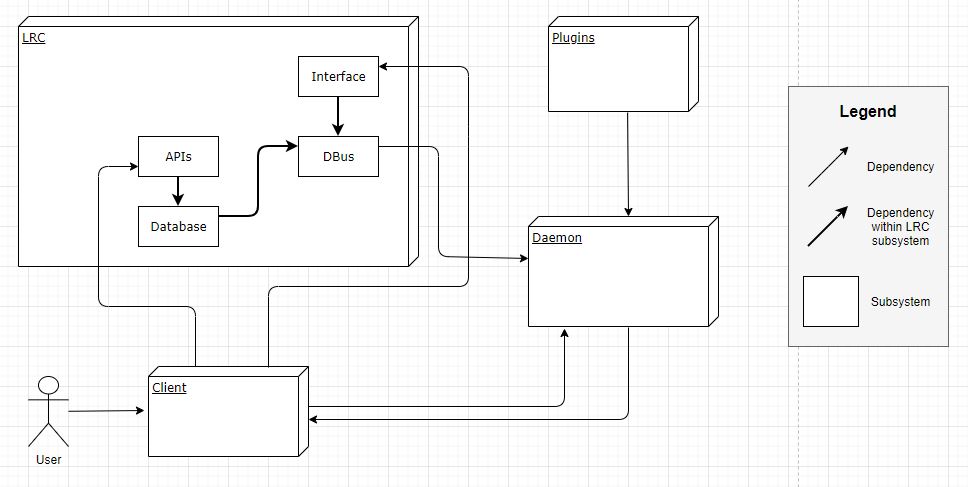
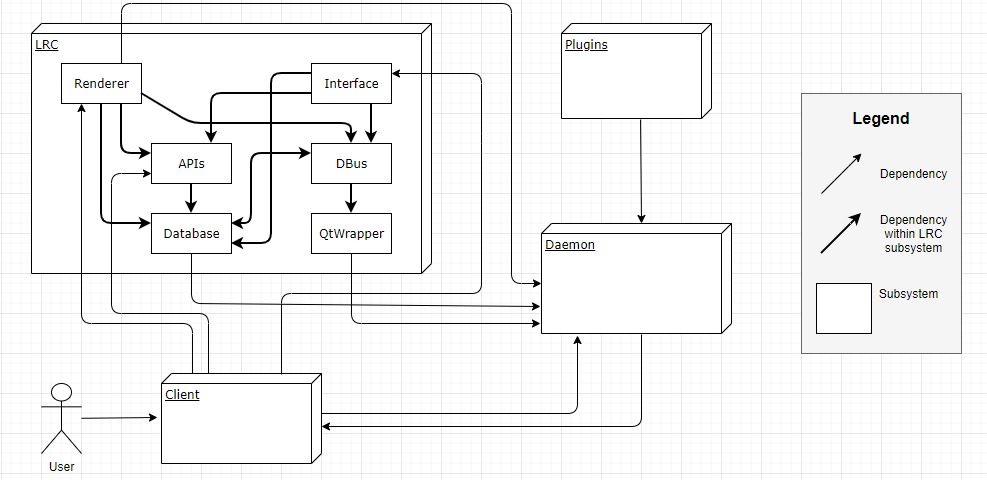
**In-Depth Architecture**

**Conceptual Architecture LRC Subsystem**

****

**Concrete Architecture LRC Subsystem**

****

**LRC Internal Reflexion Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Justified/Unjustified** | **Dependency From** | **Dependency To** | **Rationale** |
| Justified | Interface | APIs | The interface has direct dependency to some API to allow for interaction between the clients and the daemon. |
| Justified | Interface | Database | The interface needs to interact with the database to utilize tools such as, the CallManager. |
| Justified | DBus | Database | The database is able to retrieve information from the DBus in relation to some request. |
| Justified | Database | DBus | The DBus fetches information for the database regarding a specific request, when required. |
| Justified | DBus | QtWrapper | The QtWrapper extends from DBus and is responsible for packaging request data for the daemon. |
| Justified | Renderer | Database | The renderer has direct dependency to the database to allow for easy use of data/tools. (e.g., VideoManager) |
| Justified | Renderer | APIs | The renderer utilizes some API relevant to the request at hand. |
| Justified | Renderer | DBus | The renderer makes call to DBus in order to interact with the daemon and handle video rendering requests. |

The conceptual architecture for the browser system is a pipe and filter style, as mentioned previously. The concrete architecture implements a repository style architecture, while still maintaining the properties of a pipe-and-filter style. Based on the internal reflexion analysis above, all dependencies are justified. The renderer utilizes the database and appropriate APIs in order to gather tools relevant to the request at hand. The renderer communicates with the DBus to handle the requests such that, the DBus can utilize the QtWrapper to allow the daemon to easily understand and display the video feed. The interface depends on the database and APIs to allow for efficient interaction between the clients and the daemon. Lastly, the DBus and database are co-dependant such that, they are able to gather and relay data, respectively.

**Lessons Learned**

There were a number of challenges that we encountered as a group throughout the derivation of this report. One of the most significant challenges was learning how to use *Understand*. Fully grasping how to use *Understand* proved to be a challenge. However, once we got the hang of it, this tool added great value and helped all group members to better understand each individual subsystem as well as the concrete architecture as a whole by giving a visual representation with quantified data for each listed dependency. Additionally, we went through numerous iterations in revising the overall concrete architecture as well as the relevant subsystems. This task proved to be tedious but we managed to derive a concrete architecture and analysis that we feel well represents Jami. From these two challenges, we learned that software systems can have a large number of dependencies that were unexpected (in the conceptual architecture, as well as in each iteration of the concrete architectures).

**Conclusion**

So, the first lesson learned was to be patient when working large software systems as it can often cause unexpected complications and setbacks. Additionally, we learned that working with large software systems have a significant amount of unexpected dependencies and are much more complicated than they appear from a conceptual architecture standpoint. We often found ourselves getting lost within Understand after diving down a specific component. Lastly, we learned how important communication and organization were when developing our concrete architecture. Being in a remote environment definitely increased the severity of this challenge as we were often spontaneously messaging one-another through Discord about how to alter our concrete architecture. We found that setting stricter deadlines and having more frequent team sprint calls helped us organize our process in finalizing our concrete architecture.