Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 12 (21): 2195-2201, 2012
ISSN 1812-5654 / DOI: 10.3923/jas.2012.2195.2201

Heavy-Weight and Light-weight UML Modelling Extensions of Aspect-Orientation in the Early Stage of Software Development

Aws A. Magableh, Zarina Shukur and Norazean Mohd Ali
School of Computer Science and Information Technology, Universiti Kebangsaan Malaysia,
43600 Bangi Selangor, Malaysia

Abstract: Aspect-Orientation and Object-Orientation complement each other in a number of aspects. Hence, it is imperative to investigate level of adopting Unified Modelling Language (UML) by the Aspect-Orientation. This study employed a systematic literature review to examine the approaches of Aspect-Oriented UML (AOUML). The rapid growth of complexities of systems, of late have eventually paved way for the emergence of new concerns. In fact these new concerns have cut-cross other concerns and core classes in the system by their nature. Therefore, it is crucial to focus on the concept of crosscutting concerns (Aspect), throughout the whole development life cycle, as they are accountable for generating, disseminating and interweaving depictions. The scope of this study is to depict and examine the current state of art of Aspect-Orientation modelling using UML. The UML diagrams have been implemented on the top of Object-Orientation concepts, it has not been meant to be used to model Aspect-Orientation. Thus, the motivation of this study is to propose a complete tailored formwork that represents Aspect's constructs using all UML diagrams based on AspectJ constructs. The objective of this present study is find out the shortenings, lack of support, advantage and disadvantage of the existing well-known approaches of Aspect Modelling based on a carefully selected, evaluation and compression criteria. The examination and analysis have revealed that there are some deficiencies of Aspect representation in the early stage of software development, while using the existing UML. We have concluded that extensive research has to be carried out, for us to get a complete comprehensive framework modelling approach that covers all UML diagrams, rather than just a few, moreover we suggest that the assumption of modelling extensions have to depend on a reliable base.

Key words: Aspect-oriented modelling, crosscutting concerns, aspect representations, aspect-oriented UML, aspectJ, aspect-oriented programming, aspect-oriented software development

INTRODUCTION

Software reusability is a portion of code that can be reused with other systems with minor or no modification. This indicates building a system, by integrating existing pre-engineered components. Reusable modules and classes reduce implementation time board (Zakaria *et al*., 2002). OO (Object-Orientation) promised to improve the software reusability by using different kind of techniques such as inheritance and polymorphism. Inability of the OO paradigm to address the entire separation of concerns concept is one of the reasons that reduce software reuse in OO (Zakaria *et al*., 2002). In the object-orientation programming concepts, systems are expressed as a collection of incorporated classes and class's instance (objects). However, usually the complex system and sub systems have some features that naturally cut cross other classes, components or modules (core classes) which lead accumulatively to increase the level of dependencies among these elements, when the dependencies are high, the reusability is low. Thus, in the OO, these crosscutting components (Aspects) are not represented in objects. All these are leading to have the aspect-orientation concepts on board (Zakaria *et al*., 2002).

Aspect-Orientation (AO) is not considered as a replacement for Object-Orientation (OO), moreover, AO concept is mainly focused on Aspect-Orientated Programming (AOP), with less attention paid to the early stage of Aspect-Oriented Software Development (AOSD) such as, design and modelling (Laddad, 2002).

AOP allows programmers to implement the concept of separation of concerns which is so important once it comes to software development processes (Magableh and Kasirun, 2007). Also, AOP overcomes the problem of code spreading over the core concerns, which is called as Code Tangling and Code Scattering. Nevertheless, OO is

Corresponding Author: Aws A. Magableh, School of Computer Science and Information Technology, Universiti Kebangsaan Malaysia, 43600 Bangi Selangor, Malaysia

incapable of efficiently solving this issue, when it implements the crosscutting concerns. Additionally, AOP solves these issues by implementing new modularity unit called as Aspect. The AOP has achieved a very considerable growth in the industrial environment and academic researches, which has led to an interest in AO techniques for all software life cycle stages (Przybylek, 2010).

Due to the importance and usability of AOP, it has been extended to cover the rest of software development stages. In fact, currently there are many AOM approaches, however, the most well-known and established approaches are those capable of handling aspect modelling using UML, because UML is the most popularly used modelling language tool in the industry (Ali *et al*., 2007). However, there is a lack of uniform standards. With the current state, UML is not capable of representing properties of AO constructs and the crosscutting nature of the aspects, at the design level (Grundy, 2000).

UML modelling extension mechanisms have been categorized into two types. The first one is the UML Meta Object Facility Metamodel (Heavy-weight) and the second one is the Constructing UML Profile (light-weight). The UML Meta Object Facility Metamodel (MOF metamodel) is referred as heavy-weight extension. The metamodel constructed can be as communicative as needed (Rui *et al*., 2009). Constructing the MOF metamodel is harder than constructing a UML profile and does not have enough supporting tool as compared with UML profile. Whereas, the second type UML Profile (light-weight) is usually called as light-weight extension because all the existing UML profile extension construction techniques do not implement any new UML Meta-model elements. Constructing UML profile extension techniques are usually considered to be predefined set of constraints, tagged values, graphical representations and stereotypes. Constructing UML profile extension method supplements aspect based on the flexibility and extendibility nature of the standard UML domain modelling (Przybylek, 2010).

The objective of this study is to provide a review on the current existing Aspect-Oriented Modelling approaches using Unified Modelling Language. Moreover, the study aims to provide some carefully selected evaluation and comparison criteria to compare the approaches against, which shows the limitation of these Aspect-Oriented Modelling approaches. Finally, this study is an eye-opening on the improvement that could take place to enhance the existing Aspect-Oriented Modelling Approaches such as taking all AspectJ constructs and all UML diagrams in consideration.

EVALUATION AND COMPARISON CRITERIA

Here, the selected AOUML modelling approaches are compared and contrasted. The core idea of this comparative analysis is to identify the gaps in the field and to set the platform to address those gaps by answering the research questions. This will enable us to unambiguously depict the approach to be employed for assembling the criteria catalogue and to describe a general narrative schema. The procedural rationalization, which follows the criteria design and assemblage, offers an essential platform for executing an extensive analysis of existing AOUML Modelling approaches. While performing the analysis we will make sure that none of the measurable criteria are omitted. Furthermore, the more clearly defined criteria and the apt measurement resources are not also excluded. We have illustrated the apparent catalogues for these criteria; generally, the comparison criteria should fulfil certain level of standardization and guidelines (Shukur and Mohamed, 2008) as follows:

- **Language specification (LS):** This type depicts few principles associated with the UML modelling languages. The UML Modelling Language Version (UMLMLV) is used by a particular approach, which indicates whether the approach uses latest version of UML or not, the UML edition numbering is indicator of this factor. The Extension Mechanism (EM) articulates the UML extension employed by a particular approach, precisely this factor specifies whether the extension is Light-Weight (LW) or Heavy-Weight (HW) and the indicators are either HW or LW. Diagram Type (DT) identifies the UML diagrams incorporated in the extension of a particular approach and the name of the diagram is the indicator of this factor
- **AspectJ constructs/syntax (AJC):** This is a very significant criteria, as far as this study is concerned, because we are going to investigate the complete list of AspectJ constructs, which are widely used and well-recognized in the AspectJ languages in the industry to execute reverse engineering (button-up). This extensive study will enable us to escalate the reliability, level of understanding and stable changeover from one stage to the other (Kande *et al*., 2002). The criteria have been inspired from Laddad (2002).The full or partial support indicates the values of this factor
- **Maturity issues (MI):** This catalogue indicates the capabilities of the selected UML Aspect modelling approaches. Even though there are a lot of approaches available, this study has focused only

the most popular and recognized approaches, which have drawn the attention of scholars (Reddy *et al*., 2006). This factor constitutes the maturity of the Modelling Examples (ME), which is employed to exhibit the dependability of the proposed modelling and the maturity of the Application in Real-World Projects (ARWP), which illustrates the applicability of the approach in realistic world

- **Tool support (TS):** This category emphasizes a number of criteria, which mainly focus on a tool support for the selected approaches. This catalogue is further classified as Modelling Support (MS), Code Generation (CG) and Model Kernel Extraction (MKE)
- **Complete framework support (CFS):** This factor highlights a composition of other factors. We call an approach as a complete framework, if it captures all the AspectJ constructs using all UML diagrams 2.4 (not only few) and has a comprehensive tool support for auto AspectJ generation and model kernel extraction in to text file such as Petal file

LITERATURE-BASED SURVEY

The Aspect-Oriented UML modelling approaches are bifurcated into as constructing UML profile and UML Meta Model Extension. Principally, constructing UML profile extension is called as light-weight extension. This is due to the fact that all the existing constructing UML profile extension techniques, do not apply any novel UML Meta-model elements. Furthermore, the constructing UML profile extension techniques are generally recognized as a predefined set of limitations, marked values, graphical representations and typecasts. Consequently, the constructing UML profile extension technique enhances the aspect, depending on the adaptability and extensibility of the standard UML domain modelling (Przybylek, 2010). The second category UML Meta Model Extension signifies the fundamental rules and norms for constructing the domain conceptual models to the Meta Model. The UML Meta Model Extension is recognized as model of a modelling language. Furthermore this category is a heavy-weight extension, due to the capability of proposing novel UML Meta models to signify the aspects and their crosscutting nature (Rui *et al*., 2009). Few studies of AOUML modelling have revealed that, there are a total of fourteen, matured and well-established approaches.

Implementation of software components using aspects: The Aspect-Oriented Component Engineering (AOCE) is aimed at distinguishing an assortment of slices or aspects from a system. A component is capable of providing services to other components or will get the services form its counterpart. Fundamentally, the aspects impact a lot of other components that are acknowledged by breakdown processes, such as perseverance and allocation. Further more aspects are employed by developers to illustrate various perceptions on the capabilities of components during requirements engineering and design. The AOCE depicts the aspect and its details. It offers a novel framework for the purpose of illustrating and analyzing the potentials of component from different angles (Grundy, 2000).

A toolkit for weaving aspect-oriented UML design: UML All Purpose Transformer (UMLAUT) toolkit is a different kind of tool used for the MOF mechanism. It is an aspect oriented UML model employed for effortlessly constructing explicit weavers for generating high-level comprehensive design models. The UMLAUT facilitates the developers to program the weavers at of UML Meta mode level. It offers an extendible and reusable general purpose operator for various applications with definite needs. All the AO designs might be developed with an application specific weaver that optimizes the weaving process confirmed by UMLAUT (Ho *et al*., 2002).

An UML-based aspect-oriented design notation for aspectJ: This is light-weight UML extensions intended to be a design notation for AspectJ. Basically it broadens the current UML standard notations and proposes a novel AspectJ weaving process (Stein *et al*., 2002).

Theme: An approach for aspect-oriented analysis and design: Theme/UML is used to generate distinct design models for each "theme" that evokes from the requirements phase later it summarizes the concerns, which signify some kind of functionality in a system. The Theme/UML is recognized as a heavy-weight extension of the UML metamodel version, due to its capability of augmenting novel elements to the basic representation. Fundamentally, the Theme/UML will not restrict the UML diagrams, which are used for modelling. On the other hand, package and class diagrams are exclusively used for modelling structures, whereas the sequence diagrams are employed for modelling behaviours (Clarke and Baniassad, 2005).

Weaving with state charts: This is aimed at independently modelling an aspect from a specific type of aspect-oriented programming language. Here the structural dependencies are expressed by class diagrams. This approach also signifies the machines model and the

behavioural dependencies of concerns and offers a directive to constantly enhance the modelling from class diagram to a prototype (Elrad *et al*., 2005).

Aspect-oriented software development with use cases: The significance of the use case driven software development method has been recognized by expanding the UML 2.0 metamodel. The Aspect-Oriented Software Development with Use Cases (AOSD/UC) constitutes an efficient process, which is capable of separating the concerns from the entire software development life cycle. In case of the design phase, the component diagrams are converted into class diagrams, while sequence diagrams are employed to model the behavioural features. Furthermore, the AOSD/UC models the concerns with the help of use case slices stereotype and lacks support tools (Jacobson and Ng, 2005).

Aspect-oriented software development with JAVA aspect components: This indicates a hybrid mechanism, where it amalgamates the UML profile extension and the abilities of UML, to produce various domain models and UML Meta object Facility model mechanism, by proposing new Meta object/notation. It proposes a platform dependent JAVA Aspect Component (JAC). The JAC comprises novel UML notations. It supports all the steps of Aspect-Orientation development, which ranges from design, to deployment. This approach employs the UML profile mechanism to design aspects by augmenting stereotypes to qualify classes implementing aspects and non functional concerns (Pawlak *et al*., 2005).

Directives for composing aspect-oriented design class models: Aspect-Oriented class design model comprises a set of aspect models and a primary model. Each aspect model depicts a attribute that crosscuts the essentials in the primary model. The aspect and primary models are aimed at obtaining an incorporated design view. It characterizes a composition method, which employs composition algorithm and decree, where the former is used when the default composition algorithm is known or likely to produce erroneous models. The prototype of this approach facilitates the composition of default class diagram (Reddy *et al*., 2006).

Presenting crosscutting structure with active models: This presents crosscutting structure, which is generally carried out using two means such as: (2) tree views, which enables the developers to manually combine information across numerous views and (2) static structure diagrams, which might be probably victimized by the intense graphical intricacy. An active model is an approach, which addresses these issues by introducing the accurate crosscutting structure at the appropriate time. "The right information is determined through automatic projection and abstraction operations that select elements and relationships likely to be of interest and that abstract those elements and relationships to control the diagram complexity when too many similar cases occur. The information is presented at the right time through a combination of a user-driven expansion operation that adds detail to the model and interaction features that show some information only on demand by the user" (Coelho and Murphy, 2006).

Join point inference from behavioural specification to implement: This presents a novel join point selection mechanism depending on the condition of machine specifications. The interfaces of a system encompass the requirement of the effects of method approved on the state of the module instance. This requirement does not describe the potential aspects, but explicitly describes the apparent behaviour of the module. We have illustrated the capability of a smart join point selection mechanism to conclude points that might be located deep inside the implementation of a module and to infer the particular pointcut that is totally articulated in terms of its specification element. It enhances the class diagram and the composite structure to obtain the static structure of the system and employs the machine extension to signify the behaviours of the system (Cottenier *et al*., 2007).

A concern architecture view for aspect-oriented software design: The concern architecture model is employed to cluster aspect designs in the context of software architecture. It offers an aspect-oriented perception while designing software. This model can be also viewed in the context of aspect analysis for analyzing the influences of the modifications or adaptabilities in concerns to be addressed by aspects. It includes a number of novel stereotypes to model aspects such as: <<Aspects>>,<<Concerns>>, <<Bind>>, <<replace>> and <<Unify>> (Katara and Katz, 2007).

Weaving multiple aspects in sequence diagrams: This has introduced an Aspect-Oriented UML approach using the standard UML. However, it has not proposed any new notation and has not used the ability of UML extension assuming to maintain the standards. Furthermore, it is originally based on Message Sequence Charts (MSC) a standardized scenario language and employs the UML 2.0 sequence diagram. In fact, no extensions are added to the UML Sequence diagram; instead a simplified Meta model for Sequence diagram has been designed, which

complies with the original UML Sequence diagram by converting model in the supplementary tool support (Klein *et al*., 2007).

Extending the UML metamodel to provide support for crosscutting concerns: It has presented an extension to the UML metamodel to unambiguously obtain the crosscutting concerns. It introduces an autonomous means to any programming language and cross platform. The newly produced metamodel can be represented in standard XMI format, moreover it lacks own tool to illustrate the modelling; however it employs the current CASE tools to read this XML format. This language-independent aspectual description can facilitate model transformations that are significant to software development and maintenance, such as forward engineering, reverse engineering and reengineering (Sharafi *et al*., 2010).

Separation of crosscutting concerns at the design level: an extension to the UML metamodel: Aspect-Oriented UML Modelling has proposed an extension by instituting a novel package called as AoUML, which comprises elements that signify the primary AO concepts such as: aspect, advice, pointcut, parent declaration, introduction and crosscutting dependency. It has also proposed reuse elements from the UML 2.1.2 infrastructure and superstructure specifications (Przybylek, 2010).

RESULT ANALYSIS

Table 1 depicts the analysis and the evaluation based on this study selected criteria. It shows that none of the most used (surveyed) approaches did use the latest UML editions 2.4 and it implicitly shows that none of these have tried to improve the proposed approach to fit the latest UML edition.

Table 1 gives an idea that some of these approaches depend on the UML edibility to be extended to model different domain model (lightweight), some more used new notations (heavyweight), some did not amend the standard UML and tired to use it as is and this is what Klein proposed. Pawlak tried to propose a use of both UML extensions.

Table 1 depicted that only few types of UML diagrams have been used by either LW or HW extensions. Majority of the approaches focused on proposing an extension to model aspects using class, sequence diagrams. Some more tried to use communication, package and use case diagrams. It has been concluded that none were focused on the whole UML diagrams as one framework and none proposed a complete set of aspect modelling notation using all UML diagrams.

Table 1 explains explicitly that some of the approaches are using AspectJ as a baseline for modelling aspects. However, none of the approaches proposed a complete modelling set for all AspectJ detailed constructs.

Table 1 addressed the maturity concerns of the approach. Maturity has been measured by providing a modelling example and that example being a useful application in the real life. As stated in Table 1, some of these approaches have used a modelling example to demonstrate their proposition and some other have not done that. For those who have done it, some approaches provided too easy and duplicated examples and some other provided too complicated examples which makes it so hard to ready, understand and make use of it.

Table 1: Evaluation and comparison criteria of different studies

	Criteria								
	LS				MI		TS		
References	UMLMLV	EM	DT	AJC	ME	ARWP	MS	CG	MKE
Ho *et al.* (2002)	1.1	HW	Class	Partial	N	N	N	N	N
Grundy (2000)	1.x	HW	Class, component	Partial	Y	N	N	N	N
Clarke and Baniassad (2005)	1.x	HW	Sequence, class, package	Partial	Y	Y	Y	N	N
Elrad *et al.* (2005)	1.x	LW	State, class	Partial	Y	N	Y	N	N
Stein *et al.* (2002)	1.x	LW	class, collaboration	Partial	Y	N	Y	N	N
Pawlak *et al.* (2005)	1.x	LW/HW	Class	Partial	Y	Y	Y	Y	N
Coelho and Murphy (2006)	2.0	HW	Class	Partial	Y	N	Y	N	N
Jacobson and Ng (2005)	2.0	HW	Use case, sequence, communication, component	Partial	Y	Y	Y	N	N
Reddy *et al.* (2006)	2.0	HW	Package, class	Partial	Y	N	Y	N	N
Cottenier *et al.* (2007)	2.0	LW	Sequence, state, class, package	Partial	Y	Y	Y	Y	N
Katara and Katz (2007)	2.0	LW	Package	Partial	Y	N	N	N	N
Klein *et al.* (2007)	2.0	N/A	sequence	Partial	Y	N	Y	N	N
Przybylek (2010)	2.2	HW	class, package	Partial	N	N	N	N	N
Sharafi *et al.* (2010)	2.3	LW	sequence, class	Partial	Y	Y	N	N	N

AJC: AspectJ constructs, MI: Maturity issues, TS: Tool support, UMLMLV: UML modelling language version, EM: Extension mechanism, DT: Diagram type, AJC: AspectJ constructs, ME: Modelling examples, ARWP: Application in real-world projects, MS: Modelling support, CG: Code generation, MKE: Model kernel extraction, LW: Light-weight, LS: Language support, HW: Heavy-weight, Y: Yes, N: No

Finally, Table 1 shows that majority of the approaches did not propose their own modelling tool; they used already existing tools and plug-in. That leads to lack of support for AspectJ code generation, modelling extraction into XML format such as SVG.

After the analysis and brainstorming took place on the result of this study, it has concluded that there some limitations which have to be addressed by the future researches. This study has proved that none of the existing approaches is based on the latest UML 2.4 edition. Additionally, it has been shown that majority of the existing approaches are based on some UML diagrams to represent and model Aspects and none of them have came out with a complete set of modelling notations for all UML diagrams and some UML diagrams such as timing diagrams have not been put on the table of the discussion yet. Finally all the approaches have provided some partial support to model Aspects based on AspectJ detailed constructs. This study has opened new venue for research to see the ability to provide a complete Aspectual UML 2.4 modelling framework based on AspectJ detailed programming constructs to maintain the constancy and tractability in all Aspect-Oriented Software Development.

CONCLUSION

In this study, we had reviewed different Aspect-Oriented UML modelling approaches. Additionally, perspectives of various authors had been elucidated and analysed. Moreover, we had studied the UML extension mechanisms provided to model aspects using UML. We had analyzed the surveyed approaches by selecting some common and critical factors and had presented in a tabulated format for clear understanding and readability. Each and every one of these selected approaches had been evaluated based on this study analysis criterion. Finally, we had found out that UML with its current state is unable to efficiently represent aspects, moreover, not all approaches depends on Aspect-Oriented programming to model aspects, majority of these approaches do not represent all aspect-oriented programming using all UML diagrams rather they focus one class diagram, sequence and state diagrams, which should not be enough to effectively represent constructs of the aspect in the early stage of software development.

REFERENCES

Ali, N.H., Z. Shukur and S. Idris, 2007. A design of an assessment system for UML class diagram. Proceedings of the International Conference on Computational Science and Applications, August 26-29, 2007, Kuala Lampur, pp: 539-546.

Clarke, S. and E. Baniassad, 2005. Aspect-Oriented Analysis and Design: The Theme Approach. Addison-Wesley, New York, ISBN: 9780321246745, Pages: 366.

Coelho, W. and G. Murphy, 2006. Presenting crosscutting structure with active models. Proceedings of the 5th International Conference on Aspect-Oriented Software Development, March 20-24, 2006, New York, pp: 158-168.

Cottenier, T., A. van den Berg and T. Elrad, 2007. Join point inference from behavioral specification to implementation. Proceedings of the 21st European Conference on Object-Oriented Programming, July 30-August 3, 2007, Berlin, Germany, pp: 476-500.

Elrad, T., O. Aldawud and A. Bader, 2005. Expressing Aspects Using UML Behavioral and Structural Diagrams. In: Aspect-Oriented Software Development, Filman, R.E., T. Elrad, S. Clarke and M. Aksit (Eds.). Addison-Wesley, New York.

Grundy, J., 2000. Multi-perspective specification, design and implementation of software components using aspects. Int. J. Software Eng. Knowledge Eng., Vol. 10.

Ho, W., J. Jezequel, F. Pennaneac and N. Plouzeau, 2002. A toolkit for weaving aspect oriented UML designs. Proceedings of the 1st International Conference on Aspect-Oriented Software Development, April 22-26, 2002, Enschede, The Netherlands, pp: 99-105.

Jacobson, I. and P.W. Ng, 2005. Aspect-Oriented Software Development with use Cases. Addison-Wesley, New York, ISBN: 9780321268884, Pages: 418.

Kande, M., J. Kienzle and A. Strohmeier, 2002. From AOP to UML: Towards an aspect-oriented architectural modeling approach. http://infoscience. epfl. ch/record/54711/files/IC_TECH_REPORT_200258.pdf

Katara, M. and S. Katz, 2007. A concern architecture view for aspect-oriented software design. Software Syst. Model., 6: 247-265.

Klein, J., F. Fleurey and J.M. Jezequel, 2007. Weaving multiple aspects in sequence diagrams. Trans. Aspect-Oriented Software Dev., 4620: 167-199.

Laddad, R., 2002. AspectJ in Action: Practical Aspect-Oriented Programming. Manning Publications, Greenwich, CT., USA.

Magableh, A.A. and Z.M. Kasirun, 2007. Collaborative aspect-oriented requirements tool. Proceedings of the 3rd Malaysian Software Engineering Conference: Striving for High Quality Software, December 3-4, 2007, Selangor, pp: 12-17.

Pawlak, R., L. Seintuier, L. Duchien, L. Martelli, F. Legond and G. Florin, 2005. Aspect oriented software development with JAVA spect components. Proceedings of the 4th International Conference on Aspect-Oriented Software Development, March 14-18, 2005, Chicago, Illinois, USA.

Przybylek, A., 2010. Separation of crosscutting concerns at the design level: An extension to the UML metamodel. Proceedings of the International Multiconference on Computer Science and Information Technology, October 18-20, 2010, Wisla, Poland, pp: 551-557.

Reddy, R., S. Ghosh, R. France, G. Straw and J.M. Bieman *et al*., 2006. Directives for Composing Aspect-Oriented Design Class Models. In: Transactions on Aspect-Oriented Software Development I, Rashid, A. and M. Aksit (Eds.). Springer, New York, USA., ISBN-13: 9783540329725, pp: 75-105.

Rui, W., M. Xiao-Guang, D. Zi-Ying and W. Yan-Ni, 2009. Extending UML for aspect-oriented architecture modeling. Proceedings of the 2nd International Workshop on Computer Science and Engineering, October 28-30, 2009, Qingdao, pp: 362-366.

Sharafi, Z., P. Mirshams, A. Hamou-Lhadj and C. Constantinides, 2010. Extending the UML metamodel to provide support for crosscutting concerns. Proceedings of the 8th ACIS International Conference on Software Engineering Research Management and Applications, May 24-26, 2010, Montreal, QC., Canada, pp: 149-157.

Shukur, Z. and N.F. Mohamed, 2008. The design of ADAT: A tool for assessing automata-based assignments. J. Comput. Sci., 4: 415-420.

Stein, D., S. Hanenberg and R. Unland, 2002. An UML based aspect-oriented design notation. Proceedings of the 1st International Conference on Aspect-Oriented Software Development, April 23-26, 2002, Enschede, The Netherlands.

Zakaria A., H. Hosny and A. Zeid, 2002. A UML extension for modeling aspect oriented systems. Proceedings of the 5th International Conference on the Unified Modeling Language, September 30-October 4, 2002, Dresden, Germany.