Analyzing the Comprehensibility of Aspect-Oriented Modelling and Design of Software System

Deepika Shukla
Asst. Professor (Computer Sc. And Eng. Dept)
Institute of Technology, Nirma University, Ahmedabad(India)

ABSTRACT
Implementing any big software system is a complex task. One of the major reasons for this is that, there one would like to modularize but for which the implementation would be spread out. Such concerns are more commonly known as Aspects. For example security aspect has to be taken care irrespective of the fact, whatever business logic is being implemented. These concerns cannot be modelled appropriately using traditional Object-oriented approach as these Aspects, manifest themselves as behaviours that are tangled and scattered across a system. Due to this fact, it affects the comprehension capabilities of modelling artefacts of the system also these issues lead to problems achieving traceability of aspects throughout the development lifecycle. Aspect-oriented Analysis and Design (AOAD) has been accepted as an alternative approach to tackle such concerns in an effective manner. This paper presents a comparative study of effectiveness of Aspect-oriented Analysis and Design versus Object-oriented Analysis and Design approach and analyses the results of both of these approaches on the comprehensibility of software systems' knowledge.

General Terms
Object Oriented, Modelling and design, Aspect, Analysis and Design

Keywords
Aspect Oriented, Aspect-Oriented UML, Object-Oriented Analysis and Design, Aspect Oriented Software Development. Unified Modelling Language

1. INTRODUCTION
Object Oriented Analysis and Design (OOAD) has been the first choice of system analysts and software engineers for modelling a software system for nearly last four decades. Also it is well accepted fact that for performing OOAD, the UML diagrams are considered as the industry standard. However, y the complexity of the current software systems is increasing exponentially. The complexity is not only in terms of key functionality of the systems but also certain other concerns which span the entire system architecture. Such concerns are also called cross-cutting concerns or Aspects. Examples might be locking in a distributed application, exception handling, or logging method calls. Object oriented paradigm ideally fails to address this additional complexity or aspects adequately, leading to tangling of code and scattering of concerns across the architecture. Due to the inappropriateness of Object oriented Analysis and Design approach, in handling the cross-cutting concerns (Aspects). Aspect oriented analysis and design approach has been proposed as a technique for modularizing cross-cutting concerns [4], and is finding increasing interest and popularity as an established modelling language among academicians, researchers and practitioners.

Software systems today are becoming more and more complex due to the inherent need like they should be web based, processing data in real-time, distributed where the underlying architecture remains heterogeneous. Also it is observed that the traditional approaches of software development focuses on decomposing the system into units of main business logic. They prove to be insufficient to handle secondary or supporting functions and concerns like logging, security, reliability, scalability etc. Such concerns cross-cut more than one module and core business functionality of the system. A new construct should be defined that takes care of cross-cutting aspects of a system. Not surprisingly, this new program construct is called an aspect, and the approach which modularizes the system in terms of these aspects in addition to core business logic is popularly called as Aspect oriented software development. Aspect-oriented software modelling and development can be considered as an extension to Object oriented analysis and modelling [1]. It is considered as the key to self-adaptive systems. This paper presents a comparative study of effectiveness of Aspect-oriented Analysis and Design versus Object-oriented Analysis and Design approach and analyses the result of both of these approaches on the comprehensibility of software system's knowledge.

The rest of this paper is organized as follows. Section 2 covers an introduction about the baseline paradigm of software development and understanding of terms related to it. Section 3 comprises of detailed survey of related literature whereas in Section 4 comparison of both the techniques is shown. The diagrams shown in the questionnaire is in industry standard UML. Section 5 results are presented. Section 6 finally sought conclusion and future scope of the field and probable research areas.

2. ASPECT ORIENTED SOFTWARE DEVELOPMENT
Aspect-oriented software development basically can be divided into two broad categories of studies.

- Aspect-Oriented Analysis and Design (AOAD)
- Aspect-Oriented Programming (AOP)

Where, AOAD can further be divided into Aspect-Oriented Requirements Engineering (AORE), Aspect-oriented Architecture (AOA) and Aspect-oriented Modeling (AODM). Whereas AOP focuses mainly on two areas; (i) Development of new programming languages and platforms, which are capable to program crosscutting concerns and (ii) Study of existing aspect-oriented languages. Fig 1, shows the categorization of Aspect-Oriented Software development. Majorly three types of constructs constitute any Aspect oriented modelling method. First, constructs are required for modelling base elements, second, there has to be constructs for modelling crosscutting elements, and third, the method

must include certain constructs for modelling crosscutting relationships. UML has been accepted as the de facto standard for object-oriented modelling and is well received as dominant language for specifying base elements of an aspect-oriented model in software community. Whereas for modelling cross-cutting concerns the standard UML lacks the building blocks and so many extensions to the existing form of UML have been proposed in the literature.

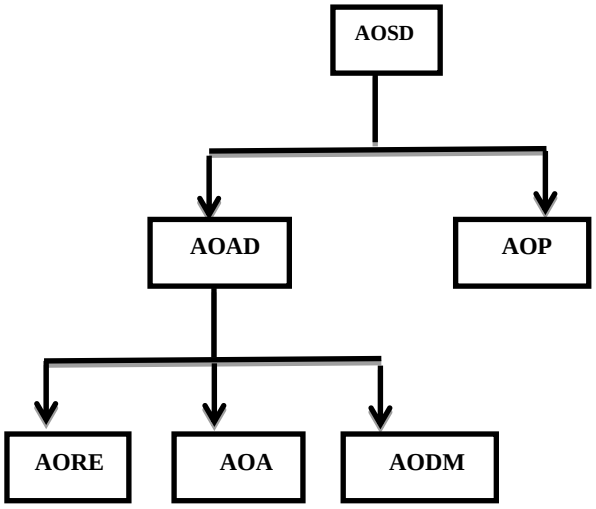

Figure 1: Categorization of Aspect-Oriented Software Development

2.1 Introduction to various terms related to Aspect-oriented paradigm.

- Join Points: These are points in the execution of the system where behaviour supplied by the aspect is combined. In code form they could be manifested as method calls. Join points are used to define the dynamic structure of a crosscutting concern.
- Pointcuts: They are set of join points and are used to specify at which join points crosscutting behaviour to be executed.
- Advice: These define code to be executed whenever a join pint of a particular set of join point is reached.
- Introductions: are used to crosscut the static type structure of the classes. i.e.; with introductions additional attributes and behaviours may be added to the class as if they were declared in the classes themselves. New generalization and realization relationships can also be inserted into the class structure thus changing the super-classes and super-interfaces of the classes.
- Aspects: "Modular units of cross cutting implementation" [4] and serve as a container for pointcuts, pieces of advice, introductions etc.

3. RELATED LITERATURE SURVEY

Aspect-Oriented software development is relatively new approach of software development but within very short span of time is gaining popularity among developers, researchers and practitioners. Most of the literature that exist on the topic can be found related following sub-areas.

There exist several survey papers which talks about Identification and separation of concerns at early stage. Basically they explain and advocate the need of identifying aspects in the Requirements Engineering phase of software development. For example in [5] [6] [7] [8] and [17] emphasize on Aspect oriented Requirements Engineering in general and for component based software systems. That is at requirements level. Other group of research papers talk about Aspect based modelling and design and in the effort they have developed a new approach. A wide literature is there which makes an effort towards comparing the standard object oriented paradigm with Aspect-oriented paradigm [3]. Lot of work is also available in which they justify the lacuna present in the current UML building block and prove that current form of UML is not appropriate for performing Aspect-Oriented Modelling and thus they suggest an extension to the existing UML notations for modeling the aspect in analysis and design phase[14]. Many authors have worked upon applying and adapting Aspect-oriented paradigm for a particular application domain. In [10] the concept of Aspect-Oriented is applied to Web application. Also in [11],[12], the use of Aspect-Oriented programming is done for Embedded system. Also in good number of papers current approaches in the use of UML diagrams to support aspect-oriented analysis and design are discussed and evaluated in terms of their ability to support the specification, change, maintenance, testing and reuse of aspects during requirements elicitation and throughout the software engineering life cycle. An approach based on UML activity diagrams is proposed, discussed and evaluated in [13], whereas the work presented in [14] has extended UML class diagram. The authors have used very generic pointcut and advice language which in turn has facilitated to model with aspects. As a result of particular approach, better separation of concerns as well as more redundancy reduction in UML class diagrams has been obtained thus making them more readable and understandable and in turn maintainable.

In [15] Sequence Diagram describes the Aspectual Sequence Model based on the Unified Modeling Language (UML) to identify the crosscutting concerns in the framework of Model Driven Architecture (MDA). An approach on modeling traversing features in concurrent software system was presented, which is based on aspect-oriented techniques and statecharts of unified modeling language is described in [16]. A use case driven approach is explained in [17] to explain the interaction analysis in Aspect-Oriented models. While reviewing this vast amount of literature, one can easily deduces that, most research on AOP modeling with UML is based on software forward engineering. Due to the fact modelling of aspects is done in early stage of software development process.

Research also is being performed in the area of Aspect oriented software development where authors have tried to understand the aspect from reverse engineering. For example in The area also has a related topic of research on Aspect Mining but little literature could be found on the same. In [18] & [19], it has been tried to understand, identify and remodularize the extracted aspects from the existing O-O system source codes for software reverse engineering. Also a complete area of research is being done for developing aspectoriented language and many researches are present in which the authors are working on AspectJ language which is more or less accepted as industry standard for implementation of Aspects. One thing in most of research papers, it is found and emphasized that Aspect-Oriented development approach is better than the traditional approaches of software development. Not always the case is same according to few authors and researchers. Few authors are very sceptic about benefits of Aspect-Orientation. For example in [21] authors are of the view that there are many misconceptions outlines some important aspect-oriented modeling issues, such as the modular nature of aspects, their resemblance with classes, and their high coupling with the base program. The author's .have mentioned that in most of the Aspect-oriented work where

AspectJ is used for the implementation, Aspects are considered as constructs analogous to classes which according to them, Aspects contradict the basic principles of object oriented paradigm. Whereas [22] presents a UML-based approach to justifying that an aspect-oriented program conforms to its expected crosscutting behavior or not. In this work aspect-oriented UML design models have been explored to derive tests for exercising interactions between aspects and classes.

4. EXPERIMENT AND OBSERVATIONS

Here, a brief planning and design of the experiments conducted to analyse the comprehensibility of AOAD (Aspect-Oriented analysis and Design) versus (OOAD) Object-Oriented Analysis and Design is provided. Based on the data that is collected from a series of controlled experiments.

4.1 Methodology

A questionnaire was prepared and was given to subjects. In addition to that, the subjects were shown/explained class diagrams of a system (Figure Editor) (i) based on OOAD using UML and (ii) based on AOSD using UML. The data was collected and then analyzed. Table-1 presents the questionnaire.

Table 1: Questionnaire used for taking the responses from the subjects

1. Your degree of awareness with Aspect Oriented software development.
   a) High b) moderate c) Low d) Not aware
2. After understanding both the models, which modelling paradigm would you prefer, for explaining system to others
   a) OO Model b) AO Model
3. Approximate time taken to understand the Object-oriented model for the system.
   _____min/hr._____
4. Approximate time taken to understand the Aspect-Oriented model of the system...
   _____min/hr._____
5. Aspect-oriented modelling using UML increases the understanding of system requirements and functions.
   a) Strongly agree b) agree c) disagree d) strongly disagree
6. The degree of class reusability would be more when AOSD is used
   a) Strongly agree b) agree c) disagree d) strongly disagree
7. AOAD reduces the interdependence of classes /aspects/requirements
   a) Strongly agree b) agree c) disagree d) strongly disagree
8. Current UML building blocks are sufficient to express cross-cutting concerns.
   a) Strongly agree b) agree c) disagree d) strongly disagree
9. It would be more convenient to do AOAD if current UML building blocks are extended.
   a) Strongly agree b) agree c) disagree d) strongly disagree
10. It would be more convenient to do AOAD if entirely new modelling language is developed for the same.
    a) Strongly agree b) agree c) disagree d) strongly disagree

5. RESULT ANALYSIS

The subjects were explained the Analysis and Design level class diagrams for a software system. Firstly, they were given exposure to class diagram based on OOAD using UML and secondly they were exposed to class diagram based on AOSD (Aspect-Oriented Software Development) using UML constructs. Table-2 shows the consolidated data obtained from the experiment. The results were compiled and raw figures were converted to percentage. Figure 2 is graphical view of the results obtained. The graph is created for Q1 and Q5 to Q10 where the responses were not quantitative. It can be observed from the above results that, as far as understanding of system requirements, class reusability and interdependence among classes/aspects/requirements, is concerned Aspect-oriented Analysis and Design score over Object-oriented Analysis and Design. But when the parameter is capability of UML to express cross-cutting concerns, from Q8 to Q9 items, the results clearly show that, there is a clear-cut requirement of complete framework which will ease the Aspect-oriented Modelling. The results of Q10 gave a surprising insight. A good number of the subjects were possessing apprehension towards change of modelling language. A separate research can be carried out for knowing the reason for such a scenario. However it is out of scope of this study.

Table 2: Results of the Questionnaire in %age

Item	High	Moderate	Low	Not Aware
Q1	50	30	10	10
Item	**First**	**Second**		
Q2	40	60		
Item	**Mean Time taken**			
Q3	3 min			
Q4	6 min			
Item	**Strongly Agree**	**Agree**	**Disagree**	**Strongly Disagree**
Q5	10	60	30	0
Q6	10	60	20	10
Q7	30	50	10	10
Q8	20	20	50	10
Q9	50	20	30	0
Q10	30	30	30	10

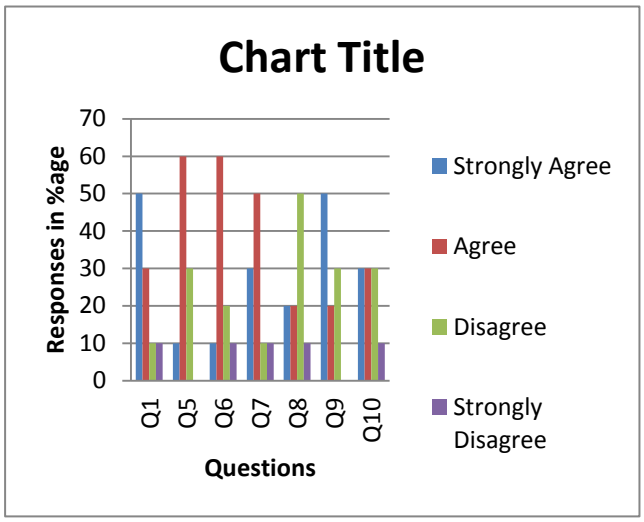

Figure 2: Graphical view of the results obtained from the experiment

6. CONCLUSION AND FUTURE RESEARCH AREAS

From the literature available on the subject and the small experiment that was performed it was found that the software development process can benefit from the Aspect-oriented Analysis and Design. Identifying the Aspects in the early stage of Software system development can give good returns. It was also found while conversing with the subjects and looking to the work profile of the subjects that, when system is to be explained to the end-user then OOAD artefacts would be more useful whereas for explaining the system to the development team AOAD(Aspect-oriented analysis and Design approach would be more useful and degree of comprehensibility increases. One of the major problems in AOAD is that it is still not mature enough and has gained popularity among practitioners especially academicians. Also it was felt during the experiment stage that, subjects were possessing low degree of awareness with the topic of the study presented in this paper, which also affected the result of the work. From the extensive literature survey, it can also be concluded that, most of the Aspect-oriented modeling is done using either standard UML or some kind of extension to standard UML. Also most of the work is related to one or the other UML diagram. That is either class diagram, Activity diagram etc. From this fact it can easily be said that, a complete framework of modeling notation is lacking for performing AOM (Aspect Oriented Modeling), which can be considered as research direction. As in this paper, only ten subjects were involved. Due to which the work and the results can be considered constrained and specific rather than generalized. In future the experiment set can be increased so that more generalized conclusion can be offered.

7. REFERENCES

[1] Gregor Kiczales, James Hugunin, Erik Hilsdale, Mik Kersten, Jeff Palm, Crista,Lopes, Bill Griswold, and Wes Isberg "ASPECT ORIENTED PROGRAMMING", Kiczales

[2] Silvia Abraha˜ o, Carmine Gravino, Emilio Insfran," Assessing the Effectiveness of Sequence Diagrams in the Comprehension of Functional Requirements: Results from a Family of Five Experiments", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 3, MARCH 2013

[3] I.Jacobson,Pan-Wei-Nq,"Aspect-Oriented software Development with Use Cases (Addison-Wesley Object Technology Series)",2004.

[4] Aws Magaablah,Zarina Shukur and Noorazean Mohd Ali," Systematic Review on Aspect-Oriented UML Modeling:A complete Aspectual UML Modeling Framework",Journal of Applied Sciences, 2012.

[5] John Grundy,"Aspect-oriented Requirements Engineering for Component-based Software Systems", Proceedings of RE'99, 7-11 June,

[6] Editorial Article," Aspect-Oriented and Component-Based Software Engineering", IEE Proc-softw..,Vol 148, No 3, June 2001,pp 87-88.

[7] Awais Rashid,Peter Sawyer,Ana Moreira,Joao Araujo, "Early Aspects: a Model for Aspect-Oriented Requirements Engineering", Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02), 2002,pp

[8] A.Rashid,A,Moriera and J.Araujo,"Modularisation and composition of aspectual requirements", Proceedings of the International conference on Aspect-Oriented software development,pp 11-20,2003

[9] www.wikipedia.org

[10] Matthias Niederhausen*, Zoltain Fiala, Norbert Kopcsek, Klaus Meissner,"Web Software Evolution by Aspect-oriented Adaptation Engineering", 1-4244-1450-4/07/007 IEEE

[11] Marco A. Wehrmeister, Carlos Eduardo Pereira, and Franz J. Rammig," Aspect-Oriented Model-Driven Engineering for Embedded Systems Applied to Automation Systems", IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 9, NO. 4, NOVEMBER 2013,pp 2373-2386.

[12] Leo Espinoza , Heldy Espinoza , Wenying Feng," Modeling a Facilities Management and Information System by UML", 2013 10th International Conference on Information Technology: New Generations,pp 66-70

[13] Nada Albunni ,and Miltos Petridis using UML for Modelling Cross-Cutting Concerns in Aspect Oriented Software Engineering".

[14] Gefei Zhang,"Towards Aspect-Oriented Class Diagrams", Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05) 0-7695-246",5-6/05,2005

[15] J.Zhang, Yuejuan Chen Guangyuan Liu, Hui Li, "Using Sequence Diagram to support Aspect-Oriented Programming in MDA" 2009 International Conference on Intelligent Human,pp 359-362.

[16] SU Yang QIN Jun,"Approach on Modeling Crosscutting Features in Concurrent System".

[17] Interaction Analysis in Aspect-Oriented Models Katharina Mehner Mattia Monga, Gabriele Taentzer, 14th IEEE International Requirements Engineering Conference (RE'06),0-7695-2555-5/06 2006.

[18] ZHANG Ping, SU Yang,"Understanding The Aspects From Various Perspectives in Aspects-Oriented Software Reverse Engineering", 2010 International Conference on Computer Application and System Modeling (ICCASM 2010).

[19] SU Yang, ZHOU Xuan-wu, ZHANG Min-qingP,"Approach on Aspect-Oriented Software Reverse Engineering at Requirements Level", 2008 International Conference on Computer Science and Software Engineering,pp 321-324.

[20] Bruce C. Hungerford, Member,Computer Society, Alan R. Hevner, Member and Rosann W. Collins, Member," Reviewing Software Diagrams: A Cognitive Study", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 2, FEBRUARY 2004, pp 82-96 .

[21] Iqbal, Saqib and Allen, Gary, "Aspect-Oriented Modelling: Issues and Misconceptions" University of Huddersfield Repository, http://eprints.hud.ac.uk/9007/

[22] Dianxiang Xu, Weifeng Xu, W. Eric Wong Testing Aspect-Oriented Programs with UML Design Models".