2DV608 Assignment 0

Software Design





*Author:* Yetnayet Belachew

*Supervisor:* Mauro Caporuscio

*Subject:* software design

*semester:* 2022

*Course code:* 2DV608

**Task 1 – Software Design**

Read the “No Silver Bullet - Essence and Accident in Software Engineering” paper by F.P. Brooks. Reflect on the impact it has on software design in general. Enumerate three core problems (among those discussed in the paper) and for each problem two to three possible mitigations.

According to the article "No Silver Bullet - Essence and Accident in Software Engineering", there is no such thing that can guarantee that an extreme solution can be found for a problem within the field of software engineering. In other words, there is no single methodology that can be considered as the most effective technique for solving software problems. No piece of software built in the past is error-free or fully reliable. Software engineering difficulties are divided into two categories by Fred Brook, the Essence difficulty and the Accidents difficulty. Software complexity and implementation difficulties are considered essence difficulties, or those difficulties that are inherent in the nature of the software.

Accidental difficulties occur during the production of software. They are caused by the complexity introduced during the process of translating concepts into a programming language.

Beyond bringing awareness to the fact that good software depends upon good design techniques as well as there was no silver bullet for software engineering, t

The writer specifically mentioned that no piece of software is error-free, and instead that it should continue to grow as it is built. This concept is carried out by the open-close principle, which I believe has a key role to play in the modern age of software development.

The three core problems in software engineering that need improvement are productivity, simplicity, and reliability.

**For example**

**Software reliability** is influences by the number of errors that are committed by the programmer. The fewer errors and failures in the software, the more reliable it is. By designing software that is easily changeable and maintainable, software designers can minimize the problem of software reliability. In his article "No Silver Bullet -Brooks stated, "All successful software gets changed". Another way to reduce software reliability problems is to reduce complexity. The more complex the software becomes, the less reliable it becomes. Simplifying software can improve maintainability and flexibility, reducing complexity and reducing software reliability.

# Software Simplicity

The opposite of complexity is simplicity. The simpler the software, the easier it is to maintain and change. As a result, it could turn out to be more reliable. To improve the software's simplicity, the complex methods of designing it need to be accountable for reducing it. This will improve the software's

productivity. However, simplicity does not mean writing the code as quickly or delivering the product as soon as possible; unless there is a higher chance of introducing bugs. How can you ensure simplicity?

* Follow the SRP principle: do one thing and do it well, unless you must split your code if there is more than one issue at hand.

# Software performance

really depends on software complexity. less complex or simpler code will be easy to maintain and easy to test. Subsequently, the level of performance increases.

**Task 2 – The Design Question**

Read the three chapters by F.P. Brooks (The Design of Design - Essays from a Computer Scientist). Reflect on the software design. Enumerate at least two problems that you recognize and describe when you experienced this and how you found a workaround.

In an article in The Design of Design-Essays from a Computer Scientist by FP Brooks, the author states that design has three phases. Idea, implementation, interaction phase. The value of conceptual design comes first, and good design has conceptual completeness, economy, and clarity. The writer provided advice that argued that frequent discussions of design concepts would be of great help

to communication within the design team. The ultimate goal of uniting the idea and goal can be achieved. Brooks considers that "studying the design process can helps anyone get to be a better designer and teach others how to design.". Before you start coding or formal drawing, you must first plan the entire design process. The most serious drawback is that designers often have vague, incompletely specified goals, or main objectives. The solution to this is the concept of "who to plan first the whole design process before starting coding or formal drawing”.

As Brooks said, "The hardest part of designing is deciding what to design." Designing the entire process is like having a clear roadmap for your goals. As a result, you can avoid ambiguity between teams. The writer shared his experience as a programmer when he started coding without planning to design for a project he had to do. And the result was a waste of effort.

Modeling and specifying the requirements of a real-time software project begins with investigating and analyzing the specific problem. You can then collect specific requirements and design your software project. However, as the writer argued, the client needs the help of the software designer in deciding what needs to be included in the requirements specification. The writer argued that "the designer's main service is to help clients discover what they want to design." Therefore, by supporting the client's decision, the requirements specification becomes clear, and then the design process of the entire project becomes clear. After that, the team has a unified purpose for the project.

References

[1]. No Silver Bullet - Essence and Accident in Software Engineering” paper by F.P. Brook.

[2]. “The Design of Design - Essays from a Computer Scientist” by F.P.

Brook.

1(1)