The Relation between Source Code, Documentation and Testing

Ling-Wei Wu Thomas Wohlfarter

University of Innsbruck

Summer Semester 2017

Lecture:

Introduction to Scientific Working

June 1, 2017

Contents

Source Code, Documentation and Testing are often considered as important as the article in "Unholy Trinity of Software Development". However, what is the relationship among three of them? Should they be separated in different files or is it better to keep them close to each other? Should we use code generators? Which option is more convenient? Is it really necessary to document and test the source code of a project? Based on this paper, "Unholy Trinity of Software Development", [1] it will explore answers of the mentioned questions. In this article, "Unholy Trinity of Software Development", there is a conversation between the computer scientist, George V. Neville-Neil alias "Kode Vicious", and a software developer, who is asking KV about his opinion by some related questions.

1 Introduction

There are two kinds of software developers. The first kind of develops document and test their code strictly and carefully. The other kind of developers for whom documentation and testing of code is just a pain in the neck. The second group sees the increasing effort of writing a program. Regardless of these arguments, these two components belong to a software project as the actual source code itself. However, as every software developer knows, this is a controversial issue and worthy to explore. How these three key components of software development relate to each other? In the first section, we will concentrate on the relation between the documentation and source code. In the second part, the issue will focus on the testing issue of source code. As the previous section, there will be some arguments for and against separating testing from the code. The last section in this paper, there will be a conclusion of the topics that were discussed before. In the end, we provide an overview of the argumentations in Table 1 and Table 2 based on precious sections discussed.

2 Source Code and Documentation

There are many reasons which provide support documentations of the source code. The most important one would probably be the simplified cooperation between different software developers. When working in a big software development company or a huge developing team, it is very likely that any programmer is not the only one who needs to see his or her implementation. Many developers are working together on one project. Documenting the code helps other programmers to understand the source code. Also the service and refactoring process of software is simplified with a good written documentation. A software project often takes a lot of time until it can be delivered to the customer. During the period, the decisions have to be made by the implementer, who very possibly and easily forgets important details when there is no documentation throughout [4]. Also the tests of a program have to be described in its process with documentation. However, documentation does not automatically lead to a better software project. Documentation can be as useless rather than useful. There are some rules for writing good documenta-

tions which are not redundant or unreadable [2], [5].

The software developer in the paper [1] claims, that some of his coworkers refuse to write documentations inside the source file. They argue that documentations make the code unclear when programming, because the lines used for documentations reduce the overview of the programming environment. "Kode Vicious" counters the argument with modern programming environments and text editors which have special features. For example, the feature, so-called code folding, which every modern editor offers, would break the argument with the less space problem. Nevertheless, sometimes much documentation also leads to different problems. For instance, the readability could be reduced. Since there is less contrast between documentation and source code, important comments could be ignored. As mentioned above, documentation is not always useful. Source code could possibly be written in a way that it is self-describing and this seems to make documentation redundant. This aspect would also be mentioned from the coworkers in the paper who like their documentation in a different file or even in another different directory.

In addition, the good reason, why the source code and its documentation should be located near each other, is because of making less effort, especially when developers are updating or refactoring the code. When the functionality of a program needs to be changed, the programmer may have to rewrite classes or functions. This would mean that the documentation also has to be changed. Therefore, having the documentation in the source file or in the same directory would be more convenient than updating in two different files, which may be also located in different directories. Another less important reason would be stated in the following. Software developers, who look into the directories of a program at the first time, have to get an overview of the programs structure first. Searching the documentation at a different location from the source code which should be documented can cause some struggles or troubles for the programmer. When developers need to find the right documentation for the corresponding right code, it is simpler that in the beginning, just putting the code and documentation together in one file. The documentation of a function usually has the variables and types described. When the programmer skims through the documentation and reads the variable, it is easier for him or her to switch to the code, which is usually located below. So the programmer gets a better view of how the code is written and how the code is documented. In the other word, when the documentation is located in a complete different directory or far away from the source code, it would be irritating because the files have to link to each other. It would make updating or refactoring just more complicated and inefficient when paths have to be set to the two corresponding files.

3 Source Code and Testing

Definitely, testing for the source code is not a fewer important component as the documentation. Tests ensure that a program behaves the correct way. Writing tests makes a guarantee not only for the implementer but also for the customers who pay money for

the software and expect that the software works appropriately. Like the documentation, testing can be separated from the source code. Testing could be located in the source file or in a complete different directory. At the same time, testing also could be redundant when it is not done correctly [3].

Testing can probably be a bit easier to separate from the code than documentation. In the following, we firstly propose some advantages of separating testing and the code. When developers have to test a function or a class, they have to think in the very different way because they were implementing the source code. In a software project, the code implementer and the tester sometimes have different tasks. The programmer of the source code does not test her or his own code. That is because the tester has a different point of view. This kind of cooperation makes it easier to find bugs or logical mistakes in the source code while two persons see the program from two aspects. Some programmers and computer scientists [7] even suggest to write the tests before writing the source code. This would make every component of the source code more independent. Another point of view against inline testing is the low contrast between the actual code and the test part. This could lead to confusion and reduced readability although tests are normal signed with a special kind of syntax. Tests can also be linked to its corresponding source code [6] with tags. The number of software tests is an indicator of stability and resistance. The more parts of a program are tested, the more stable the program is. However, when the software delivers to the customers, they do not need to see these tests of the actual program. While the customers use the software, they just need it works correctly and precisely. Therefore, testing is not necessary to be put into the source code. It is also easier to have them separated from the source code. Readability is a big issue when it comes to inline tests. Sometimes tests can be a lot larger than the code, which should be tested. Having them in the source file would reduce the readability fatal.

We could also find one strong argument supporting tests in the source file as well as documentation. Having testing and the code separated makes the tests easier to get outdated. Changing the source code means the developer sometimes need to update the tests at the same time. If they are separated, updating the tests has more chance to be forgotten.

4 Conclusion

When it comes to the question "Should documentation be separated from the source code?", there are some proponents and some opponent. Both sides have strong and weak argumentation (Table 1). When evaluating these aspects, we consider the point of view that documentation should be in the source file. It makes it simpler for every personnel who has to work on the program or use it. As researching this topic, there are much more proponents to find than opponents. Most of the critics are generally against documentation. Nevertheless, as every software developer knows, source code has to be documented no matter where the documentation located in. When it is asked to

separate tests from source code, we come out a different conclusion. There are nearly no arguments supporting inline testing in the source code found (Table 2). Tests should be located in an extra file. "Kode Vicous" from the paper "The Unholy Trinity of Software Development" [1] has an similar point of view in this topic.

4.1 Overview

In this overview, we provide two tables, which compare documentation and testing included "+" or excluded "-" in the source files, based on the previous discussion.

keywords	+	-
reduced view	×	
reduced readability	×	
simple updating		×
not as convenient		×
no searching		×
paths to docs		×

Table 1: Documentation

keywords	+	-
simple updating	×	
specific tester job		×
better readability		×

Table 2: Testing

5 Future work

There are already some tools like *Doxgen*, which help to separate documentation from source code. Other tools link documentation and source code with tags, so that it is not that inconvenient as mentioned previously. There are more new, modern text editors and new features developed constantly, which significantly simplify the programming process. Consequently, in nearly furture, we assume that there will be more tools developed with good managements of software developments. Untill then our controversial issue of separating documentation or tests from code is not a topic to argue over anymore. With these future tools, everyone can decide by his or her own.

References

[1] George Neville-Neil, 2016, *The Unholy Trinity of Software Development*

[2] Java, SQL AND JOOQ, 2013, *The Golden Rules of Code Documentation*

[3] Code To Joy, 2008, *The Golden Rule of Testing and JUnit Assumptions*

[4] Dennis Brandl, 2011, *Engineering and IT Insight: Keep documentation in sync with code*

[5] Douglas Kramer, *API Documentation from Source Code Comments: A Case Study of Javadoc*

[6] Dietmar Winkler, Stefan Biffl, Johannes Bergsmann, *Software Quality. The Future of Systems- and Software Development*, 8th Edition

[7] Ashfaque Ahmed,Bhanu Prasad, *Foundations of Software Engineering*