## Elaboration Phase Status Assessment

### Results of Elaboration Phase Objectives

#### Detailed Diagramming of all desired features of library

We completed these detailed diagrams (Visual Paradigm file available here). And as expected, these diagrams greatly improved our ability to implement the library without the continued iterations of regular refactoring during implementing that occurs from diving in too early. This was a time saving measure that certainly paid for itself and cemented foundationally what we were trying to achieve before we touched an IDE.

#### Implementing Networking Library with Security/Encryption/Authentication

We successfully implemented our networking library, and resolved all of it’s multithreading and deadlocking issues we encountered on the way. We’ve also since rewrote this code again and reduced it’s complexity further, greatly improving it’s maintainability and reliability. The code is now an extremely reliable foundation for the consensus library to be on top of.

We also successfully implemented network security, and all nodes now are able to transparently apply encryption and authentication for their network communication. We did have to change during implementation from certificate based authentication to a zero-knowledge password proof due to reduced complexity and library availability. We’re now using the tried and tested Microsoft’s own .NET Framework Cryptography libraries for our security procedures.

#### Implementing a Distributed Consistent Log AND Implementing Fault Tolerance

We have successfully implemented the full Raft Consensus algorithm into our project, and that has allowed us to not only achieve a distributed consistent log, but also maintain a fault tolerant service.

#### CCRD Use Case In Chosen Architecture and Production Environment

We’ve successfully implemented the CCRD, using our chosen architecture and in our production environment. This has been thoroughly evidenced above in our Test Results and Executable Architecture sections.

### 

### Status of Project Risks and Mitigations

|  |  |  |
| --- | --- | --- |
| **Risk** | **Open/Closed** | **Status** |
| Estimates for milestones are inaccurate | Open | We are aware our previous estimations have been inaccurate during iteration plans, this has been identified as lack of experience doing this type of programming work. However upon implementing our contingency strategy of spending time analysing our estimates, we’ve been able to increase our accuracy fourfold. We expect this increase in prediction accuracy to continue to increase as well. This is further discussed in Issues Encountered below. |
| Code quality issues | Open | This relates to not being able to “ship” a functioning product at the end of the project. The risk of this is greatly diminished as we’re displayed we’re successfully using style guides, Peer Programming, unit testing, and have now implemented the hardest/largest parts of the final functionality. However, technically we could still bring about this issue on ourselves next session, so we’re leaving it open for now, but setting the probability lower. |
| Poor software quality | Open | Neither symptom or trigger has occurred, mitigation of steps mentioned above are working to ensure code quality is high and those steps are shown to be working. |
| Security too complex | Closed | We’ve successfully implemented security into the library. It was initially too hard, as we predicted, so we changed to using a zero knowledge proof authentication method, and Microsoft’s own security algorithm implementations rather than a library. |
| Prototype failure | Closed | We’ve successfully implemented this library into a standalone Prototype |
| Multithreading introduces high level of difficult in troubleshooting | Closed | Although it’s been absurdly difficult, there are currently no bugs or deadlock issues in the code we’re aware of after extensive testing. We aren’t looking to implement any more features which change the functionality of any multithreaded flows, so we’re confident we won’t have issues in this area again. |

### Issues encountered

#### Debugging Multithreaded Applications

Although the risk is marked closed now, we’ve spent tens of hours debugging various issues relating to multithreading issues during this elaboration phase, including deadlocking issues between threads. Issues both in the multiple threads of networking, and the multiple thread of consensus. We had to learn to use our tools better to debug issues over multiple (10, 20, 30+ threads), we had to learn to design better, we were forced to implement an extensive trace level logging system to follow code flow, and we had to research deadlock avoidance techniques and implement/audit a locking order for objects to avoid those scenarios. Although we’re better for it now we’ve learnt and completed those challenges, it certainly showed that the difficulty of this project was an unreasonable expectation for university students. If members of our team did not have previous experience with multithreading, were not strong developers, or simply didn’t have the time to sink into solving these issues, this certainly could have been the end of our project and sent us back to the drawing board.

#### Accurate Programming Task Estimation

As we can see from our iteration plans the only consistent thing we could predict is that we’d predict incorrectly. We did however innact the Contingency Plan of spending time trying to iteratively identify root causes, and better calculations for estimating. We improved our estimations from ~400% out to ~110% out, and we can more reliably predict times using a “double it, plus a bit” method. This root issue is we don’t have the real world experience to accurately estimate how longs things will take yet. We’ll be keeping aware of this during future estimates and continuing to attempt to improve.

### Current Progress of Project

We’re confident in considering our project status as “very good”, and we believe we’ll be able to reliability follow through into our Feature and Transition phases next session to successfully deliver the project.This is primarily due to successfully overcoming most if not all the major obstacles/risks in the project already (i.e. networking, security, multithreading, and consensus algorithm complexity).

Looking ahead to next session, we’ll settle into a more relaxed agile/iterative development of simply adding features to the existing code base. Our biggest risks next session are maintaining code quality standards being adhered to, and overall reliability of the library. To mitigate these risks first iteration next session is focused on conducting a static code review of the code to focus on reliability, and then increasing unit testing coverage to full coverage to enable probability of that desired reliability. We’re confident this will dissolve most if not all of the rest of the risk associated with the project.

We’ve successfully achieved our Inception and Elaboration Phase goals, and we’re ready for next session’s challenges.