**Name:** Loic Galland (lg222sv)
**Word count:** 954 words (excluding titles and assignment questions)

## 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 and for each problem two to three possible mitigations.*

I will begin to talk about the first paper "No Silver Bullet - Essence and Accident in Software Engineering" by F.P. Brooks. I will go through three core problems in Software Engineering by showing what they are and show possible mitigations.

Designing software is no easy task. Complexity is one of the core problems which makes it so difficult. The complexity of software is huge due to its enormous number of different states that all the functionalities can take. If we just look at variables in Java, each variable has a different state. A boolean variable can be 'true' or 'false', an integer can have numbers from –2,147,483,648 to 2,147,483,647. Complexity also comes because all software is coded in a different way and by different people. Reading someone else code is not an easy task, to grasp the meaning or how their program is working.

There are many ways to reduce rather than resolve this problem. One of the possible mitigations is Unified Program Environments (UPE) such as Unix. Unix is an OS that attacks the accidental difficulties of using programs together. It works with integrated libraries, unified file formats, pipes, and filters. In addition, the OS allows multitasking where multiple users are connected to the same machine at the same time. This is letting programmers be more effective and reduce the complexity of connecting multiple programs together.

Another mitigation to be considered is expert systems. An expert system is constituted of a generalized interface engine and rule base. The interface engine deals with fuzzy or probabilistic values and rules. It is also developed in an application-independent way. The only changeable part of the application is encoded in the rule base. Tools are created to help with developing, changing, testing and documenting rule base. By doing this, it helps to reduce accidental errors which reduce the complexity of the software.

Another core problem is changeability. Software always needs to change due to the pressure for extended function as users use the product on the edge or even beyond normal use. The software also has a greater change rate compared to cars for example. Where cars get superseded by a newer model, the software is just changed or updated to include the new functionalities, fixes… With new technologies, software needs to adapt and redesign their application to make them compatible.

Using MVC (Model View Controller) models can help to make it easier to change/update software. An MVC model works by separating the models, views and the controller. This allows the controller to stay the same regarding the view and therefore not dependent on it if the software view needs to be redesigned. Using this model allows the software to change without impacting its core functions.

Avoiding hard-coded variables could also be mitigation for changeability. By coding the methods in such a way where the key variables can be changed from one place in the code. Instead of having to change all the instances of variables in the code. This will improve the changeability of the software and make it more efficient for developers to change the software.

The last core problem is the invisibility of the software. To this day we have no accurate way to visualize methods or software used because coding is abstract lines of code. A powerful tool that humans have is the visualization of geometric objects, which become difficult when the software is just abstract. Therefore, it can be hard for clients to create the exact thing they want.

One of the mitigations of this issue is using prototypes. By creating a small prototype of the important functionality where developers focus on the function rather than the design. This will allow the user to visualize the idea that the developers understand and compare it with their original idea to help the developers make exactly what the client wants.

**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 this answer, I will start to talk about the second paper *The Design of Design - Essays from a Computer Scientist* by F.P. Brooks. I will explore two problems that I have encountered and explain how I managed to solve it.

To begin with, the biggest problem with young developers is experience. I can't remember the exact moment where experience really impacted severely my chance of finishing software. But rather multiple while programming that I have picked up from previous experiences. From encapsulation to design models, … Recently I have worked on a Design project using MVC models and I finally understood how they were working and now I am able to recreate very fast. With today's technologies, the best way to solve some of the issues we are facing is to search on the internet for the answer. Most of the problems due to inexperience were solved by using the help of the internet and to the people behinds answers or solution on forums.

To continue, another problem that many, if not all developers face, is that the constraints keep on changing as the development goes. I experienced this problem during an assignment where I was supposed to create the Hangman game. As I was going with the project, I realized that some of the constraints that I had prepared earlier needed to be updated to fit exactly what I wanted. I needed to update one of the rules to make the game more enjoyable. I had to set again the rules and constraints to match my new vision of the program. To solve the problem, I changed the way the method was working to match it to the new requirement. I also realized that these steps are needed to be able to create software that the client wants and not only a vague idea of how the client wants it.