CHAPTER 9:

CONCLUSION AND FUTURE SCOPE

A high quality software design transforms an ill-defined requirement into an implementable product design specification. The quality of this design is important especially in the large scale projects, because it is nearly impossible to produce a high quality implementation of large scale programs from a poor quality design. The primary objective of this work is to improve the development strategy by defect injection and defect removal with high quality design by considering the clarity of the information of large scale project and the customer constraints or priorities. This large scale project development strategy defines cycle contents as bite-sized elements that are separately designed and implemented in development cycle. Each cycle is progressively unit tested and integrated, and, at the end, we have the integrated, complete program ready for system integration or system test. By progressively integrating each cycle, we avoid the need for special drivers.

With experimentation and the values of phase_yield, design_review_yield, code_review_yield, compile_yield and process yield as measures, we have proved that this work improves review quality of the development strategy and performance of compile and test time. The results proved that percentage of total defects found are increased by building checklists which consequence in low COQ and process benchmarking increased the A/FR as a measure of high quality design for large scale projects. It is proved that proposed development strategy, gradually scaling up personal software process with defined baseline processes and recording logs, suitable for even very large projects that involved multiple teams controlled and directed by a central project management.
Further work can be carried out to reduce the limitations of this thesis work. In future the focus will be to achieve all of the engineering project goals and objectives while honoring the preconceived project constraints during the improvement process for high quality design. Typical constraints like scope, time, and budget can be optimal by optimized Resource Allocation, effective Risk Management and wherever possible with Agile practices.

**Contribution & Conclusion: Achievements**

- Compile and Test Time improvements
  - Maximum from 50 to 30
  - Average from 30 to 13 and
  - Minimum from 10 to 4

- Yield Improvement
  - Design Review, Code from 75% to 100%
  - Code Review from 80% to 100%
  - Compile from 58.55% to 66.7%
  - Test 48.55 to 57.1%
  - Post Development from 41.45 to 50%

- Test Defects per KLOC
  - Reduced from 41.2 to 15 in an Average

- The principal lesson of this study is that we have to measure our own work before we will believe how much more effective reviewing is than testing.