ABSTRACT

Over the last decade, software is evolving at a rapid pace along with more advanced technologies. Developing complex software systems is not a straightforward process but it includes implementation of high level concepts and techniques. Developing complex software systems became easy. However, the progress in software engineering concepts did not keep track with increasing complexity of modern software systems. It is difficult to meet current and future needs in software development using today’s most popular programming paradigm such as object-oriented programming.

The two fundamental principles that drive the research from the early days of software engineering to deal software complexity are separation of concerns and modularity. A lot of techniques exist in literature which follows these fundamental principles of software engineering. Some success in this direction has been achieved. But, still complete separation of concerns is not achieved even in today’s most popular programming paradigm like object-oriented programming. In these traditional techniques, some concerns may be easily encapsulated with their building blocks such as classes, modules, procedures etc. But, same is not possible for another. They are non-modular and spanning over multiple classes, modules, or procedures in a software system and are therefore called crosscutting concerns. Typical examples are persistence, logging, exception handling, synchronization, auditing, security etc. Due to their vary nature, crosscutting concerns are responsible for scattering and tangling.

Aspect-Oriented Software Development is another step towards achieving improved modularity and advanced separation of concerns. It aims at alleviating the problems of scattering and tangling by addressing crosscutting concerns. Crosscutting concerns are encapsulated in separate modules, known as aspects, so that localization can be promoted. It later uses composition mechanism to weave them with other core modules at loading time, compilation time, or run-time. This results in better support for modularization hence reducing development, maintenance and evolution costs.

AOSD was first introduced at programming level, where aspects are handled in code. Many aspect-oriented programming approaches have been proposed such as AspectJ, AspectC, AspectC++, JBoss AOP, JAsCo, HyperJ, adaptive programming, and composition filters. A
lot of work also has been carried out at the design level mainly through extensions to the UML meta-model. Research on the use of aspects at the requirements engineering stage is still immature. Handling crosscutting concerns in the early stages of software development is beneficial rather than handling them in later stages of software development because it not only makes the design simpler, but also helps to reduce the cost and defects that occur in the later stages of development. Aspect-Oriented Requirements Engineering (AORE) focuses on identifying, analysing, specifying, verifying, and managing the crosscutting concerns at the early stages of software development. It does not replace but rather complements any of the existing requirements methodologies.

Over the last few years, several research efforts have been devoted for handling crosscutting concerns at the early phases of software development especially at requirements level. These efforts are meaningless unless all the crosscutting concerns are properly identified. Many approaches only consider non-functional concerns as crosscutting concerns. However, crosscutting concerns may also be functional, such as auditing, or validation. In last few years, many researchers contributed their significant work in this area for resolving conflicts but most of them have used simple reasoning techniques based on intuitive that do not allow a rigorous engineering approach to the problem.

Four major contributions have been made in this thesis. The first contribution made is cognition and analysis of AORE literature. This is achieved by presenting an extensive AORE literature which helped to set the context and the background for this research work. The second contribution is development of an integrated AORE process model which supports the identification of concerns, specification of concerns, identification of both the functional and non-functional crosscutting concerns, and composition of concerns. The third contribution is made by proposing a fuzzy interval based approach to handle conflicts in AORE, which is a new of its kind and that will incorporate one domain in other. The forth contribution is establishment of comparison criteria to evaluate proposed approach against other approaches existing in literature.