**Structured Analysis Report – The Godfather**

*Outsourced – Vince Seely, Tod Jones, Dakota Methvin*

1. **Introduction**

The-Godfather is a general-purpose, multi-featured Discord bot written in C#. Development began in August of 2017 and features have regularly been added/refactored/improved upon since. The project is active on GitHub with an average of 30 commits per week. Currently, this codebase appears to be suffering from bloat and patchwork development. Within a few minutes of browsing the repository, Vince was able to identify several areas in need of refactoring, or areas that would benefit from complexity-reducing rewrites. Additionally, as this is a general-purpose bot, there are several features we believe would add to its usefulness.

1. **Current Issues**
   1. **Known Bugs**

At this point, no bugs are known. We are in the process of designing tests and ensuring complete coverage of areas selected for maintenance tasks. Several areas are key candidates for the introduction of bugs, but until we have a running baseline, these issues are theoretical at best.

* 1. **Possible Enhancements**

Identified enhancements can be found in our Maintenance Plan documentation at <https://github.com/VinceSeely/the-godfather/blob/master/Documentation/CS3860%20Deliverables/Maintenance%20Plan.docx>.

1. **List of System Documents**

All documentation can be found in the project repository at <https://github.com/VinceSeely/the-godfather>.

1. **Structured Analysis**

**4.1 Structure Chart / Class Diagram**

Our analysis of this project based on the SciTools Understand generated class diagram was inconclusive. Two major issues arose, one the fault of the existing project, one partly the fault of the Understand tool.

Firstly, the existing project is a tangled mess with no clearly pre-defined structure. While there may have been a direction originally, feature bloat and lack of documentation has led to an overly complex object structure. Where we expect to find programming best practices (like layering the system when possible, providing flexible extension hooks for modules, and managing complexity by keeping coupling low), we instead find classes exporting their responsibilities on a per-module basis to other, single-use instances.

Utilizing a proper MVC architecture would resolve most of these issues, but the amount of time and work required to restructure the entire project is outside the scope of our class.

Secondly, this sprawling pseudo-architecture is so vast that Understand has trouble rendering it all. A copy of the generated UML Class Diagram can be found in the project repository at [https://github.com/VinceSeely/the-godfather/blob/master/Documentation/CS3860%20Deliverables/Understand%20Graphs%20and%20Reports/TheGodfatherUMLClassDiagram.png](https://github.com/VinceSeely/the-godfather/blob/master/Documentation/CS3860%20Deliverables/Understand%20Graphs%20and%20Reports/TheGodfatherUMLClassDiagram.png%20).

The detail contained in this file is too fine to be displayed even at the image’s massive 53391px by 936px full size. (Fun fact, that’s over 50 megapixels which is larger than all but the highest end professional cameras.) When attempting to navigate the in-application vector, Understand has inoperable lag even on a high-end desktop processor with ample memory.

**4.2 Call Graph**

Interestingly enough, a majority of the calls in this project are to the asynchronous local database the original developer implemented. There appear to be several switch-style calls inside, however, which inflate the apparent complexity but may not represent computational load beyond maintaining the call stack.

The rest of the calls appear to be synchronizing the local instance of the bot across all users in the Discord channel. This is likely hooked through Discord’s provided services.

A copy of the generated Call Graph can be found in the project repository at <https://github.com/VinceSeely/the-godfather/blob/master/Documentation/CS3860%20Deliverables/Understand%20Graphs%20and%20Reports/TheGodfatherCalls-Main.png>.

**4.3 Data Dictionary**

The generated Data Dictionary constitutes 489 pages of undefined classes, variables, and methods. Per the norm for this project, nothing is documented.

A copy of the full report (Data Dictionary is the first entry in the file) can be found in the project repository at <https://github.com/VinceSeely/the-godfather/blob/master/Documentation/CS3860%20Deliverables/Understand%20Graphs%20and%20Reports/TheGodfatherReport.txt>.

**4.4 Other Metrics**

Project Complexity: nearly all classes in this project have low complexity, with a few exceptions introduced by extensive path options (likely switch statements to handle input or matching operations). While this means individual classes are not difficult to optimize, the downside is that the project as a whole suffers from high coupling.

File Metrics: the average class contains 100 lines of code and 0 comments.

1. **Feasibility Analysis**

As we have not identified any critical bugs in this project (and likely will not as it is still maintained by the original developer), we are firmly set on other methods of maintenance. As outlined in our Maintenance Plan, we plan to reduce dependencies on external libraries which would reduce complexity between classes in the project. As identified above, the project has no clear handling and reintroducing this functionality to the project’s core would provide a single anchor point for extensibility.

Since this action would be refactoring (vs rewriting), the potential for introducing bugs or losing functionality is minimal. Vince would be spearheading this effort and would be assisted by Tod. They have identified DSharpPlus (a Discord extensibility library) as the artifact they intend to remove.

This maintenance would also accommodate perfective actions, since modules can focus on interacting with the bot instead of handling networking and Discord hooks. Dakota’s efforts will be focused on adding a new module as a proof of concept for this claim. He has identified the Ubisoft player statistics API as the functionality to implement. This would provide an opportunity to test the refactoring’s effectiveness and lack of volatility.

1. **Maintenance Plan**

Our Project Plan can be found in the project repository at <https://github.com/VinceSeely/the-godfather/blob/master/Documentation/CS3860%20Deliverables/Project%20Plan.mpp>.