ABSTRACT

There are two ways of constructing a software design:

One way is to make it so simple that there are obviously no deficiencies,

and the other way is to make it so complicated that there are no obvious deficiencies.

- C. A. R. Hoare

Software Projects are now critical elements in many businesses, but all too often the work is late, over budget, or of poor quality. [19, 20] We need to come up with the new improved development strategies to overcome the poor quality design issues, which will take care of the first two problems of schedule and budget. [6, 7]

We can be highly productive when we write very small programs. Our productivity, however, falls off sharply when we develop larger programs. [1, 14] While some of this is due to the added tasks that come with larger projects, a significant part of the problem is caused by defects. As we write larger programs, the difficulty of finding and fixing problems increases exponentially. [16, 18] If we allowed to wrote consistently very-high quality small programs, we would not only produce better products but we also could substantially improve our own and our organization’s productivity. [13] Unfortunately because of various constraints and the work environment such kind of freedom is very rare and developer needs to work on large scale projects directly. [1] Even the best software engineers make many errors, and some of the defects they introduce can be unbelievably hard to find. Such personal difficulties, while inconvenient for individual engineers, can be a severe problem for the engineers’ organizations. Software design quality starts with the individual engineer; the way to review and verify the work done. [20] If even our smallest project programs are not of the highest quality, they will be hard to test, take time to integrate into larger systems, and be cumbersome to use. Then just we can imagine about the situation in case of large scale projects.

As we work to improve the quality of the software we develop, we should focus the ability of our process to produce quality products. In case of large scale projects seek the most effective development strategies with proper process model to find defects as well as the most effective
ways to prevent them. [12] It will definitely improve the quality of the design. [6, 7] Also recognize that the costs of finding and fixing defects escalates rapidly the longer we leave them in the product [4]. The suggested development strategy is thus to ensure our program is of the highest quality design with cyclic reviews and verification approach when we first produce it.

One potential problem with all the strategies when apply to the large scale projects is that some critical function may not be developed until the later development cycles. [14, 16, 18] This in turn has the double problem of delaying exposure to the principal project risks and of testing the most complex element for the first time in the full system context. If the rest of the system has many defects, this could greatly complicate testing of the later slices. Here we are suggesting the improvement in the strategy that tests the most difficult components with the smallest amount of untested code. The defects are then more easily identified and fixed which enhance the software engineering performance.

There is no one best strategy, so need to improve as per the requirement of the large projects design reviews and verification processes. We must examine the system structure, assess the principal development risks, and select a strategy to fit our situation. Our goals are to have the strategy naturally fits the system structure and to expose the principal risks as soon as possible. Here the high quality design is achieved by dealing with process measurement, process analysis, benchmarking, yield management, and defect management.

By doing design and code reviews, in the cyclic manner suggested in the strategy, we will see more improvement in the quality and productivity of our large scale development. On larger projects, it will pay us to review our requirements, our specifications, our designs, and our code. This strategy focuses on managing the defects in the software we produce. By improving our defect management, we will produce more consistently reliable components. These components, in turn, can then be combined into progressively higher quality large scale projects. [13, 16]