Linnéuniversitetet
Kalmar Växjö

Linnaeus University

2DV608 - Software Design
Assignment 0

Author: Yuyao Duan
Email: yd222br@student.lnu.se
Course Code:2DV608
Semester: Spring 2022
Supervisors: Mauro Caporuscio, Francis Palma, Diego Perez

Table of Contents

1. Task 1

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.

1.1 Essential difficulties of software – complexity

One of the most obvious question regarding software engineering that Brooks mentioned in his paper is that the nature of software entities itself is very complex that is more complicated than any other human construct [1]. With the development of computer hardware, demand of human production needs for software increases in nonlinear fashion manner [1]. Experts should not ignore that the development of a software entity is not simply repetition of same elements in size but more complex interconnection logic may appear [1].

Since the complexity of software is an essential property, the industrial experts who want to come up with potential mitigations should firstly hold a **broader view** and **mind** to tackle the issues related the software complexity [1]. Furthermore, some **technical strategies** such as implementing better team communication methods can reduce product flaws, cost overruns, and schedule delays which could also reduce complexity of software development in long run [1]. Moreover, timely adjustment of **management strategies** can also be a critical aspect to mitigate software complexity [1].

1.2 Past achievements for solving accidental challenges

As Brooks pointed out, the most fruitful achievements to attack building software difficulties are related to the accidental challenges instead of the essential ones [1]. Industrial experts can put efforts on finding solutions for reducing accidental difficulties. Up to now there are series of technological progress having been developed [1].

The development of **high-level languages** can be considered as a significant development in computer science field [1]. It largely improves the productivity, reliability, simplicity, and comprehensibility of software programming which could largely free a program in concern bits, registers, branches, channels, disks etc. [1]. Furthermore, **time-sharing** is suggested as a major boost for productivity in the productivity of programmers as well as product quality improvement [1]. This due to time-sharing could preserve immediacy and maintain the overview of complexity of programs.

1.3 Requirements refinement and rapid prototyping

Another challenge part of building a software system is finding out precisely what to develop [1]. The detailed technical requirements consisting of all the interfaces to people, to machines, and to other software systems are the one of the most difficult part of conceptual work [1]. If any part of it done wrong, the whole system could be crippled and more difficult to rectify in the end [1].

To mitigate this potential problem, industrial experts can start to apply **incremental development** thinking i.e. grow, not build, software [1]. The conceptual structures of the software are too complicated to come up with accurate specifications in advance [1]. As Brooks points out, teams can grow much more complex entities in four months than they can build [1]. Another way to overcome this challenge is the organizations or teams should understand the importance of the **great designers** [1]. The good designers will normally follow good practices not the poor ones which could contribute with faster, smaller, simpler, cleaner structures with less effort [1].

2. Task 2

Read the three chapters by FP 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.

2.1 Starting a design without knowing the goal

In this article, Brooks pointed out a critical problem that designer often has an unclear, incompletely specified goal, which should be the primary objective of the focal design task [2]. Identifying what to design is the core of this stage. Without clarifying the design of real needs, a simple uncomplicated software project may fall into endless revising status which in the end lead to time-consuming, extra unnecessary financial cost, as well as frustration of both clients and software designers. In Brooks' case, a design of simple database system turns out to be a "long-distance running" with clients and the meeting with the clients turns out to be "squeeze toothpaste" for "real needs" [2]. Project presentation turns out to be *help session* for customers to decide what they really want [2].

This issue can be considered as our priority issue when my team worked with the project of 1DV508 the project course. A software program never comes alone – it always serves a business or operation of everyday life. Jumping out of the box i.e. the purely technical point of view and embracing business thinking was our shortcut to figure out the client's real demand. In most cases, similar successful projects can be found. We applied this thinking and identified the success factors from the previous projects. Then we used these experiences for our meetings with clients for the initial meetings to construct the product demands as clearly as possible.

2.2 Working with changing constraints

In this paper, Brooks pointed out a situation that greatly affects the software design which is known as "constraints keep changing" [2]. This means no matter how well the goals, demands, design tree, and functions were precisely identified, the design process is still an iterative process [2]. The constantly changing business environment forces designers and developers adapt to changes [2]. Designers should bear a flexible mind and tend to recognize the new opportunities which could benefit the project in long run [2]. One point worth noticing that Brooks gave a vivid example of divergent design thinking – changing design outside the design space which in some cases can significantly save cost [2].

Again, in our "Hotel Management Application" project of 1DV508, we had an issue regarding database design i.e. how should we design it to make sure it meets project requirements as well as practicable for us seven students from different places working in a remote manner. Initially we had trouble with providing a reliable solution since no matter how hard we tried it is very difficult for our project to be compatible with Gitlab. This is later on solving by actually a very simple solution – using container technique. Same as Brooks bought 5-foot strip of land from neighbor, we put our database in Docker instead of making rapid change to database itself. We successfully solved the database integration problem and ensured the project developed smoothly in a remote working approach.

References

[1] Brooks Jr, Frederick P. The mythical man-month: essays on software engineering. Pearson Education, 1995.

[2] Brooks Jr, Frederick P. The design of design: Essays from a computer scientist. Pearson Education, 2010.