Derivation Process Slide

In developing our concrete architecture, we first reflected on our initial conceptual architecture. We began adjusting the conceptual architecture as a team based on our acquired, cumulative knowledge about Jami and its subsystems. We then utilized the Understand software to generate the dependencies between each subsystem within the concrete architecture. This diagram represents the concrete architecture of Jami. It displays all of Jami’s subsystems and the dependencies between these subsystems as numeric values. By applying source code of Jami to the Understand software, we were able to get this diagram. Along with this dependency graph, we used reflexion analysis in order to build our final concrete architecture.

Conceptual Architecture Slide

The figure on this slide represents our updated conceptual architecture. At the highest-level, Jami utilizes an MVC architecture, that is a Model-View-Controller architecture. In this regard, the Daemon is the model, the DBus is a subsystem of the Daemon and is the controller, and the Client is the view. LRC is a library which functions as an intermediate between the Client and the Daemon. The derived conceptual architecture of Jami indicates that Jami is a distributed peer-to-peer and SIP based communication platform.

Alternative Architecture Slide

An alternative architecture could have been of a client-server style. The key difference between client-server and peer-to-peer systems is that in client-server architectures, there are designated clients that request for services and servers that provide those services. However, in peer-to-peer systems, peers act as both service providers and service consumers. Additionally, client-server systems require central file servers. There are numerous advantages and disadvantages to both architectural styles, however, the primary reason that the client-server style was rejected is due to its lack of privacy. The decision to make Jami a fully distributed peer-to-peer platform was ethically motivated by the inclination to protect user privacy. Therefore, it would make more sense that the conceptual architecture of Jami utilized a peer-to-peer style rather than that of a client-server.