CHAPTER VII
Summary & Future Research Directions

7.1 Summary

The “one size fits all” approach to applying SDLC methodologies is no longer appropriate [139]. Each SDLC methodology is only effective under specific conditions. Traditional SDLC methodologies are often regarded as the proper and disciplined approach to the analysis and design of software applications [140]. Examples include the code and fix, waterfall, staged and phased development, transformational, spiral, and iterative models. Lightweight methodologies on the other hand are a compromise between no process and too much process. These new methods were developed to efficiently manage software projects subjected to short timelines and excessive uncertainty and change. Nine types of lightweight SDLCs are Adaptive Software Development (ASD), Agile Software Process (ASP), Crystal, Dynamic System Development Method (DSDM), Extreme Programming (XP), Feature Driven Development (FDD), Rational Unified Process (RUP), SCRUM, and Whitewater Interactive System Development with Object Models (Wisdom).

Strengths of these new light methodologies include their simpler processes and easier acceptance by developers who are only familiar with code and fix techniques. In addition, these lightweight SDLCs aid developers in thinking clearly about the end products they are creating. Disadvantages include their inability to handle large development teams. Lightweight methodologies are most appropriate when there are uncertain and volatile requirements, responsible and motivated developers, and customers who wish to become involved. On the other hand, lightweight methods are inappropriate with teams of more than fifty and/or the project has a fixed scope.
Lightweight methodologies re-examine the traditional assumptions that have historically been made about the commitment of resources to requirements analysis and process improvement [141]. Traditional SDLCs operate on the fundamental assumption that it is worth investing resources to identify a flaw in a process because the process will be used over and over again. On the other hand, lightweight SDLCs recognize that when everything is changing and there is no assurance that processes will be reused that it makes little sense to expend the effort.

The purpose of implementing e-Governance is to provide good governance. As all e-Governance Applications have certain common constraints, it was the need of software professionals from long time for the development of a specific SDLC Model for e-Governance, which will meet all the constraints & Challenges. Our SDLC Model addresses all those challenges. Using this Model rapid development lifecycle, fund optimization and utilization, interoperability in all e-Governance Applications can be achieved successfully. This model provides a framework to develop reliable, robust, extensible and pluggable e-Governance application. In addition, the developmental cost, and implementation costs can be lowered as well as services can be reached at the doorstep of the citizen providing more accessibility.
7.2 Future Research Directions

This Model is independent of any engineering practice and can be adopted in any organization both in a new project and an existing project. Its salient features, including Iterative development, high level accountability, small teams, Regular builds, Configuration management, and frequent monitoring, all help in developing a robust system. Project management is becoming a hot area for research today on methodology, process and Meta process but one logic fit all concepts can’t work in this dynamic world.

In e-Governance Application, each parameter responsible to affect the project, changes so frequently that, it became a necessity to adopt a specifically designed Framework or Model for successful e-Government roll out. This research Model is open for customization to suit individual e-Governance Applications.