Devi Ahilya University, Indore, India **Institute of Engineering & Technology**				**II Year M.E. (Computer Engineering Sp. in Software Engineering)** **(Part Time)**			
Subject Code & Name	**Instructions Hours per Week**			**Credits**			
SEP4E5 Aspect Oriented Software Engineering	**L**	**T**	**P**	**L**	**T**	**P**	**Total**
	3	**1**	**2**	**3**	**1**	**1**	**5**
Duration of Theory Paper: 3 Hours							

Learning Objectives:

The objective of this course is to master basics of aspect-oriented software development, which enables a higher degree of the separation of concerns through crosscutting concern modularization. The course provides an overview of aspect-oriented approaches to software development throughout all of its stages, as well as programming languages connected with these approaches. The course also covers the relationship of aspect-oriented software development and software product lines. Students will gain experience with AspectJ, which is the most important aspect-oriented programming language of today.

Pre-requisite:

Familiarity with Object oriented programming, Object oriented design, UML is essential.

COURSE CONTENTS

Unit 1: Introduction to AOSD

This module provides a broad overview of aspect orientation. It introduces students to the aspect-oriented paradigm's origins and foundations, providing a solid basis and a common terminology to be used in subsequent modules. The fundamental concepts include all major elements of the paradigm: separation of concerns, crosscutting concerns, modularization, aspects, join points, point cuts, advice, and aspectual composition. Module 1's only prerequisite is knowledge of software engineering in some existing, well known paradigm.

Unit 2: Aspect-oriented analysis and design

This module covers a broad spectrum of software development activities, from initial requirements definition to architecture derivation and detailed design production. Each of these life-cycle stages can be realized using various aspect-oriented approaches. This module underlines the problems of tangling and scattering caused by crosscutting concerns in non aspect-oriented analysis and design approaches. It also presents aspect-oriented approaches for aspect identification, modularization, and composition, using several case studies for illustration. An in-depth experience with a particular analysis and design technique and its related tools is a final important goal of the module. Students achieve hands-on experience of aspect-oriented analysis and design through exercises. The prerequisites are Module 1 and familiarity with some requirements engineering, architecture, and design approaches. Knowledge of object-oriented (OO) analysis and design techniques (including UML) is desirable.

Unit 3: Aspect-oriented programming

Several AOP languages exist today, and most are extensions of existing languages. This module focuses on hands-on experience, giving special care to programming practices in AOP. The module covers various aspect languages, highlighting their differences and commonalities to teach students to abstract from concrete languages and understand aspect orientation's essential mechanisms. It also touches on implementing aspect language execution models to help students better understand the impact of aspects on program execution (for example, in terms of performance).The prerequisites are Module 1 and experiencein or knowledge about software implementation by means of contemporary languages, preferably in the OO paradigm.

Unit 4: Formal foundations of AOSD

This module extends or reevaluates formal notions, such as semantics, specification, and verification, in an aspect-oriented context. It surveys several semantic approaches, concentrating on one or two of them. It also covers specification of aspects, so that analysis of their desired properties becomes possible. The module presents formal methods for verifying and refining aspect systems, extending classical model checking to aspects, and relating static analysis to classes of temporal properties. The module uses these formal approaches to

■ define and compare declaring and weaving aspects,

■ specify the properties an aspect adds and determine whether these are true when the aspect is woven to a system, and

■ show that composing an aspect doesn't disturb the system's desirable properties.

The module also surveys existing work and defining and analyzing interactions among multiple aspects. The prerequisites include Module 1 and notions of computer science's formal foundations, especially logic and automata theory. Some instantiations might also require Module 3.

Unit 5: Aspect-oriented applications

Module 5 illustrates the practical use of various aspect-oriented technologies, such as programming languages, aspect-oriented analysis and design, and more generally, any software engineering methodology that embraces aspects. It presents case studies of applications that benefit from AOSD, covering system level elements (such as middleware) and end-to-end user applications (such as e-banking or e-government applications). The module's main subject isn't technologies that create system-level elements and end-to-end user applications; Modules 2 and 3 will have covered these. The prerequisites are Module 1 and, depending on the instantiation, Module 2 and/or Module 3.

Unit 6: AOSD and other paradigms

Aspects are always used in a context. Therefore, to develop applications using aspect-oriented techniques, it's important to relate AOSD to the development paradigms, methodologies, and programming languages that you used to implement the underlying base application. AOSD's context is almost always class-based object orientation. However, as AOSD spreads to other contexts, this relationship will diversify and become more important. Also, other advanced development paradigms have been developed that can be related to AOSD, either because they're complementary or because they target the same problems as AOSD (albeit differently). This module provides insights into the relationship between AOSD and these other advanced development paradigms (for example, development methodologies other than the OO paradigm), different general-purpose programming languages for the base code, and component-oriented software engineering. The prerequisites are Module 1 and at least one other module.

Text Books:

1. Ivar Jacobson and Pan-Wei Ng. Aspect-Oriented Software Development with Use Cases. Addison-Wesley, 2004.
2. RamnivasLaddad. AspectJ in Action: Enterprise AOP with Spring Applications. Second edition, Manning, 2009.
3. Mastering AspectJ: Aspect-Oriented Programming in JavaBy Joseph D. Gradecki, Nicholas Lesiecki, Wiley 2003

Reference Books:

1. Robert E. Filman et al. Aspect-Oriented Software Development. Addison-Wesley, 2004.
2. Siobhan Clarke and Elisa Baniassad. Aspect-Oriented Analysis and Design: The Theme Approach. Addison-Wesley, 2005.
3. Aspect-oriented Programming with AspectJ. Ivan Kiselev, Sams, 2003

Learning Outcomes:

The students after completion of the course shall be having the knowledge of aspect-oriented software development, which enables a higher degree of the separation of concerns through crosscutting concern modularization. Students will be able to build solutions with AspectJ on completion of the course.