Peer Review of Workshop 2 on Jack Perrott-Webb

By Daniel Svensson, ds222ey.

I'm am using the e-book version of Larman's *Applying Uml and Patterns* which, annoyingly enough doesn't have pagination corresponding to the printed book. Therefore I include both the e-book (pdf) page number and the section or figure where the reference can be found.
Jack Perrott-Webb

As stated in the review this project isnät really finished so I will skip some parts and simply give feedback where it's possible.

Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction? Wrong relations? Correct UML notation?

From looking at the class diagram the classes are divided into namespaces ("model.X", Controller.Y" etc). This is not the case in the actual implementation. In the implementation there's also a Member who is missing from the class diagram but not the sequence diagrams. The sequence diagram gives the impression that a class called Controller does a lot of the work. This is not the case, that class doesn't exist. I guess it implies the Boat and Member as controllers even though they are also models. Actually looking at User, it seems to be the controller but then it can't exist as two entities in the sequence diagram. Program class is missing from the class diagram which would be expected if it was just responsible for starting the application but it actually seems to be doing a lot of work, controller work I think. BoatDatabase has, according to the CD, two Boat(s) through association. BoatDatabase doesn't exist and accordingly has no boats and no associations. The CD simply doesn't help at all and the sequence diagram aren't much better.

Architecture

Is there a model view separation?

Yes.

Is the model coupled to the user interface?

Models has no dependencies to views.

Is the model specialized for a certain kind of IU?

Yes. The Member and Boat store all properties as strings. String IDnum, string BoatsNum etc. These are typically ints or some other numeral and shouldn't be changed in the model to please the UI. The model shouldn't have a clue what the UI wants in terms of data types [1, p480, sect 18.4]. The GetCompactList() method and others also violates this and shouldn't be present in the model.

Are there domain rules in the UI?

No

Is the requirement of a unique member id correctly done?

No. It's not there.

What is the quality of the implementation/source code?

The code is simply not there for the most part. But the code that is there contains methods and properties named in a descriptive way.

What is the quality of the design?

Objects are connected using associations and not with keys/ids.

They aren't really connected at all. The Member has a string called BoatsInfo but no actual Boat objects.

Is GRASP used correctly?

Not really. The models have responsibilities it shouldn't have. The Member, of course, has knowledge about itself, and can provide neat strings with all its information collected but it has no reason to, it shouldn't know to do that. The Program, the starting point and owner of the Main method does some kind of port pertaining to the menu, I think. I don't know why though.

It's hard to say about cohesion since so much is unfinished. On a glance they seem cohesive but nothing is actually done so it's hard to say where it will end up.

Actually, the coupling seems to be too low in that it doesn't really exist any association between Member and Boat. The fact that the models' properties are all string avoids coupling from the view to the model but it also violates standards and really presupposes the model having knowledge about the view.

No static or global elements and no hidden dependencies that I have noticed, which is good.

The information isn't really encapsulated at all. Everything is public in the Member and Boat [1, p918, Glossary: 'public].

The member seems to have boats in some sort of BoatsInfo-string. This really should be a class. And it is, just not used by the member.

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

No they wouldn't. The class diagram is no help at all since it doesn't conform at all with the implementation. If I were to develop from scratch using the class diagram I would end up with no Member(s) and a BoatDatabase containing two Boat(s).

The sequence diagrams show about everything that can be done at once. They really should be divided into separate diagrams, focusing on one use case each. Also it's hard to understand who is who when comparing sequence diagram to the implementation.

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

Descriptive naming of methods, classes and properties. Perhaps a bit unnecessary to name a Member method "m_AddToFile" and not just "AddToFile".

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

There’s a lot to be done before this can be answered correctly. The intentions are also hard to figure out since the sequence and class diagrams aren’t really done either. So the first step, I would think, is to get the idea of the application down on paper as an actual class diagram and a couple of sequence diagrams. Other than that, at this point, the biggest weakness is that the model seems to know what kind of UI it’s dealing with based on its output.

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

No.

References

1. Larman, C, Larman C., Applying UML and Patterns: An introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition, 2004, ISBN: 0131489062. E-bok: https://aanimesh.files.wordpress.com/2013/09/applying-uml-and-patterns-3rd.pdf