Comparing Software Documentation Approaches

Matthew Wisby & Isaac Schemm **Faculty Mentors: Dr. Joline Morrison & Dr. Mike Morrison**

Department of Computer Science, University of Wisconsin-Eau Claire

Research Question

Our research problem is: what is the best method to document software? At UW-Eau Claire's Computer Science department, students have written software in the Java programming language that facilitates peer review of computer science assignments. The software has been previously written by other students, who have since graduated. Documenting the system is important because it helps users of the software to better understand how it works and to make the process of modifying the software easier. There haven't been previous attempts to document the peer review software, so this is the first time it has been attempted.

We are unsure of which method would best increase our understanding of the system, but we expect that class diagrams will help the most for developers working on the software, and use-case diagrams will help the most for students and instructors using the software. We also expect that the front-end (user interface) aspects of the software will be better described by use-case diagrams, while the back-end (database) aspects will be better described by class diagrams.

Method

To document the software, we explored each class of the software and documented how they relate to each other. We used two methods, Unified Modeling Language class diagrams and use-case diagrams, to document the components of the system. We then came up with possible situations in which users or developers might want to learn more about the system, and we determined which of the two methods gave more valid information in each case.

Unified Modeling Language (UML) class diagrams

The Question and IDItem classes are two of the simplest classes in the peer review software. The first set of values represent the variables each class uses-the name of each variable is preceded by its type. The second set of values are the methods of each class, which are sections of code that can be run by simply calling the method's name on an instance of the class. These types of diagrams allow the user to see the class's complexity and what types of values it holds.

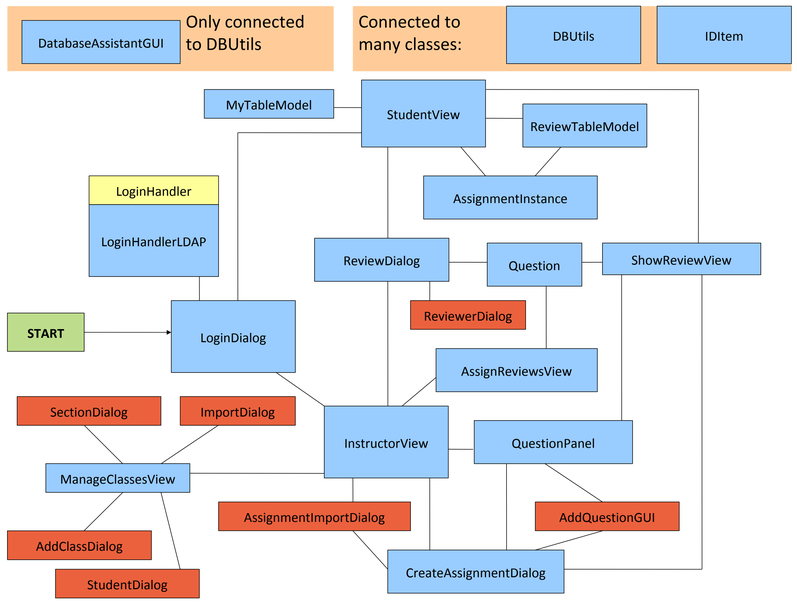
The plus sign (+) before each value indicates that it is public – any other class can access a public variable or method.

Question
+int questionType +String questionText +ArrayList<String> questionChoices
+boolean equals() +String toString() +void insertQuestion()

IDItem
+int id +String name
+boolean equals() +int compareTo() +String toString()

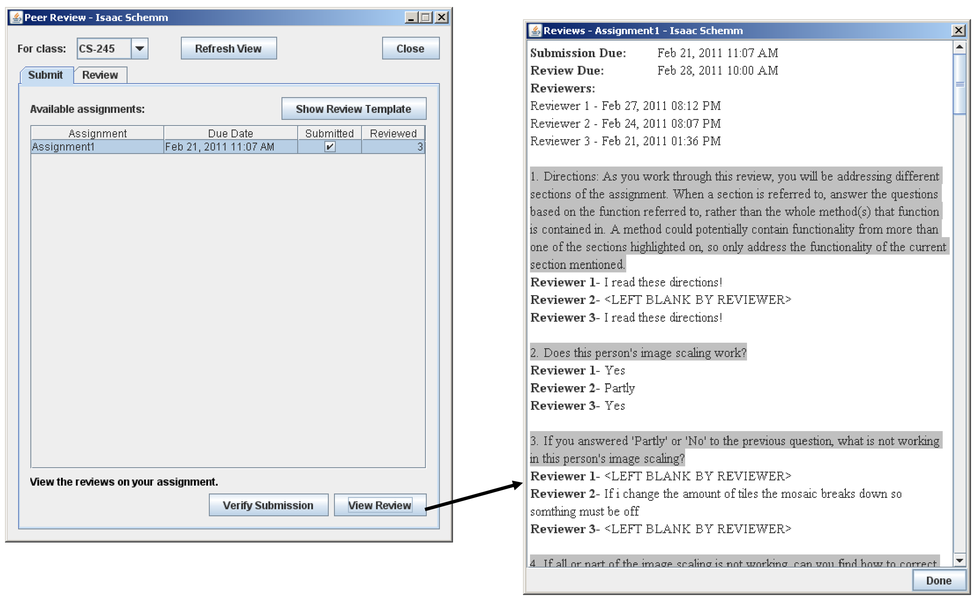
An example of a simple class diagram

This diagram shows the names of classes and the connections between them, but does not give information on the variables or functions stored in the classes.



Use-case diagrams

Use-case diagrams show how a particular program can or should be used. This partial use-case diagram uses screenshots connected by arrows. Another option would be to give a name to each dialog and use those names on the diagram instead of screenshots.



Conclusion

We determined that both types of diagrams are useful in certain situations.

Class diagrams are very helpful for developers working on the code, but not as helpful for end-users (in our case, teachers and students) because they give information that would be unnecessary.

Use-case diagrams are helpful for end-users because they show how to use the software's interface, but not as useful for developers because they don't show the internal structure of the program. However, use-case diagrams can help developers who are unfamiliar with the code. Use-case diagrams can also help with troubleshooting bugs.

We expect that class diagrams will be the most useful to us, since we are maintaining and improving the code.

University of Wisconsin Eau Claire

Acknowledgements

We would like to acknowledge the other students who have worked on the peer review software: Brandon Holt, Luke Komiskey, Daphne Brinkerhoff, Greg Boettcher and Hannah Miller. We would also like to thank the Blugold Fellowship for providing the opportunity for us to work on this research project.