演讲稿部分

Part 1

// 刘浩宇

Last class, we discuss software re-architecting and impact analysis. There are two types of software re-architecting: physical and logical. Impact analysis is a process to evaluate the cost and ripple propagation of system modification. It is important to note that functional enhancement should not be considered until the new architectural structure is available. A tool for impact analysis should be developed before re-architecting to identify areas of system modification and estimate the cost. Effective impact analysis is crucial for the success of software re-architecting. In summary, proper analysis and planning are necessary for successful software re-architecting.

Part 2

// 陈戒

Next, I would like to discuss the views of other scientists.

Software re-architecting is a crucial solution for large-scale systems like enterprise software. While rewriting smaller systems from scratch is feasible, re-architecting for large-scale systems is a challenging manual task with high stakes. Avoiding this task only makes future solutions more difficult, burdening software systems and our increasingly dependent society with flawed, inefficient, and insecure designs.

To ease the re-architecting process, various techniques and tools have been developed to support software evolution and maintenance tasks. These include, but are not limited to, program comprehension, change impact analysis, cost-benefit analysis, copy-and-paste code reuse, and automated software testing, all contributing to a more efficient and effective re-architecting process.

Part 3

// 陈实

This time we learned about software re-architecting. We are considering what benefits software re-architecting can bring to us? We believe that software re-architecting mainly solves the problem of complexity, and in software development, the lower the complexity, the better.

First of all, re-architecting should solve the complexity of the program logic, avoiding the emergence of branches as much as possible is an important way to reduce the complexity of program logic. Divergent branches will sharply increase the complexity of the program.

Secondly, should solve the complexity of architecture design, architecture design involves module design and system design, as far as possible to extract some common modules or subsystems, may be invoked by all other modules or systems.

Moreover, should also solve the complexity of system deployment, and if necessary, we can use a batch file to simplify the deployment steps and turn multiple steps into one step.

Finally, the complexity of technology, the development trend of technology is that, the simpler the development, the more powerful the function. Replacing the old technology with new technology can improve the development efficiency, and enhance the maintainability of the entire project.

PPT部分

Part 1

**Software Re-architecting**

1. Software re-architecting has two types: physical and logical.

2. Physical re-architecting reorganizes software structure without changing its functionality and external behavior.

3. Logical re-architecting reorganizes software structure without changing its functionality, external behavior, or source code.

**Architecture-based Impact Analysis**

1. Impact analysis is a process that evaluates the cost and ripple effects of system modifications.

2. The main focus of impact analysis is on the cost and propagation of system modification.

3.To avoid complexity, functional enhancement should not be considered until the new architectural structure is available.

4. A tool for impact analysis should be developed before re-architecting to identify system modification areas and estimate costs.

Part 2

* Software re-architecting is a necessary solution in the landscape of large-scale software systems, such as enterprise software. For smaller systems, it may be justifiable to rewrite a codebase from scratch when its architecture does not meet new needs. But rewriting a large-scale code base is too expensive and “totally out of the question”. A proper re-architecting can save the life and sustain the growth of large-scale software systems. However, it is widely believed that re-architecting is a huge manual task and is very hard to accomplish. A wrong re-architecting or a poorly executed one can have a devastating effect on the quality of a large-scale software system. Avoiding it temporarily by applying superficial and rapid solutions, only makes it more difficult in the long run. As a result, large-scale software (and our society that increasingly depends on it) will continue to be burdened with incorrect, inefficient, incomprehensible, and insecure designs [1].
* The broad re-architecting process can benefit from a range of techniques and tools developed to support certain tasks related to the software evolution and maintenance. Areas include but not limited to: program comprehension, change impact analysis, cost-benefit analysis, copy-and-paste code reuse, and automated software testing. [2]

[1] Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L. Nord, and Ian Gorton. Measure it? manage it? ignore it? software practitioners and technical debt. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pages 50–60. ACM, 2015.

[2] Moazzen, E. (2018). Identifying the Problems of Software Re-architecting and a Knowledge Representation Framework to Address Them (Unpublished doctoral thesis). University of Calgary, Calgary, AB. doi:10.11575/PRISM/32050

Part 3

We are considering what benefits software re-architecting can bring to us?

Software re-architecting mainly solves the problem of complexity.

* The complexity of the program logic. Avoid the emergence of branches as much as possible.
* The complexity of architecture design. Some common modules or subsystems, may be invoked by all other modules or systems.
* The complexity of system deployment. We can use a batch file to simplify the deployment steps.
* The complexity of technology. The simpler the development, the more powerful the function of the technology. New technology can improve the development efficiency, and enhance the maintainability of the entire project.