Name: Tongwei Zhang T1-OP4 ID: 40044711

1. Code-and-Fix Model – Without requirements analysis and hard to be changed later

Code-and-fix mode was the simplest, earliest and the most frequently used software development process. Usually, there was little or no plan in the initial phase. The programmer just needed to write the code when they had an idea and fixed the code when they encountered problems. The code-and-fix model only contained two steps: 1. Write some code 2. Fix the problems in the code. The advantage was the user can code right away and saw the result immediately. However, Its shortcomings also brought serious problems. Because it lacked a clear definition of the requirements document. If the major architecture problems were found in the late process. The programmer usually had to rewrite most parts of the application. It made changes is harder and more expensive[1]. On the other hand, after a number of fixed, the code became so poorly structured that subsequent fixes were very expensive. The code was expensive to fix because of poor preparation for testing and modification. So it was not suitable for the larger and more complex software application and we needed concrete phases in software development and test/evolution plans obviously.

2. Benington's Program Production Life Cycle – Lots of specifications

In the 1950s, the complicated software application had appeared, like Semi-Automatic Ground Environment. The traditional software development approach could not support the life cycle management of the complex application. In order to solve the problems which large software applications development brought about. Herbert D. Benington presented a software development life cycle: Program Production Life Cycle. It was the first explicit representation of a software development life cycle and defined main phases which were quite similar to later models such as specification and assembly testing[2]. However, this model was derived from hardware engineering, and there was not a mature industrial software development process at that time. From the diagram of this model, we noticed that there was not any feedback loop in this model. So, it could bring about high maintenance cost and the over-specific planning processes did not suitable for efficient development and fast-changing market.

3. Royces Waterfall Model – Insufficient feedback loops

Winston W. Royce presented one of the famous software development life cycle model, waterfall model. The waterfall model was based on classical engineering practice and driven by documents. It mainly had five steps: specification, design, implementation, testing, maintenance. Each step of the process yielded documents. One phase should move to the next phase only when its preceding phase was completed and perfected[3]. The problems of waterfall including 1. There was no jumping back and forth or overlap between them. 2. Before coding starts, there must be a set of design documents. 3. Although people could estimate the cost by adding the estimated costs of each phase and then added a safety factor, estimation was often wrong since we may not have enough information during the early phases to make accurate predictions about the effort needed. Many software projects failed because of cost overruns. 4. It also required a lot of rework if late-changing happened. So, the waterfall was not suitable for the large-scale and main-stream software applications development except some embedded systems which had a rigid and formal process.

Reference

[1] http://zone.ni.com/reference/en-XX/help/371361R-01/lvdevconcepts/lifecycle_models/
[2] R. Kneuper, "Sixty Years of Software Development Life Cycle Models," IEEE Annals of the History of Computing, vol. 39, (3), pp. 41-54, 2017. . DOI: 10.1109/MAHC.2017.3481346.7
[3] 20. W.W. Royce, Managing the Development of Large Software Systems, Proc., IEEE Wescon, pp. 19, 1970.