VT21, 2DV608

Assignment 0

Xingrong Zong

xz222bb

Task 1: Software Design

Reflect the impact it has on software design in general. Enumerate three core problems and for each problem two to three possible mitigations.

This paper was written by F.P. Brooks. It clearly points out that software engineering needs to be improved. However, the difficulties that software development faces are sophisticated. The past developments mostly were done by removing artificial barriers that have made the accidental tasks inordinately hard. As all software construction involves essential tasks, accidental tasks, and abstract software entity, that still only focuses on accidental tasks will not give an order of magnitude improvement.

This paper provides a well understanding of software engineering development and the difficulties it is facing. It explains and summarizes the difficulties from different stages and perspectives, which also gives many suggestions and ways to improve software, it stresses the importance of software design in software progress and software engineering development.

First core problem: there are no tools been invented to help with software productivity, reliability, and simplicity what electronics and transistors did for computer hardware. Also, software progress cannot catch-up computer hardware progress because it is so fast.

Second core problem: to improve software, the essence (the difficulties inherent in the nature of the software) of modern software systems need to be considered: complexity, conformity, changeability, and invisibility. Also, the accidents (the difficulties that today attend its production but that are not inherent).

Third core problem: the essential complexity of software developments increases nonlinearly with its size. The more complex, the more difficult to enumerate all the possible states of the program and more unreliability. Therefore, from the complexity of the functions and the structure comes the difficulty of using and extending programs.

To improve software productivity, reliability, and simplicity, the most powerful ways is to progressively use high-level languages for programming. It fully eliminates a whole level of accidental complexity in the program. A high-level language accomplishes the most for the level of human’s sophistication thinking about an abstract program, which consists of conceptual constructs: operations, datatypes, sequences, and communications. In addition, unified programming environments can improve productivity. Because they attack the accidental difficulties of using programs together, by providing integrated libraries, unified file formats, and pipes and filters.

Another technique is object-oriented programming. It consists of two separate ideas: abstract datatypes and hierarchical types. Both remove a higher-order sort of accidental difficulty and allow a higher-order expression of design. The complexity of software is an essential property, and such attacks make no change in that.

Most importantly, great designers are the key to improve the software art. As they interact and stimulate with each other to get these ideas.

Task 2: The Design Question

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.

The book was written by F.P. Brooks. It explains what software design is by using examples that adapt to daily life. It represents that to carry a software design, a design concept is needed first even it is invisible. Because it provides conceptual integrity: unity, economy, clarity; and helps designers to work together as a team. The book introduces different kinds of design, such as system design, artistic design, routine design, adaptive design, and original design. It lists the requirements that need to consider for the rational model of a software design with the following order: goal, desiderata, utility function, constraints, resource allocations, budgets, and crucial budgets, and design trees. This rational model is a systematic step-by-step process by providing clear steps for planning a design project. It furnishes clearly definable milestones for planning a schedule and for judging progress, which also helps the communication within the design team, and between the team and other stakeholders.

The book points out that in software design, the rational model is ideal but it is not practical. There are some problems that designers may face when they have software design, such as having a vague, incompletely specified goal at the beginning, figuring out the design tree along with the progress, the goodness function cannot be evaluated incrementally, keeping changing the desiderata and their weightings, keeping changing the constraints inside and outside the design space, and etc.

I recognize and have experienced a few problems from the above while doing some project as a student. First is when I started a project and made a model for planning, I did have a vague, incompletely specified primary objective. It happened from my ideal thought about the final product, at the same time having a misunderstanding of my lack of ability to reach that ideal goal and the key requirements for the product. Therefore, it became hard to work or to accomplish each milestone in the progress. After getting the feedback from the professor and other students, I realized the main thing I need to do is to simplify the goal I set and focus on the foremost function, which also needs me to understand and make the decision of the project about what the foremost requirement was.

The other problem I have experienced was figuring out the design tree along with the progress. I noticed I was not able to make the design tree beforehand, as there are alternatives and more alternatives come after step-by-step. It would not be decided until it works and can support the next step, and even so, it still has a chance to come back later and make changes because of future difficulty. Then, my design tree was made along with the progress.

Reference:

F.P. Brooks, “No Silver Bullet – Essence and Accident in Software Engineering”

F.P. Brooks, “The Design of Design – Essays from a Computer Scientist”