimulus | เกิดจากหลุด, ข้ามการทำงาน, พัง, ค้าง, ไม่ตอบสนอง |
| Artifact | การประมวลผลของระบบ, ระบบเครือข่าย, อุปกรณ์เก็บข้อมูล |
| Environment | Normal Mode, Overload Mode , Degrade mode, Fall back |
| Response | บันทึกข้อมูล , แจ้งเหตุการณ์ไปยังผู้เกี่ยวข้อง, ทำการปิด/ยุติการทำงานที่ผิด ตามที่ได้มีการกำหนดไว้, หยุดการทำงานของระบบ ตามเวลาที่กำหนดโดยขึ้นอยู่กับความรุนแรงของปัญหาที่เกิด |
| Response Measure | วัดเวลาที่ระบบจะกลับมาใช้งานได้ปกติ หรือ เวลาที่ใช้ในการซ่อมแซม |

**Modifiability**

|  |  |
| --- | --- |
| **Portion of Scenario** | **Possible Values** |
| Source | End user, develop, system admin |
| Stimulus | ต้องการเพิ่ม / ลบ / แก้ไข / เปลี่ยนฟังก์ชั่น-ฟีเจอร์ |
| Artifact | GUI, Desktop Browsers กับ Mobile Browsers (windows, iOS,Linux ,Android) |
| Environment | Run time, Compile time, Design time, Build time |
| Response | กำหนดจุดที่เปลี่ยนใน Architecture , เปลี่ยนโดยไม่มีผลกระทบกับฟังก์ชั่นและระบบอื่นๆ, ตรวจสอบการเปลี่ยน, การนำไปใช้งาน |
| Response Measure | ค่าใช้จ่าย(Cost), เวลา |

Ref:

[Architecture | Jitsi Meet](https://jitsi.github.io/handbook/docs/architecture/)

[Understanding the Architecture and Components of Jitsi Meet - Meetrix.IO](https://meetrix.io/blog/general/understanding-the-architecture-and-components-of-jitsi-meet.html)