Name of the Scholar: Zeba Khanam
Name of the Supervisor: Dr S.A.M Rizvi
Department : Department of Computer Science,Jamia Millia Islamia
Title : Legacy System Migration

ABSTRACT

Keywords: *Legacy Software, Refactoring, Aspect oriented programming, Reengineering, Software Maintenance, Software Modularity, Cohesion and Coupling.*

Legacy systems are important for the operation of organizations, holding incredible detailed business rules forming the backbone of the information flow of organization that consolidates information about its business. The amount of burden the legacy information systems are currently dealing with are subsequently posing numerous and important problems to their host organizations. There are number of factors depending on the type of organization that result in systems which are often difficult to understand and expensive to maintain. The demands of the market are constantly changing. Even if a new software product fulfills the business requirements at the time of release then it will hardly maintain this position for a very long time. A careful analysis of the scenario identifies a number of key drivers for having the system migrate and evolve. There are number of factors that contribute to the degradation of software such as adding new functionality in a short time period, the patch work done to fix the error, unavailability of proper documentation and the fact that software maintainers are replaced by people unfamiliar with the original software. Since the software requirements change, legacy systems must be evolved accordingly. Different approaches such as wrapping, migration and redevelopment are existing and are used to maintain legacy information systems depending upon the type of the legacy system and the type of maintenance required. Unfortunately, these approaches have not explicitly considered the concerns that are difficult to capture in single components, and tend to *crosscut* many components. Examples of such crosscutting concerns include distribution, synchronization, persistence, security, logging and real-time behavior. The concepts of reengineering and

refactoring have been gaining increasing attention during the last 10-15 years. The Reengineering process almost always begins with refactoring. A spaghetti or a badly written code is impossible to reengineer without refactoring it first because it is only after that initial step that the code somewhat comes out of the mess and conceptual design of the system could be analyzed. Today refactoring is a necessary process that improves the design, readability and maintainability of a system.

Aspect Oriented Programming (AOP) is a programming paradigm that offers a novel modularization unit for the crosscutting concerns. Functionalities originally spread across several modules and tangled with each other can be factored out into a single, separate unit, called an aspect. Although AOP was originally proposed for the development of new software, systems written using traditional modularization techniques may also benefit from the adoption of the more versatile decomposition offered by AOP, in terms of code understandability and evolvability. Many code quality problems are caused by idiomatic implementation of crosscutting concerns. Improving the implementation of legacy software systems may help to consolidate their value, and reduce their cost. These so-called *crosscutting concerns* result in programs that are difficult to understand and reason about, to divide into manageable pieces, to reuse, and to evolve.

The goal of this thesis is to investigate manual and automated techniques that can be used to support the migration of existing procedural programming code to AOP. To migrate an application to the new paradigm, identification of the crosscutting concerns is major task and is required (aspect mining). Then refactoring is applied to transform the scattered concerns into aspects. The major contribution of this thesis lies in exploring the refactoring opportunities in procedural software using aspect oriented programming; this would lead the benefits of AOSD to be extended to imperative programming that would contribute in solving the problems related to software evolution. The procedural language is C and aspect oriented language is Aspect C.