Qualitative Analysis and Evaluation of AO Approaches

Namrata Sharma, Prerna Tyagi, Vaibhav Vyas

[1]*M.Tech(C.S) student, AIM & ACT Banasthali University, Rajasthan, India*
[1]*nsbhardwaj11154@gmail.com*
[2]*M.Tech(IT) student , AIM & ACT Banasthali University, Rajasthan, India*
[2]*prernatyagi022@gmail.com*
[3]*Asst. Prof., AIM & ACT Banasthali University, Rajasthan, India*
[3]vaibhavvyas4u@gmail.com

Abstract *— Aspect Orientation is key approach for handling the crosscutting concern in an appropriate manner by clearly distinguishing between core concerns and crosscutting concerns. Crosscutting concerns that depict functionality, which cuts across all the modules of the software system, were not addressed by traditional methodology. An aspect is the new component of modularization that allows crosscutting concern separation from other concerns. Crosscutting concerns which is known as aspect concerns generate from requirements which are non-functional such as security, persistence and logging. Initially, Aspect orientation developed at programming level but now it expands to the development phase. Aspect oriented programming currently emerge as well-accepted programming methodology and in most of the modern language show implementation of AOP. In the java language, the implementation of AspectJ has wide range of community acceptance. AspectJ developers use basic AOP rules or guidelines to add innovative and new features to software.*

There is several proposed Aspect oriented modelling approaches having different concepts or notations and demonstrate different stages of maturity. There is a need to identifying and specifying differences and commonalities of currently existing AOM approaches. This paper illustrates seven popular approaches based on evaluation criteria. Each approach is specified on the basis of evaluation criteria in the tabular manner. Evaluation criteria is divided into four components to show the comparison of each presented approach and four approaches are discussed in detail. In short the paper presents the summarization of each approach with strength and limitations.

Keywords—*Aspect Oriented Modelling, Aspect Oriented Programming, AspectJ, HyperJ, ASoC.*

INTRODUCTION

With the advent of new technology and integration of the upcoming technology with our day-to-day life, the need of the software development is increasing in large proportion. So, software engineering is the branch of software development or we can say that it is one of the emerging branches in today world which develop systems that are not only large in size but also complex in functionality. As these systems are based on real world so they should be reliable, quick in response and ease of use. All these supplementary features associated with software system hence crosscut the modular structure of system. System becomes more complicated when some functionality crosscut many module of software so this becomes more and more challenging for developer.

Aspect-Oriented Software Development (AOSD) methodology provides a way to develop software in step by step process. So AOSD gives a systematic way to deal with the development of software which is most for identifying the specific concerns that are crosscutting concerns. These concerns come in every modules of software.

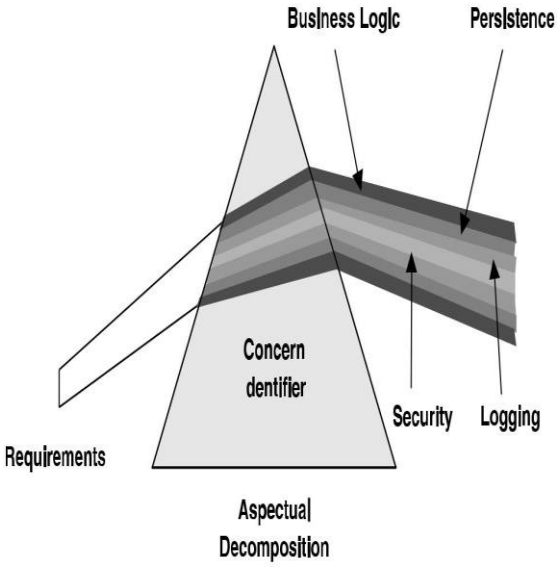

Figure 1.1 Prism analogies in which light passes through prism in the form of requirement after passing through prism, requirement separate out into distinct concern. This is the concern identification process where all the concerns are found out. Here concern identifier act as a prism which helps in aspectual decomposition of the requirement of the system.

AOSD implements new idea of modularization. AOSD represents the convergence of different ASoC Approaches, such as Composition Filters, Multi-Dimensional Separation of Concerns Adaptive Programming, Subject-Oriented Programming, and Aspect-Oriented Programming are used. AOSD, nevertheless, is a fairly young but rapidly advancing research field.

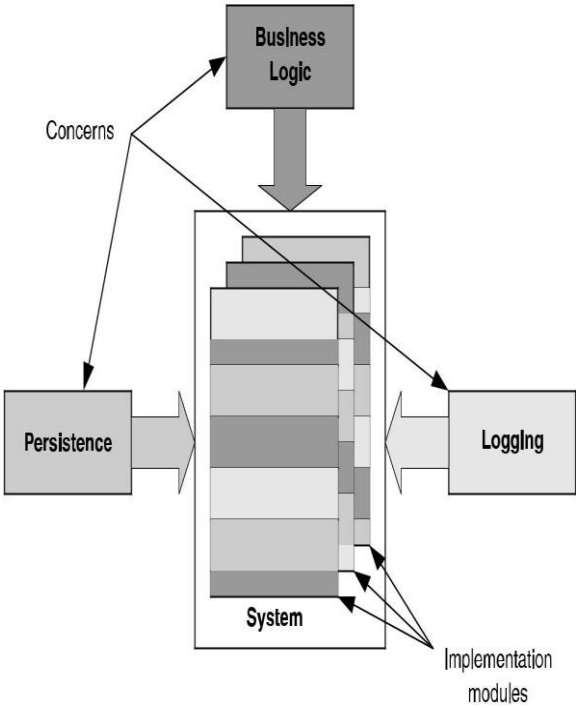

Figure 1.2 a system can act as composition of many concerns. Each implementation module addresses some element from each of the concerns the system needs to address

Aspect oriented modeling application is not only limited to the programming phase but also to the other area of the software development life cycle. These areas are requirement gathering, analysis phase and last the design phase. This software development is also encouraged by the emergence of Model-Driven engineering (MDE). MDE this develop models as the primary artefacts [8] in software development. Now, AOM is getting more attention.

Concern- It represents a specific consideration or requirement which is to be addressed for meeting the system goal. Concern known as crosscutting if it cannot depicted in language decomposition technique like methods and classes in OO approach. In AOSD methodology, this restriction is named as tyranny of the dominant decomposition [4]. The component of crosscutting concern is scattered on the other concerns or tangling with the other concerns of the specific system. The example of crosscutting concerns given by the logging module, which crosscut many module of system.

Concern Module-A concern is represented by one module or more than one module of the system. Kojarski et al. [6] reused the above term for the entire concern component that collectively represent a concern or a segment of concern. In formalized language like UML it shows depiction of concerns. There are some of the approaches present which highlights the concept of aspect for modularizing or some deals with crosscutting concerns whereas other unit of modularizing allows non-crosscutting concerns called as base concerns. This different category of concerns used to divide the aspect-oriented approach into two sub category known as asymmetric approach that allows aspect-base dichotomy and other one is symmetric that not allow this[7].

Composition plan- It is defined as the set of rules or collection of rules which is used for combining the concern modules Kojarski et al. [6] used the weaving plan for renaming in the favour of more general purpose term of composition which produces the combining of many modules as a whole. The execution of composition plan provides result in the form of composed model to represent the whole system. There are two types of phases present in it one is detection and other is composition. In the detection phase there is specification or identification of concern elements that are combining in the composed model and in other phase of composition allow actual combining of these concern elements. Also, there are two ways of composing the concern modules named as dynamic or static. In Dynamic, when module is executed that is during the run time concern modules are combined or integrate. Static composition allows concern modules are available at designing time analogous to compiling time.

Concern Composition Rule- Concern composition contain rules which are present in the composition plan and there is detail description given by one rule to show how these concern components are formed.

According to the composition mechanism, concern composition rule divide the most important concepts into subclasses.

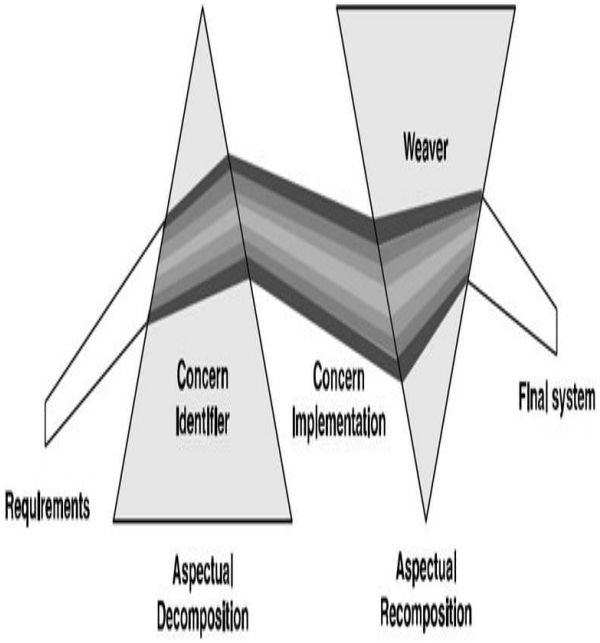

Figure 1.3 Here in first step requirements are broken down into distinct concerns with the help of concern identifier and each concern after passes through aspectual decomposition become independent and again concern are integrate with the help of weaver object to give final system.

System software provides the insight of set of concerns. For instance, banking system give the account of following concerns like interest computation, ATM transaction, account management, authorization for access to different services, customer management, statement generation and so on. Along with system concerns software needs to specify process concerns such as traceability, ease of evolution, comprehensibility and maintainability.

We have looked in this paper concerns can be two types in which one is crosscutting concern which give peripheral and system-level requirements which crosscut several modules. Other is core concern which gives central functionality to the module. An enterprise application has following crosscutting concerns like logging, performance, administration, storage management, resource pooling, authentication, security, transaction, error checking, multithread safety, data persistence. These are all concerns which are present in all the modules. For instance, concern like authorization crosscut all modules related with access control. Other concerns such as storage management crosscut all modules having object of stateful business. Figure 1.2 shows all these concerns of an application.

Figure 1.2 highlights implementation modules which give business and system-level concerns. This figure gives a view where system portrays as a composition of several concerns which are tangled with the existing implementation technique. So concern independency cannot be maintained.

Identify System Concern- In the identification of concerns the first step is the separation between core concern and crosscutting concern and then focus on the each concern separately and decrease the total complexity of implementation phase and design. Figure 1.1 shows how a light beam in the form of requirement pass through a prism of aspectual decomposition which helps in separation of concerns in the form of crosscutting and core concern which are needed to accomplish the requirements.

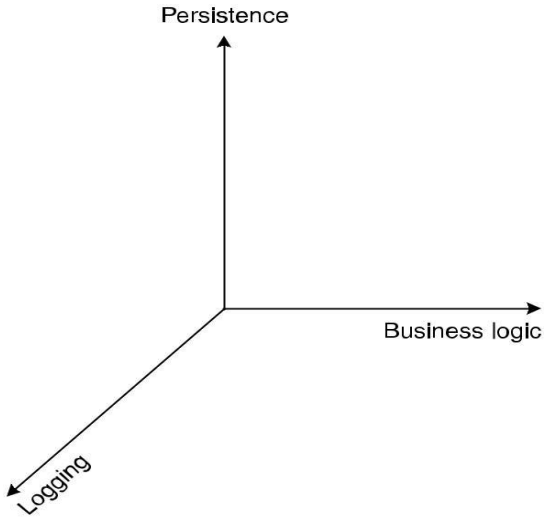

Figure1.4 it is a view which support multidimensional concern decomposition. Here each concern is mapped to its own axis to provide the orthogonality for concerns.

There is a other way of decomposing the concerns of the system and them project them in concern space which is N-dimensional and each concern forms dimension in the concern space. Figure 1.4 shows a 3-D concern space having business logic as core concern and logging and persistence as the crosscutting concern. The main idea behind this system view is to show that each concern is independent and emerge as without affecting the other concern. For instance, if there is a change in the persistence requirement in relational database to object database does not affect the security requirement or the business logic.

Specification and identification of concern is the important aspect of software development instead of focusing more on methodology. After that we can

address each concern separately and then make design task more manageable. But the real problem lies in the implementation of concern to the module. Although implantation helps in gaining independence of concern but this is not always the case.

A one dimensional solution – If we have a look on the nature of crosscutting concerns then these concerns tends to crosscut every module of the system and existing implementation technique allow mixing of each concern with the core concern. Figure 1.4 shows a 3-D concern space and the code which depicts concern is a constant flow of calls so it is in this sense is 1-D. Since the mapping of concern to implementation provide a mismatch.

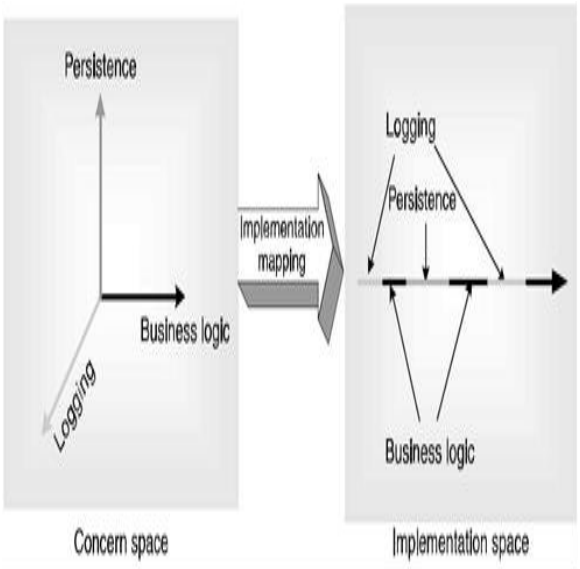

Figure 1.5 Mapping the N-dimensional concern space using a one-dimensional language. The orthogonality of concerns in the concern space is lost when it is mapped to one-dimensional implementation space

As implementation space is 1-D provide implementation of core concerns along with mix up of crosscutting concern. During design level, there is conversion of requirements to the each distinct concern but in the implementation phase each concern independency does not maintain by the existing programming methodologies.

Modularizing-The appropriate way to deal with complexity is to make it simple. In software designing, the complexity of complex system is reduced by first gathering the requirements and then separates them to concerns by modularizing them. However, object-oriented approach was developed for the modularization concept but it only develops for the core concerns of the system so it shows limitation to the crosscutting concerns. Due to this the concept of Aspect-Oriented Programming arises. In AOP methodology, the first step is to modularizes the crosscutting concern by specifying role and nature of each concern of software, implement the role of every concern in the module, having loosely coupling among modules.

There is loosely-coupling among the core modules through interface, but this is not true for crosscutting concerns. A concern implements on two sides: client side and server side. OOP helps in modularizing server part in interface and classes. However, if a concern is crosscutting type then the client request to server for service and server respond to all clients.

Let us take an example showing crosscutting concerns implementation in OOP. Through abstract interface an authorization module gives its service. Using the interface makes the coupling of implementation of interface and clients loosen. There are different clients available that use authorization service with help of interface makes the coupling of implementation of interface and clients loosen. There are different clients available that use authorization services with help of interface are most time unaware about the implementation it uses. If there is any change in implementation it uses does not require change to clients. One authorization replacement with another, means to instantiate correct implementation. The implementation of one authorization may be switched with other having no change or little for every client module. Each client having the code embedded for calling of API and the modules that has authorization service has all these call. These all are mixed with core coding.

Figure 1.4 shows the light beam passes through a prism where in the first step of aspectual decomposition the requirements are break down into several concerns. Now we add second step in which concern implementation along with aspectual recomposition are shown. After that there is a role of object known as weaver which makes the integration of concerns possible after they implements.

The most important change which the Aspect Oriented programing provides is the protection of mutual independency of each concern during their implementation. This implementation has mapping back to equivalent concerns. All this develop a system which is easy to understand or simple to implement and adaptable in different scenario.

Evaluation Criteria based on Language Component

Language Component

Language and Platform Dependencies

AspectJ and Unified Modelling Language form basis of the current Aspect Oriented Languages. The UML used for analysis and design of object-oriented software. Currently the combination of UML and OOP are widely used for software development in software engineering. Aspect J is the first language that implements Aspect oriented paradigm. Aspect J is the preferable language for developer.

Modelling structure behavioural elements

Aspect component can be represented by modeling them so that they show the structure crosscutting and behavioural crosscutting. These crosscutting depict by diagram and models offered by the language like Aspect J which is widely used for aspect oriented modelling language.

Comprehensiveness

This component measures richness of modeling language. "Richness" used here in the sense that all elements of aspect are analysed and design properly. Models and diagram are provided so that the evaluation of aspectual component must provide specification, depiction and designing. In short, this criterion covers all elements and constructs of modelling.

Analysing and Design Method

This criterion of language allows step by step process of first assessment of aspectual elements and then designing method which provide diagram and figures to represent aspectual component. Analysis and design permit certain rules and regulations for modelling non-aspectual and aspectual components.

Literature review

Aspect oriented Development software development with use case (AOSD/UC)

By this approach software development is done through the use of use case diagram. This use case is used to represent concerns. Use case is used in this to distinguish many concerns. It is a systematic process which allows distinct concern representation for developing software through software development life cycle. Although this approach gives software development in order way but it is much more associated with requirement and implementation phase. This approach allows substitution of external traceability among all phases with the help of explicit trace dependencies among models. There are several distinct UML diagram uses having use case in requirement phase or communication and class diagram in analysis phase during the entire software development process. In the design phase diagram such as component are refined to class diagram whereas sequence diagram helps in modeling behavioural features. In this approach there is language extension which shows dominance by AspectJ and HyperJ. Scalability feature is available which depicts modelling to high level elements. If we talk about this approach does not come with this idea of tool support.

It is a detailed approach which allows requirement analysis to high level design and then to low level follow by implementation at last which make it a complete design language.

Specification of aspect-use case in this is represented as package .inside these package is design available. Aspect is used to denote crosscutting. Aspect is depicted using two internal boxes. One depicts pointcut declaration and other class extension. Jacobson et al. approach has been currently described in three publications in details.

State charts and UML Profile (SUP)

This approach allows analysis and design of AO depends on state charts along with language as AOD that depends upon UML profile. In this AOD language quantify as UML profile that allows modeling of aspect and core structure in the form of class diagram. Behaviour is represented as use case, collaboration diagram, state charts and state machine. Aspects are represented as in class diagram form. The stereotype <<aspect>>is used to represent aspects. Also, state chart diagram is used to specify aspects in which subset of state show the connection over a sequence of events.

State charts are used to attain the behavioural crosscutting in which crosscutting which is behavioural figure as event that initiates state transition.

JAC Design Notation

Pawlak et al. [9] suggested JAC Design Notation to model a framework which is open source comprises complete IDE for aspectual elements with design support complemented to act as middleware layer. In JAC Notation it does not support internal traceability as models are not refined but allows external traceability which goes from design phase to implementation phase. As this approach depicts the aspectual concern represented by UML which is light-weight extension of it and there is no information available about the version of UML. Scalability property is not allowed by JAC notation.

Class diagram is used for representation purpose by this approach. For the designing procedure no information and rules are available for JAC Notation. It uses pointcut advice composition process. Here, class give representation on concerns like crosscutting. For elements the approach use is asymmetric and also provides support for composition asymmetry. Having concern composition rule helps in representation of symmetric approach using design notation. At modeling stage there is no availability of

composition but overdue up to implementation so no composed modelling is provided.
However, composition semantics provide by JAC framework. This approach does not provide conflict resolution along with models of interactions. There are five kind of stereotypes are available in UML meta-class operation.
The JAC notation has been defined and applied to many recognizes aspects such as authentication, caching, tracing, session in the reference of client-server application. These all provide the applicability of the JAC Notation to aspect in common. The JAC framework, along with the IDE available for modeling of aspect element. The aspect and base classes are designed using tool support. Code generation is provided by IDE for framework.

Klein Weaving Approach
Klein Weaving approach supports scenario language like Message Sequence Charts (MSC) which is normalized by ITU. UML Diagram version 2.0 is based on MSCs so this approach allows sequence diagram for analysis and there is no extension of UML available in this regards. A "Weaving Algorithm" designed by Klein for showing the composing behaviour means a scenario is designed for modeling with UML diagram in sequence.
This approach is neither implemented at the other stages of software development nor does it allow modeling of sequence diagram for refinement and no proper guidelines for designing phase. The efficiency constructs like scalability or traceability not supported by Klein approach. The pointcut advice composition is provided by this approach. Sequence diagram helps in giving modeling of all behaviour. The two scenario support by aspect having separate role in composition: the first role describes some part of the behaviour that is the advice replaced the pointcut. Each time when the behaviour is described by pointcut shown in semantics of base scenario then each time this replacement takes place. In this manner this approach follows asymmetry rule. This approach is symmetric because at one time it shows behaviour that served as pointcut or advice and at other time it may serve as base behaviour. There is a weaving algorithm having two phases which described composition semantics. In first phase, according to pattern identified in pointcut the join points are identified in base behaviour. In second phase, the base behaviour consists of behaviour in advice. As models are statically developed so implementation of the two phase algorithm is the work of future. An effect is not specified until the weaving algorithm is present. Some tasks such as handling of conflicts and specifying interaction are stated only not addressed as addressing is done in future.
The join points in this approach are sequence message in sequence diagram which match with sequence diagram define in pointcut.

Language Components Tables

Approach	**Modeling Language Dependency**	**Programming Language Dependency**
AODM	UML For modeling and assessment UML used.	Yes, AspectJ Earlier, development of AODM is done to give notation that is depending on AspectJ. Presently, adaptive programming support as AODM makes it generic.
Theme/UML	UML For design and assessment UML supported.	Yes Programming like support subject oriented. Later, HyperJ, composition filters, AspectJ is used by it.
AOSD/UC	UML For modeling UML used	Yes It uses Standards and notations of HyperJ and AspectJ
AAM	UML UML 2.0 is used for modeling.	No No platform dependency
JAC Design Notations	UML For modeling UML used	Yes JAC based Framework used.
Klein's Weaving	MSC Message Sequence Charts used for modeling	No Platform independent
SUP Approach	No Modeling language is not supported.	No Platform independent

TABLE I: In this table language and platform dependencies are shown by all approaches. Mostly the language dependency on UML (Unified

Modeling language) and programming language like AspectJ and HyperJ.

Approach	Comprehensiveness
AODM	Mid-High Hardly, there is modelling of Components advices and pointcut so not much comprehensive.
Theme/UML	High Modeling of join point, advice and pointcut are not supported.
AOSD/UC	Mid-High Comprehensiveness is shown by aspect but aspectual construct modeling is not supported.
AAM	Mid-High Primarily, for architecture of aspect system AAM is developed but complete details of aspect or non-aspect concerns not available.
JAC Design Notations	Mid No design diagram of aspect component only class diagram available.
Klein's Weaving	Low In this approach internal aspectual elements are not supported but allow to model aspect.
SUP Approach	Mid Neither non aspectual nor aspectual elements are represented,

TABLE II: In this table, the comprehensiveness one of the language component is shown which gives the complete details of design and analysis for all approach describe in table. This property ensures that the models must give representation and specification along with design of aspectual components such as advice, pointcut, aspect, join points and also the composition. It measure whether all aspect component with non-aspect component show modeling by the language used or in some case no modeling is shown. Also keep an account whether best design notation is given.

Approach	Structural Crosscutting	Behavioural Crosscutting
AODM	Yes Sequence and class diagram depict aspect concerns.	Yes Collaboration diagram represents aspect concern.
Theme/UML	Yes Class and package diagram represent structural crosscutting.	Yes Sequence diagram represent behavioural concern.
AOSD/UC	Yes Structural crosscutting is represented.	Yes Behavioural crosscutting it is represented.
AAM	Yes Class diagram depicts structural concern.	Yes Communicatio n diagram depicts concern.
JAC Design Notations	Yes Class diagram depicts concern.	No Behavioural concerns are not supported.
Klein's Weaving	No Structural concern is not depicted.	Yes Sequence Diagram for behavioural concerns.
SUP Approach	Yes Class diagram shows crosscutting.	Yes Collaboration and state machine diagram is allowed.

TABLE III: In this table, evaluation of approaches done under the criteria which check models provided by language used and measures whether the language offered behavioral and structural components.

Approach	Design Process
AODM	No This approach does allow design part.

Theme/UML	Yes Design diagram allow but requirement analysis is also supported.
AOSD/UC	Yes Concerns are depicted using use case.
AAM	No No representation of concern but in some publication Design diagram allowed.
JAC Design Notations	No No depiction of concerns using diagram
Klein's Weaving	No It does not provide process for concern modelling.
SUP Approach	Yes Use a proper design process for modeling.

TABLE IV: In this table, first assessment or analysis of requirements and other constructs are carried out and after that designing take place.

CONCLUSION

In this paper we present seven Aspect oriented modeling approach. As the research area of AOM is new so it need time to be recognized and widely accepted. AOM contribute in the field of requirement gathering, analysis and design. Requirement Engineering ensures the proper handling of functional and non-functional requirement. Analysis allows identification of crosscutting concern from the core concerns. Design phase one of the important phase checks for the aspect concern to be modelled to architecture design. We have looked so far that in some cases that AO approaches developed on the potency of the non-approaches. Also, these approaches intended to address the earlier unnoticed issues such as modularization of crosscutting concerns.

We have looked each approach presented here based on set of common criteria that shows desirable properties of every approach software engineering like comprehensiveness, language and platform dependency, Analysis and design and modelling structure and behavioural elements. Through these approaches the unexplored issues come to light such as requirement-level concerns, conflict detection with the help of composition and architectural concerns.

Also there are some of challenges present such as at the programming level there is a need to integrate AOP with AspectJ so as to develop enterprise application such as real time system, desktop application, enterprise application and embed system.

References

[1] Ruzanna Chitchyan, Awais Rashid, Pete Sawyer Alessandro Garcia,"Survey of Aspect Oriented and Design Appraoch" AOSD-Europe-ULANC-9, May 2005.

[2] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, M. Wimmer and G. Kappel, "A Survey on Aspect- Oriented Modeling Approaches", 2011.

[3] Abid Mehmood, Dayang N.A Jawawi, "An exploratory study of the suitability of UML- based aspect modelling techniques with respect to their integration into Model-Driven Engineering context", 2013.

[4] Felipe Francesco P. L. da Costa, Jorge A. M. Gonc ̧alves , Paulo R. B. Prestes, Eduardo Guerra, Fabio Fagundes Silveira," AICC-UC - An Approach to Identify Crosscuting Concerns Based on Use Cases" .

[5] Peri L. Tarr, Harold L. Ossher, William H. Harrison, and Stanley M. Sutton, " N Degrees of Separation :Multi-Dimensional Separation of Concerns", May 1999.

[6] Sergei Kojarski and David H. Lorenz, " Modeling Aspect Mechanisms: A Top-Down Approach", May 2006.

[7] William H. Harrison, Harold L. Ossher, and Peri L. Tarr. "Asymmetrically vs. Symmetrically Organized Paradigms for Software Composition", December 2002.

[8] Wesley Coelho and Gail C. Murphy. "Presenting Crosscutting Structure with Active Models", March 2006.

[9] Renaud Pawlak, Laurence Duchien, Gerard Florin, Fabrice Legond-Aubry, Lionel Seinturier, and Laurent Martelli. "A UML Notation for Aspect-Oriented Software Design", March 2002.

[10] Ramnivas laddad, "Aspect in Action : Practical Aspect-Oriented Programming", 2003.

[11] Gerd Beneken, Frank Marschall and Andreas Rausch," A Model Framework Weaving Approach Position Paper", July 2005.

[12] Éric Tanter and Jacques Noye, "Motivation an Requirements for a Versatile AOP Kernel", 2004.