Peer Review

# Jack Perrott-Webb jp222rz

## **Architecture**

**The model seems to have a clear separation from the view due to the class referred to in the class diagram as “controller.User”. Because of this the model should not be coupled with the UI because everything supposedly communicates through the controller and view.**

**Since the code is not finished, it’s impossible to tell what kind of UI the application was intended for. Since it’s made in C++, it could be used both as a console application as well as a BackEnd web application.**

**As far as we can tell from the code, the member Id variable does not extend usage to other classes.**

## **Source Code Quality**

**The code follows the standards for C++ and follows declarations and indentations well.**

**Some of the classes are named in plural (Members, Boats), which is not considered good standards.**

**The line “registry.open("registry.txt", std::ios::app)” reappears in both the Members and Boats class – could this have been derived into a new method which could have been used for both classes, instead of repeating the code?**

## **Design Quality**

**Member Id’s are only used in the Members class, therefore the application does not seem to rely on them to connect objects between classes.**

**The classes are not finished and so it’s difficult to say which responsibilities they would eventually get or how large they would become – high cohesion is therefore close to impossible to determine.**

**Classes seem to have low coupling and are currently connected to as few entities as possible.**

**The application code does not seem to contain any global variables or operations, however there are no encapsulated information, private objects or variables in the code – everything is public which could cause big problems since that means that information in other classes are practically accessible from anywhere in the application.**

**The application does not use primitive datatypes that could be mistaken for complex domain concepts.** (p.317, fig. 9.22\*)

## As a developer would the diagrams help you and why/why not?

**The class diagram seemingly gives, at first glance, a good overlook of how the application is built, with which language and how information in the application are stored (.txt files).**

**However, the sequence diagrams does not display interaction between specific classes and switches between class names such as User and MVC entities like Controller.**

**Since the model consists of 2 model classes, it would also have been useful to know which one of these that are called at which moment in the sequence, but these two are never mentioned in the sequence diagram, even though it seemingly is meant to display every single primary use case that can be reached within the application.**

**Furthermore, when comparing the two diagrams to each other, evident differences become clear:**

**The class diagram displays the User class as the controller, but the sequence diagram shows User and Controller as two separate entities – why?**

**The sequence diagram also shows something called “Members Database File” – this is not mentioned at all in the class diagram, which instead displays the two classes “BoatDatabase” and “Boats”, which in turn are not seen in the sequence diagram.**

**When looking at the code itself there also appears to be a class named “Member” – how does this class differ from “User”?**

**Why are there classes that are different/absent from the two diagrams? It’s confusing and doesn’t give a very clear view of how the application is built or how the classes connect with each other during compilation, or how many there actually are.**

## What are the strong points of the design/implementation, what do you think is really good and why?

**Since the code was unable to compile due to it being incomplete along with the fact that none of us who are writing this review have worked with specifically C++, it’s very hard to say or find the good points of the implementation in itself apart from the things already mentioned.**

**It should be noted that the creator of this application knows and have informed us through the README file that the application is not finished, which makes the peer review much easier to write since it becomes less about finding errors and more about giving suggestions for future improvements.**

## What are the weaknesses of the design/implementation, what do you think should be changed and why?

**If User really is meant to be a controller it should be given another name since the current one is very easy to relate to the same class as Member and could cause confusion. A good idea would also be to give both the Member and the Boat class a controller of their own, to enhance the clarity regarding where from and when the different tasks are issued to each class.**

**If information about a member is saved to a txt file like boats are, then perhaps the User class should have its own Database class as well?**

## Do you think the design/implementation has passed the grade 2 criteria?

No. Sadly, due to incomplete code and confusing, inconsistent diagrams, I think this design/implementation does not pass the grade 2 criteria.

**\*Sources**

**Overall Source:**

Applying UML and Patterns:

An Introduction to Object-Oriented Analysis and Design

And Iterative Development, Third Edition

By Craig Larman

ISBN : 0-13-148906-2

**Links on static variables/methods:**

<https://r.je/static-methods-bad-practice.html>

<http://blog.schauderhaft.de/2013/07/07/why-static-is-bad-and-how-to-avoid-it/>

**Link on dependencies:**

http://tutorials.jenkov.com/ood/understanding-dependencies.html