Journal of Applied Sciences

ISSN 1812-5654

science alert

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

Journal of Applied Sciences 13 (1): 1-13, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.1.13

Systematic Review on Aspect-oriented UML Modeling: A Complete Aspectual UML Modeling Framework

Aws Magableh, Zarina Shukur and Noorazean Mohd Ali
Department of Software Engineering, Faculty of Computer Science and Information Technology,
Universiti Kebangsaan Malaysia (UKM), 34600, Kajang, Malaysia

Abstract: This study presented a systematic review on Aspect-Oriented Modeling by using UML, which is the popular and well-established modeling language in the industry. Further, the Aspect-Orientation is a complement for Object-Orientation. Thus, it would be natural to investigate the adaptability of UML to Aspect-Orientation. Our research questions focused on investigating the Aspect-Oriented UML (AOUML) approaches based on the literature and for having the benefits of a complete framework for AOUML. The objective of this study was to systematically identify and analyze the Aspect-Oriented Modeling approaches using UML. Therefore, a systematic literature review has been carried out on the existing studies published in the conferences, proceedings and journals. Based on the review, it is evident that the complexity and size systems have grown up, which accumulatively led to the manifestation of new concerns. Moreover, these new concerns that have arrived onboard have cut-cross other concerns and core classes in the system by its nature. Due to this fact, the concept of Advance Separation of Concerns (ASoC) has been put on the table of discussions and the need for an approach to model and represent these crosscutting concerns (Aspect), which is responsible for producing, spreading and tangling representation, throughout the development of life cycle, which is vital and this is considered to be our motivation of systematic review of Aspectual UML modeling. As a result, we have searched the proper databases using the suitable keywords, which match our research questions; we have also collected 468 studies and screened them to minimize the number of studies to 73, which are more appropriate for the present study.

Key words: Aspects, aspectual UML diagrams, aspect modelling, aspect modularization, aspect representations, aspect-oriented UML

INTRODUCTION

The needs for the Separation of Concerns (SoC) are in demand during the software development processes (Magableh and Kasirun, 2007) and it has received wide spread attention (Dahiya and Sachdeva, 2007). Therefore, a lot of approaches have been proposed and used such as: Subject-orientation (SO), Feature-orientation (FO) and the most popular Object-orientation (OO), to represent the concerns of the system (Chitchyan *et al*., 2005). Due to the weakness of Object-Oriented Analysis and Design approach, the proposition in handling the crosscutting concerns (Aspects) (Uetanabara *et al*., 2009), Aspect-Orientation Analysis and design (AOAD) approach has been proposed to focus on the crosscutting concerns and its effect on multiple classes (Fazal-e-Amin and Oxley, 2010). In addition, the AOAD is further divided into Aspect-oriented Requirement Engineering (AORE), Aspect-Oriented Architecture (AOA) and Aspect-Oriented Design Modeling (AODM). All these fields of AO focus on efficiently handling the Aspects throughout the software life cycle. However, the most neural field is the modeling field.

In fact, there are quit a good number of researches have been carried on AODM as software design process and modelling is considered to be very vital and important topic (Shen *et al*., 2011), due to that, AODM literatures have addressed the modeling of Aspects using different techniques. Some are based on Architectures View (Katara, 2002), Aspect-Oriented Language (Groher and Baumgarth, 2004; Groher and Schulze, 2003), XML representation format (Suzuki and Yamamoto, 1999), some other are based on component engineering (Grundy, 2000) and Component Views (Muller, 2004). Furthermore, the AODM using UML Diagram is the most well-known technique (Ali *et al*., 2007a; Changchien *et al*., 2002), which has drawn the attention of a lot of researchers, the AO is considered as an extension to OO and UML.

Corresponding Author: Aws Magableh, Department of Software Engineering,
Faculty of Computer Science and Information Technology,
Universiti Kebangsaan Malaysia, 34600, Kajang, Malaysia

UML is the most accepted design language in Software Engineering, the UML is considered to be a standard in the industry and it provides a powerful set of modeling tools and diagrams (Asteasuain *et al*., 2008). Thus, the UML for aspect has to be investigated similar to OO. Finally, there are many propositions on Aspect-Oriented Design Modeling (AODM) using UML approaches. However, there is a lack of uniform standard (Albunni and Petridis, 2008; Zhang, 2005). UML with its current state of the design level is unable to properly model all domains and languages constructs (Qureshi and Manuel, 2006) as well as it is unable to represent Aspect-Orientation constructs and the crosscutting nature of the Aspects (Marco *et al*., 2008).

Programming and modeling language exist in a relation of mutual support. Thus, our research focused on button-up technique. It starts from the well-established aspect coding languages level (AspectJ), which is a hot programming topic nowadays (Zhang *et al*., 2009) and moves backward to design level to generate an Aspect-Oriented UML constructs notation (Aspectual UML). As there is a lack of Aspectual design notations in AODM for designing Aspect Oriented code (AspectJ in this case) (Muley *et al*., 2010), this would mainly depend on having a look at the coding constructs, then try to implement it in a representational notation to be used in the design modeling level. As a result, the level of encapsulation would be increased, not only at the coding level, but rather at design level too. Concurrently, the level of consistency would be maintained to high level as all the coding constructs would be denoted at design phase in the first place (Kande *et al*., 2002). This study focused on investigating the Aspect-Oriented UML (AOUML) approaches based on the literature and for having the benefits of a complete framework for AOUML.

A SYSTEMATIC REVIEW

This review is based on a structured Systematic Literature Review (SLR) (Ramey and Rao, 2011). A systematic literature review/mapping refers to the review, evaluation and interpretation of the all related available research and primary studies that are relevant to clearly formulated questions, followed by extracting and analyzing information included in the review (Ramey and Rao, 2011). SLR is composed of the following steps:

- Systematic Mapping Planning
- Conduction of the search
- Selection of the primary studies and
- Analysis and map building

The four steps and their output artifacts are depicted below in Fig. 1.

Step 1: Systematic mapping plan: Here, the plan that we have used to conduct the research is executed. It consists of three sub steps:

- Good formulation for the RQ
- Selection of the Databases and Resources and
- The Inclusion Criteria (IC) and Exclusion Criteria (EC)

Research questions: This section explains the research questions used in this study. The significance of the questions stated below, is nothing but the signature and the semantics represent all dimensions of our research towards aspectual UML modeling. Apart of that, some peer-reviewed well established studies have recommended that such questions scope to be included.

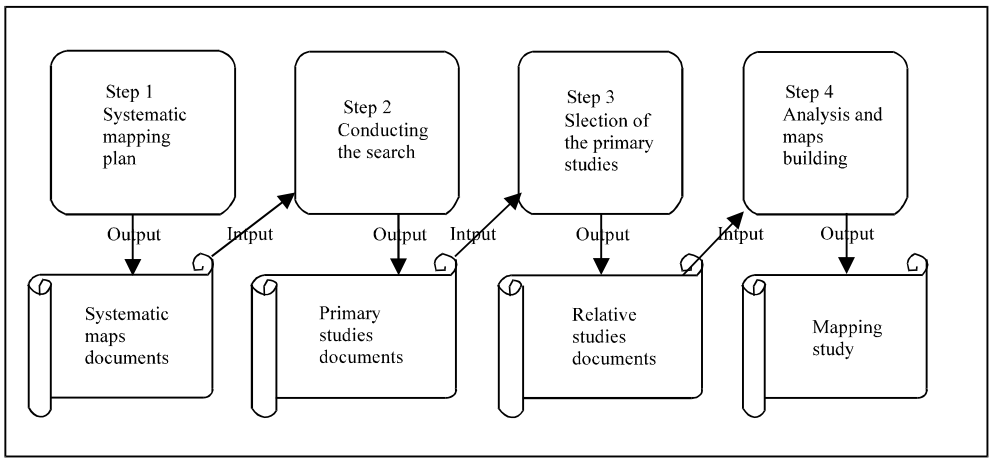

Fig. 1: Systematic literature review processes

RQ1: What are the Aspect-Orientation UML modeling techniques and approaches which have been drawn by literature?

RQ2: Does the literature have complete set of UML diagrams extensions to model Aspect-Orientation?

RQ3: Does the literature have complete Aspect-Orientation modeling framework based on AspectJ constructs and its language specifications?

RQ4: What are the implication of applying UML 2.4 latest infrastructure and super structure on Aspect-Orientation modeling?

With respect to RQ1, we are more focused on the existing Aspect modeling techniques and tools that have been used to model crosscutting in the Software Development. We have also considered the scope and limitation of a lot of techniques including the current aspect modeling techniques, in terms of usability, applicability and limitations.

With respect to RQ2, the literatures have focused on modeling Aspect-orientation either at early stage of software life cycle or at the later stages, however, none of the researches have proposed a complete Aspectual framework. For instance, some studies have proposed an Aspect-Oriented Use Cases (Araujo and Moreira, 2003), some others have proposed Aspect-Oriented Sequence diagram (Klein *et al*., 2007), few more have addressed the aspect-oriented class diagram (Zhang, 2005; Katara, 2002). However, none of them have proposed a complete set of aspectual UML diagram. Additionally, even (Ali, 2010) stated that state chart is among the most important UML diagram, however, only few studied the adoptability of Aspects into it. Apart of this, none of the researches have investigated the applicability of using UML timing diagram as instance. Therefore, this research focused on the applicability of a complete and comprehensive aspectual UML 2.4 framework, as majority of the researchers have carried their researches and proposition based on older UML versions.

With respect to RQ3, literatures have focused on reverse engineering approaches. It starts from bottom-up approach, it investigated the well established aspect programming languages, then tries to model its constructs and syntax. Some approaches have proposed and extended AspectJ, however, none of the studies have proposed a complete set of modeling for all AspectJ language specification for all UML diagrams.

Since we are focusing on the extended UML 2.4, which is the latest version of UML (Muller, 2004) that supports Aspect modeling in AOSD. The Information System Development (ISD) has many methodologies such as spiral (Kaur *et al*., 2012) in the field to address different software development phases (Yusof *et al*., 2011), AOSD is one of them, due to that we need to justify the significance our technique compared to proposals of other studies, more over we need to rationalize the employment of UML 2.4 compared with other versions of UML and ultimately elucidate the goals, aims and contributions that could be achieved by our modeling. The RQ4 is more concerned about such in AOSD.

Selection of the Databases and Resources (Search Process): In this step we have tried to find out the databases and resources that we are going to use. The selection of the databases usually depends on the following three criteria:

- Content update (the sources are always up to date with the latest publications)
- The availability of the full text of the papers and
- The Quality of research accuracy and preciseness is taken in the consideration

The Table 1 shows the Electronic Databases and Resources used in this systematic review. These databases are the most popular, standardized, well-established and most relevant to the Aspect-oriented modelling.

The Inclusion criteria (IC) and exclusion criteria (EC): IC and EC are considered as important elements of the Systematic Review (Ramey and Rao, 2011). These criteria helps us in including the primary studies that shows relevancy to the research questions and exclude the studies, which do not have relevancy to answer the research questions. Table 2 shows the EC and Table 3 shows the IC.

Table 1: Electronic databases and resources

Database	Location
ISI web of science	www.isiknowledge.com
Medline	www.mendeley.com/
IEEE explore	http://ieeexplore.ieee.org
ACM digital library	www.portal.acm.org
Scopus	www.scopusdirect.com
Aspect oriented modeling	www.aspect-modeling.org/
Disser abstracts	http://wwwlib.umi.com/dissertations
Proquest educations journals	www.proquest.com/
UKM library computer science	www.ukm.my/library/
Aspect oriented software develop	http://aosd.net/
Aspect-oriented software engin. SI	http://www.comp.lances.ac.uk/computing/aod/AODatabases.php
Science direct	www.science direct.com
EBSCOHOST	www.ebscohost.com

Step 2: Conducting the search: This step is divided into two sub steps:

- Choosing Key words and
- Search strings in databases. By approaching the end of the conducting the search (step 2)

We will come out with the final edition of the primary related studies, using the proper keywords, with the database related string searching format.

Choosing key words: The keywords have been chosen carefully. It must be simple and meaningful enough to cover up all the primary studies, which can bring back more accurate results in the searched databases. The selected keywords of our research were shown previously in the keywords section.

Table 2: Exclusion criteria created for systematic mapping

Criteria ID	Exclusion criteria description
EC1	The study is written in different language
EC2	The study is not addressing aspect oriented modelling
EC3	The Study is talking about modelling aspects using non-UML diagrams
EC4	The study is not addressing UML and aspects in conjunction
EC5	The study that is elaborating AODS but it does not address modelling part

Table 3: Inclusion criteria created for systematic mapping

Criteria ID	Inclusion criteria description
IC1	Study that is directly addressing UML and aspects-oriented
IC2	Study that is talking about UML extensions to model aspects
IC3	Study that talks about tools for Aspect modelling with UML
IC4	Study that talks about aspect oriented programming concepts

Search strings in databases: Every database search engine has its own strings settings, as everyone has its special format. Generally, the strings used to search the proper databases have to compose from AND/OR connections, as shown below:

- (UML AND (Aspects OR Aspect Modelling OR Aspectual OR AODS)) or
- (Aspect AND (Modelling OR Representation OR Capturing OR UML))

Step 3: Selection of the primary studies: As a result of applying the above keywords (Step 2) on the listed data sources (Step 1), we have identified that the total number of primary studies are 468. Furthermore, we have extracted the proper direct related studies, by removing the duplication and redundancies and by reading the abstract of the articles and the full text. As a result, we got 73 primary studies related to our topic as shown in Table 3. The qualitative focused studies are selected based on the systematic review approaches rules and conditions (Ramey and Rao, 2011). Additionally, 9 out of the 14 primary related studies have been taken from latest peer-reviewed conference/journals. Table 4 describes the screening of articles in numbers and total number of included study for each research questions. Figure 2 shows the screening process used for our research.

Figure 3 shows the trend of screening process. Initially, we had a huge number of articles and then they have been reduced by screening the articles till we get the primary studies. Figure 4 illustrates an idea about the

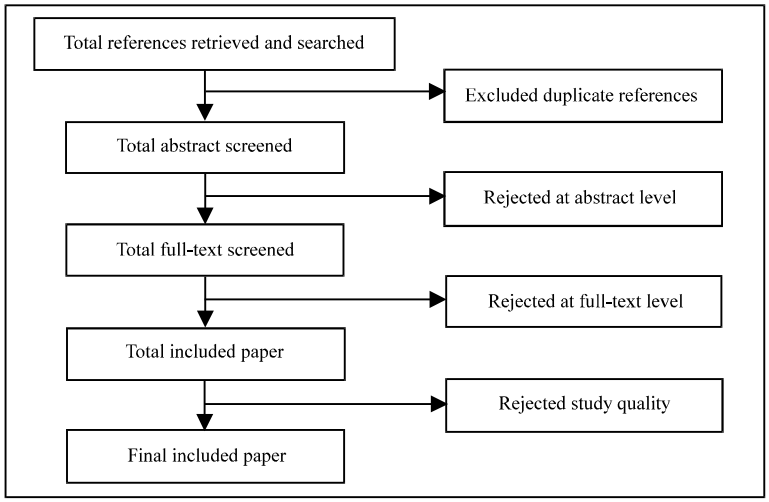

Fig. 2: The screening and selection process

Table 4: Articles screening in numbers

Research question	Total references retrieved	After excluded duplicate	Total abstract screened	After abstract screened	Total full-text screened	Total included paper	Final included study
RQ1	285	242	242	82	64	64	64
RQ2	160	132	102	55	45	6	6
RQ3	23	16	13	10	4	4	3

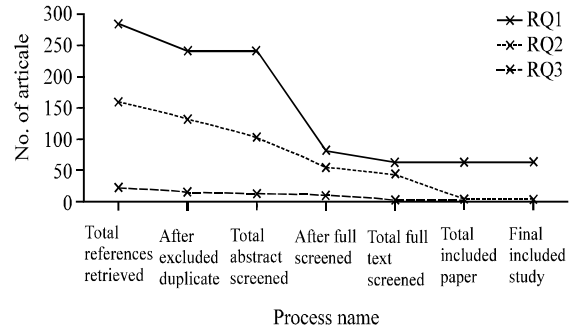

Fig. 3: Trend of articles screening process

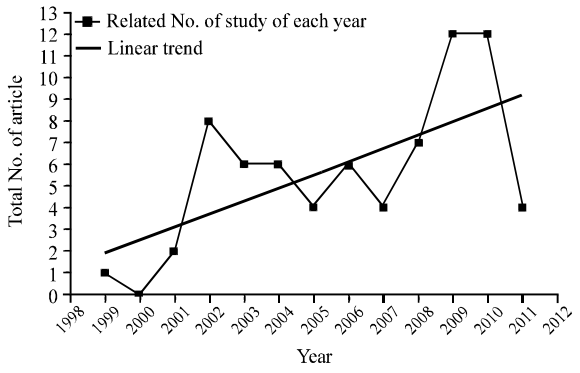

Fig. 4: Year-wise distribution

studies in terms of year distribution; we have observed that Aspect-Oriented UML study have started at the end of 1990 sec. We have identified that there are quite good number of researches have been carried out. Since 2006, the number has started increasing gradually. This gives us an indication that the field of the study was not stable and has not been standardized yet and the rooms are open for researchers.

Step 4: Analysis and build mapping: Figure 5 depicts the connection between the study areas of our research. Initially, Software Analysis and Design approaches are divided into Object-Orientation (Letherbirdge and Laganiere, 2001), Subject-Orientation (Ossher and Tarr, 1999), feature-Orientation (Cohen *et al*., 1992) and Aspect-Orientation, which is divided into Aspect-Oriented Requirement Engineering (AORE), Aspect-Oriented Architecture (Bass *et al*., 1998; Shaw and Garlan, 1996) and Aspect-Oriented Design Modeling (AODM), which specify the behavior and structure of the software system.

Our work has focused on AODM. In fact, AODM is concerned with representing and capturing the dynamic and static view of aspects and characteristics of aspects from the early stage of software life cycle. As a result, a lot of researchers came out with different kinds of modeling methods and techniques such as: Theme/UML (Clarke and Baniassad, 2005), SUP (Omar *et al*., 2002), Cocompose (Dennis and Bergmans, 2002), Aspect at Design Time (ADT) (Jose *et al*., 2000) and Aspect-Oriented UML Diagrams modeling, which is the focus of our research.

In reality, there are a lot of Aspect-Oriented UML modeling (AOUML) approaches, however, only few of them have come of age and have been presented at acknowledged conferences and journals. The AOUML approaches are categorized into two general categories. The first category is called constructing/building UML profile. Basically, constructing UML profile extension is usually called as light-weight extension, because all the existing UML profile extension techniques do not implement any new UML Meta-model elements. Also, the constructing UML profile extension techniques are usually considered to be predefined set of constraints, tagged values, graphical representations and stereotypes (Elminir *et al*., 2011). Thus, constructing UML profile extension method supplement the aspect based on the flexibility and extendibility nature of the standard UML domain modeling (Przybylek, 2010). The second category is UML Meta Model Extension. It refers to the Meta Model of basic rules and principles used to construct the domain conceptual models, it is considered as model of a modeling language. Also, this category is considered to be a heavy-weight extension, as it does proposes new UML Meta model to represent aspects and its crosscutting nature (Rui *et al*., 2009).

The major difficulty in comparing and critically analyzing AOM approaches and their benefits of modeling framework, is the lack of a common understanding for the basic ingredients of aspect-oriented modeling, Fig. 6 shows the criteria that we are based on and compared against.

Criteria categories: Here, the description of the criteria used to compare the AOUML selected modeling approaches. The motivation behind the selection of these criteria is that, to show the gaps in the knowledge and how our work will fill in such gaps by answering the research questions. Thereby, clearly stating the methodology used for assembling the criteria catalog and defining a common description schema. The methodological justification behind criteria design and

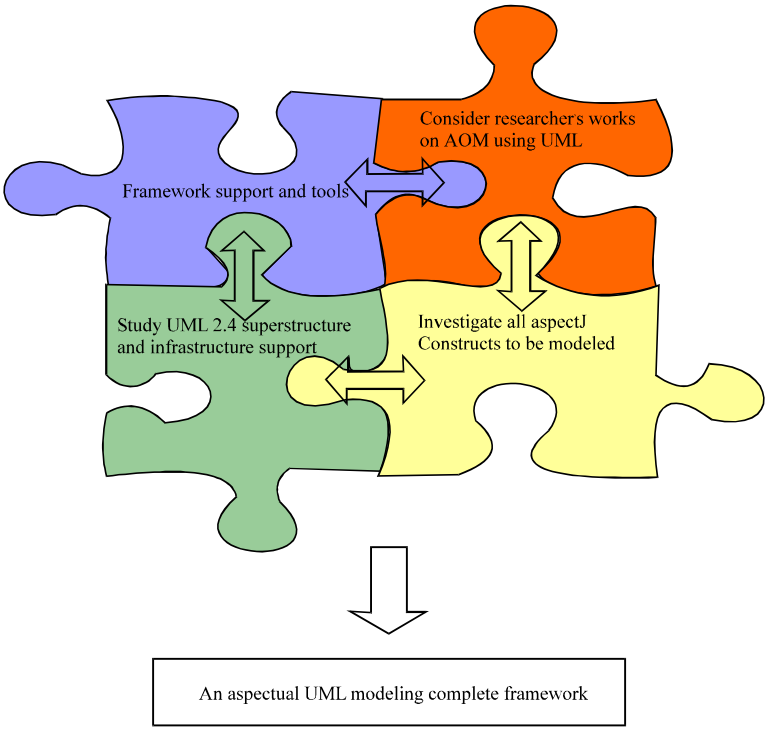

Fig. 5: Connection between the study's areas

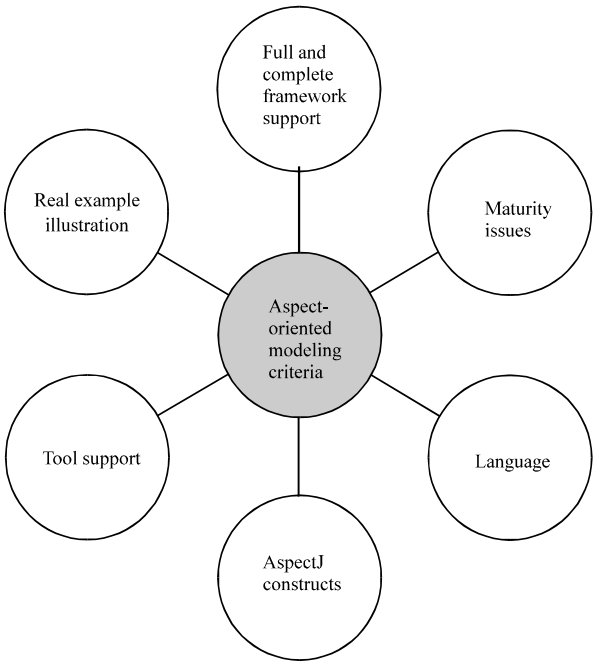

Fig. 6: Criteria categories

assembly is to provide a prerequisite for a comprehensive analysis of existing AOUML approaches and thus allows comparing different AOUML Modeling approaches in greater detail. However, no measurable criteria have been excluded. Moreover, criteria with an ambiguous definition and/or no appropriate means of measurement are also excluded as well as criteria that does not meet a standard guideline has been excluded (Shukur and Mohamed, 2008). Clear catalogues for these criteria are briefly elaborated in the next sections. An abbreviation has been added to each one of them:

- **Language Specification (LS):** This category describes some criteria related to the modeling language. The UML Modeling Language Version (UMLMLV), Extension Mechanism (EM), Language Purpose (LP), Platform Dependency (PD), Diagram Type (DT), Modeling Process (MP) this obtained from (Wimmer *et al*., 2011), Traceability (T) and Adoptability (A)
- **AspectJ constructs/syntax (AJC):** This catalogue contains vital criteria, as we are going to study full list of AspectJ constructs-most used and well-established AspectJ language in the industry-to perform a reverse engineering (button-up). This would increase the consistency, level of understanding and the smooth transition from one stage to another (Kande *et al*., 2002). The criteria have been inspired from (Laddad, 2002). The

following are the criteria: Dynamic Crosscutting Support (DCS), Static Crosscutting Support (SCS), Join Points Types (JPT), Pointcut Types (PCT) and Advice Types (AT)

- **Maturity Issues (MI):** This catalogue investigates the maturity of the approach proposed, in fact, many approaches have been proposed, but this study will focus only few of them, which we believe are well-established and have gained good reputation among researchers (Wimmer *et al*., 2011). It is divided into Modeling Examples (ME), Application in Real-World Projects (ARWP) and Usability (U)
- **Tool Support (TS):** This catalogue category has highlighted some criteria that have concentrated on a tool support for any proposed research. It has been divided into Modeling Support (MS), Code Generation (CG) and Model Kernel Extraction (MKE)
- **Real example illustration (REI):** This catalogue considers the applicability and reality of the demonstrated example used by the author. It is dived into Used Example (UE). Complexity (C)
- **Complete framework support (CFS):** This category highlights a composition of other categories. We call a proposition as a complete framework if it does capture all the AspectJ constructs, all UML diagrams 2.4 (not only few) and has a comprehensive tool support for auto AspectJ generation and model kernel file extraction

Aspect-oriented UML modeling approaches: Aspect-oriented UML modeling approaches is categorized into two general categories. The first category is called constructing/building UML profile. Basically, constructing UML profile extension is called as light-weight extension, because all the current existing propositions constructing UML profile extension techniques, do not implement any new UML Meta-model elements. Also, constructing UML profile extension techniques are usually considered to be a predefined set of constraints, tagged values, graphical representations and stereotypes. Thus, constructing UML profile extension method supplement the aspect based on the flexibility and extendibility nature of the standard UML domain modeling (Przybylek, 2010). The second category is UML Meta Model Extension. It refers to the Meta Model of basic rules and principles use to construct the domain conceptual models, it is considered as model of a modeling language. Also, this category is considered to be a heavy-weight extension, as it does proposes new UML Meta model to represent aspects and its crosscutting nature (Rui *et al*., 2009). The literature of AOUML modeling shows that, there are fourteen approaches, where they are considered mature enough and well-established.

The Aspect-oriented Component Engineering (AOCE) focuses on recognizing a mixture of slices or aspects of an overall system (Grundy, 2000). A component provides services to, or services it uses from other components. Aspects are horizontal segments through a system, which basically affect many other components identified, by the process of decomposition, such as persistence and distribution. Developers use aspects to describe different perspectives on component capabilities during requirements engineering and design. The AOCE proposes representation for Aspect and aspects details. It provides a new framework for describing and reasoning about component capabilities from multiple perspectives.

In literature (Ho *et al*., 2002), UML All Purpose Transformer (UMLAUT) toolkit is another use for the MOF mechanism. It is an aspect oriented UML models used for easily building specific weavers for producing detailed design models from high level. It enables the developer to program the weavers at the level of UML Meta mode. The UMLAUT provides the user with a general purpose operator that can be extended and as well as reused for different application with specific demands. Each AO design may be developed with an application specific weaver that optimizes the weaving process demonstrated by UMLAUT.

Stein *et al*. (2002) considered to be one of the light-weight UML extensions. It has been developed as a design notation for AspectJ, it extends the existing UML standard notations. It comes with a new production AspectJ weaving process.

Theme/UML is used to produce separate design models for each "theme" elicited from the requirements phase and then it does encapsulate the concern representing some kind of functionality in a system (Clarke and Baniassad, 2005). The Theme/UML is considered to be a heavy-weight extension of the UML metamodel version, as it does add some new element to the standard representation. Basically, the Theme/UML poses no restrictions on the UML diagrams that might be used for modeling. Nevertheless, package and class diagrams are specifically used for structure modeling and sequence diagrams are used for behavior modeling.

Elrad *et al*. (2005) aims to model aspect independently from the existing types of the Aspect-Oriented programming languages. The Class diagrams are

used to express the structural dependencies and state the machines model and the behavioral dependencies of concerns. The approach provides a guideline on how to refine the modeling continually from class diagram to the state of model machine.

Jacobson and Ng (2005), used case driven software development method has been realized by extending the UML 2.0 metamodel. Aspect-Oriented Software Development with Use Cases (AOSD/UC) comes with a systematic process that focuses on the separation of concerns throughout the software development life cycle, that is, from requirements engineering with use cases down to the implementation phase. For the design phase, component diagrams can be refined into class diagrams, while sequence diagrams are used to model behavioral features. Concerns are modeled using a use case slices stereotype and the approach does not come with any support tools as majority of the existing approaches are depending on the existing tools such as Rational Rose (Ali *et al*., 2007b) and MagicDraw which they do not provide the option to represent crosscutting concerns (aspects) efficiently as well as There are many Computer-Aided Software Engineering (CASE) Tools for visual developing class diagrams in 2D and 3 D (Sanatnama and Brahimi, 2010) which do not consider Aspects as well.

Pawlak *et al*. (2005) proposed a mix mode mechanism, where it makes use of the UML profile extension and UML ability to be extended to model different domain models as well as UML Meta object Facility model mechanism, by proposing new Meta object/notation. It proposes a Java Aspect Component (JAC), which is a platform dependent. This JAC comes with a new UML notations and an implementation to it as well. It does support for all steps of development for Aspect-Orientation from its design, to its implementation ending with the deployment. This approach uses the UML profile mechanism to design aspects by adding stereotypes to qualify classes implementing aspects and non functional concerns.

Aspect-oriented Class Design Model consists of a set of aspect models and a primary model (Reddy *et al*., 2006). Each aspect model describes a feature that crosscuts elements in the primary model. The aspect and primary models are composed to obtain an integrated design view. It describes a composition approach that utilizes a composition algorithm and composition directives. Composition directives are used when the default composition algorithm is known or expected to yield incorrect models. Its prototype tool supports default class diagram composition.

Coelho and Murphy (2006) presented crosscutting structure usually is done using two ways:

- Tree views, which involves developers in combining information across multiple views manually and
- Static structure diagrams, which be likely suffer from extreme graphical complexity

An active model is an approach that attends to these problems by presenting the right crosscutting structure at the proper 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".

Cottenier *et al*. (2007) introduced a new join point selection mechanism based on state machine specifications. The interfaces of a system include a specification of the effects of method invocations on the state of the module instance. This specification is not defined with respect to potential aspects, but unambiguously describes the observable behavior of the module. We have shown how a smart join point selection mechanism is able to infer points that might be located deep inside the implementation of a module, given pointcut that are expressed entirely in terms of its specification element. It refines the class diagram and the composite structure to capture the static structure of the system. It uses the state machine extension to represent the behaviors of the system.

Katara and Katz (2007) concerned architecture used to group aspect designs and can be seen as a software architecture viewpoint. The concern architecture model provides an aspect-oriented perspective on software design. The model can also be seen as an aspect analysis viewpoint for analyzing impacts of changes or trade-offs in concerns to be addressed by aspects. It adds some new stereotypes to model aspects such as <<Aspects>>,<<Concerns>>, <<Bind>>, <<replace>> and <<Unify>>.

Klein *et al*. (2007) proposed an Aspect-oriented UML approach using the standard UML. It did not propose any new notation as well as, it did not use the UML extension ability believing in keeping the

standardization as it is not changed. It is originally based on Message Sequence Charts (MSC) a standardized scenario language. It uses UML 2.0 sequence diagram. Indeed, no extensions to the UML Sequence diagram have been made; relatively a simplified Meta model for Sequence diagram has been designed, where conformity with the original UML Sequence diagram is accomplished through model transformation in the supplementary tool support.

Sharafi *et al*. (2010) presented an extension to the UML metamodel to explicitly capture the crosscutting concerns. It proposes an independent way from any programming language and cross platform. The newly created metamodel can be represented in standard XMI format, it does not have its own tool to draw the modeling; it uses the existing CASE tools to read this XML format. This language-independent aspectual description can support model transformations vital to software development and maintenance, such as forward engineering, reverse engineering and reengineering.

Aspect-oriented UML Modeling proposed an extension establishing a new package called AoUML, which consists of elements to represent the primary AO concepts such as aspect, advice, pointcut, parent declaration, introduction and crosscutting dependency (Przybylek, 2010). It also proposes reuse elements from the UML 2.1.2 infrastructure and superstructure specifications.

RESEARCH FINDINGS RESULTS

After applying the systematic review based on our RQ, we have compared the critical points and determined the mature approaches; we have illustrated our findings in the tabulated forms. The table lists all approaches (rows) and the criteria we are examining (columns). Each table comes with its index (legend) to elaborate the meaning of each abbreviation.

Table 5 illustrates the comparison between the selected approaches based on the language specific criteria. We have identified that none of the proposed approaches have used UML 2.4 and none of them was focusing on all UML diagrams, however, majority were focusing on class diagram (Structural modeling). Also, majority of the approaches did not propose any modeling procedure. Moreover, most of the approaches are neither tractable nor adoptable. While Table 6 shows that majority of the approaches have demonstrated their work with only a modeling example, few of them used a real life example demonstration, which makes only few of them usable.

Table 7 and 8 indicate that only few researches have demonstrated a complex demonstration example, some of them used a depreciated example such as, the compositor example, which is a well-known example for crosscutting representation. While Table 8 shows that none of the proposed approaches are consider as a complete framework, as none of them have proposed an extension to all UML language, apart from not using the latest UML edition. Moreover, none of them have provided a tool with the ability to generate AspectJ code, as well as model kernel extraction, at the same time none used to mode all AspectJ constructs.

Table 9 shows that, only few approaches have their own tool, the rest used an already implemented tool. And even those who proposed a tool did not work on having AspectJ code generator as well as Model extractor. The Table 10 clearly shows that none of the approaches have taken the full support for AspectJ constructs into

Table 5: Language criteria

	Language specification									
		EM				DT				
Approach name	UMLMLV	Meta mod	UML profile	LP	PD	Behavioral	Structural	MP	T	A
Clarke and Baniassad (2005)	1.x	Y	N	G	Y	S.D	C.D/ P.D	N	Y	P
Coelho and Murphy (2006)	2.0	N	N	G	N	N	C.D	N	N	N
Grundy (2000)	1.x	N	N	G	N	N	CO.D/ C.D	Y	Y	Y
Ho *et al.* (2002)	1.1	Y	N	G	N	N	C.D	T	T	Y
Jacobson and Ng (2005)	2.0	Y	N	G	Y	U.C/S.D/COMM.D	C.D/CO.D	N	Y	Y
Katara and Katz (2007)	2	N	Y	G	N	N	P.D	N	N	N
Klein *et al.* (2007)	2.0	N	Y	G	N	S.D	N	N	N	N
Pawlak *et al.* (2005)	1.x	Y	Y	S	Y	N	C.D	N	N	N
Reddy *et al.* (2006)	2.0	Y	N	Y	N	N	C.D/P.D	N	N	N
Stein *et al.* (2002)	1.x	Y	N	S	Y	N	C.D/CLL.D	N	P	P
Cottenier *et al.* (2007)	2.0	N	Y	G	N	S.D/ST.D	C.D/D.D	N	Y	A
Elrad *et al.* (2005)	1.x	N	Y	G	N	ST.D	C.D	P	P	N
Przybylek (2010)	2.2	Y	N	G	N	N	C.D/ P.D	N	Y	Y
Sharafi *et al.* (2010)	2.3	Y	N	S	Y	S.D	C.D	N	Y	Y

UMLMLV: UML modelling language version, EM: Extension mechanism, LP: Language purpose, PD: Platform dependency, DT: Diagram type, MP: Modelling process, T: Traceability, Y: Yes, N: No, C.D: Class diagram, P.D: Package diagram, P: Partial support, CO.D: Component diagram, U.C: Use case, COMM.D: Communication D, ST.D: State diagram, A: Adoptability, S.D: Sequence diagram, S: Specific, CLL.D: Collaboration D

Table 6: Maturity criteria

	Maturity issues		
Approach name	ME	ARWP	U
Clarke and Baniassad (2005)	Y	Y	N
Coelho and Murphy (2006)	Y	N	N
Grundy (2000)	Y	N	N
Ho *et al.* (2002)	N	N	N
Jacobson and Ng (2005)	Y	Y	Y
Katara and Katz (2007)	Y	N	N
Klein *et al.* (2007)	Y	N	N
Pawlak *et al.* (2005)	Y	Y	Y
Reddy *et al.* (2006)	Y	N	N
Stein *et al.* (2002)	Y	N	N
Cottenier *et al.* (2007)	Y	Y	Y
Elrad *et al.* (2005)	Y	N	N
Przybylek (2010)	N	N	N
Sharafi *et al.* (2010)	Y	Y	Y

ARWP: Application in Real-World projects, ME: Modeling examples, U: Usability, Y: Yes, N: No

Table 7: Real example

	Real example	
Approach name	UE	C
Clarke and Baniassad (2005)	N	N
Coelho and Murphy (2006)	N	Y
Grundy (2000)	N	N
Ho *et al.* (2002)	N	N
Jacobson and Ng (2005)	N	Y
Katara and Katz (2007)	N	N
Klein *et al.* (2007)	N	N
Pawlak *et al.* (2005)	N	Y
Reddy *et al.* (2006)	N	N
Stein *et al.* (2002)	N	N
Cottenier *et al.* (2007)	Y	Y
Elrad *et al.* (2005)	Y	N
Przybylek (2010)	N	N
Sharafi *et al.* (2010)	Y	Y

C: Complexity, UE: Used example, Y: Yes, N: No

Table 8: Complete framework

Approach name	CFS
Clarke and Baniassad (2005)	N
Coelho and Murphy (2006)	N
Grundy (2000)	N
Ho *et al.* (2002)	N
Jacobson and Ng (2005)	N
Katara and Katz (2007)	N
Klein *et al.* (2007)	N
Pawlak *et al.* (2005)	N
Reddy *et al.* (2006)	N
Stein *et al.* (2002)	N
Cottenier *et al.* (2007)	N
Elrad *et al.* (2005)	N
Przybylek (2010)	N
Sharafi *et al.* (2010)	N

CFS: Complete framework support, N: No

consideration, some have proposed a static representation for the pointcut some more have proposed dynamic approaches, however, none has represented as per the AspectJ. Finally, some have provided partial support for join point and advice type and none has presented all types of the join points and advice.

Table 9: Tools support

	Tool support		
Approach name	MS	CG	MKE
Clarke and Baniassad (2005)	Y	N	N
Coelho and Murphy (2006)	Y	N	N
Grundy (2000)	N	N	N
Ho *et al.* (2002)	N	N	N
Jacobson and Ng (2005)	Y	N	N
Katara and Katz (2007)	N	N	N
Klein *et al.* (2007)	Y	N	N
Pawlak *et al.* (2005)	Y	Y	N
Reddy *et al.* (2006)	Y	N	N
Stein *et al.* (2002)	Y	N	N
Cottenier *et al.* (2007)	Y	Y	N
Elrad *et al.* (2005)	Y	N	N
Przybylek (2010)	N	N	N
Sharafi *et al.* (2010)	N	N	N

MS: Modelling support, CG: Code generation, MKE: Model kernel extraction, Y: Yes, N: No

Table 10: AspectJ constructs

	AspectJ constructs				
Approach name	DCS	SCS	JPT	PCT	AT
Clarke and Baniassad (2005)	N	N	P	N	N
Coelho and Murphy (2006)	P	P	P	P	P
Grundy (2000)	N	N	P	N	N
Ho *et al.* (2002)	P	N	P	N	N
Jacobson and Ng (2005)	Y	Y	P	N	N
Katara and Katz (2007)	P	P	N	N	N
Klein *et al.* (2007)	Y	N	P	P	P
Pawlak *et al.* (2005)	Y	Y	P	P	P
Reddy *et al.* (2006)	Y	Y	P	P	P
Stein *et al.* (2002)	Y	Y	P	N	N
Cottenier *et al.* (2007)	Y	Y	P	P	P
Elrad *et al.* (2005)	Y	P	P	Y	P
Przybylek (2010)	P	P	P	P	P
Sharafi *et al.* (2010)	P	P	P	P	P

DCS: Dynamic crosscutting support, SCS: Static crosscutting support, JPT: Join points types, PCT: Pointcut type, AT: Advice type, N: No, Y: Yes, P: Partial support

The results administrated below have answered the research questions that we were focusing on. The RQ1 has focused on existing approaches; in the context we have investigated the well-established elicited approaches. The RQ2 talks about having a complete set of UML diagrams to model Aspects, based on the investigations we have proved that none of the existing approaches got a complete set of aspectual UML diagrams. The RQ3 focuses about modeling all AspectJ constructs and consequently, we have proved that, none of the existing modeling approaches have come out with modeling notations for all AspectJ. The RQ4 focuses about the usability of UML 2.4 in modeling aspect orientation. This question of systematical survey has been answered by depicting that, none have used the latest UML edition for modeling and none have investigated the infrastructure and the superstructure of the latest UML edition.

FUTURE WORK

As a future work, the ability of adopt Aspects modelling in the learning objects and e-Learning will be investigated due to the fact that e-Learning is becoming a major component in academia today (Jayanthi *et al*., 2007). Our future work is to provide a complete set of modelling notations consist of all UML diagrams to represent and model all AspectJ. To evaluate our modelling we will use different kind of evaluation techniques, we will try to study the ability to evaluate the AO using the metrics which has been used in object orientation (Parthasarathy and Anbazhagan, 2006) to come up with a reliable model as Reliability is one of the major concerns for software engineers (Chen *et al*., 2011).

CONCLUSIONS

After brain storming discussions, we had identified that almost all primary articles have similarities in their objectives and all of them are heading towards addressing and modeling the crosscutting concerns (Aspects) using UML, which is the focus of RQ1. Furthermore, majority of the studies were focused on extending the current UML model and make use of the extendibility feature of UML. Only few researches had proposed their own notations and extensions. The majority of the researches were focusing on one or two diagrams of the UML; none of them were addressing a complete framework. Moreover, majority of them were not providing tools for their propositions, they depend on the existing tool. Further, all the researchers had focused on the older versions UML than the current UML edition, which is UML 2.4. Moreover, none of the studies have focused on the new Infrastructure specification and the Superstructure specification of the adaptability and compatibility of UML 2.4 with the proposed model. We believe that Aspect-Oriented programming should be extended to the entire software development. Each aspect of the implementation should be declared during the design phase, so that there will be a clear traceability from requirements through source code.

ACKNOWLEDGMENTS

This study has been funded by the Faculty of Computer Science and Information Technology in University National Malaysia (UKM). This Aspect-oriented systematical review has inspired us and opened some new avenues for research. We are currently carrying on the research and have started on creating Aspectual UML 2.4 Modeling Approach, using the bottom up technique, to capture and represent all the crosscutting concerns (Aspect) constructs and elements including AspectJ as a base for the study.

REFERENCES

Albunni, N. and M. Petridis, 2008. Using UML for modeling cross-cutting concerns in aspect oriented software engineering. Proceedings of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications, April 7-11, 2008, Damascus, Syria, pp: 1-6.

Ali, J., 2010. Using Java enums to implement concurrent-hierarchical state machines. J. Software Eng., 4: 215-230.

Ali, N.H., Z. Shukur and S. Idris, 2007a. 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.

Ali, N.H., Z. Shukur and S. Idris, 2007b. Assessment system for UML class diagram using notation extraction. Int. J. Comput. Sci. Network Secur., 7: 181-187.

Araujo, J. and A. Moreira, 2003. An aspectual use case driven approach. Proceedings of the 8th Conference on Software Engineering and Databases, November 12-14, 2003, Alicante, Spain, pp: 1-5.

Asteasuain, F., B. Contreras, E. Estevez and P.R. Fillottrani, 2008. Evaluation of UML extensions for aspect oriented design. http://grise.upm.es/rearviewmirror/conferencias/jiisic04/Papers/7.pdf

Bass, L., P. Clements and R. Kazman, 1998. Software Architecture in Practice. Addison Wesley, USA., ISBN-13: 9780201199307, Pages: 452.

Changchien, S.W., J.J. Shen and T.Y. Lin, 2002. A preliminary correctness evaluation model of object-oriented software based on UML. J. Applied Sci., 2: 356-365.

Chen, L., W.U. Kaigui and H.E. Pan, 2011. Dynamic software reliability maintenance based on component monitoring and resource allocation. Inf. Technol. J., 10: 2214-2219.

Chitchyan, R., A. Rashid, P. Sawyer, A. Garcia and M. Pinto *et al*., 2005. Survey of analysis and design approaches. AOSD-Europe. http://www.aosd-europe.net/deliverables/d11.pdf.

Clarke, S. and E. Baniassad, 2005. Aspect-Oriented Analysis and Design the Theme Approach. Addison Wesley, Boston, USA.

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.

Cohen, S., J. Stanley, W. Peterson and R. Krut, 1992. Application of feature-oriented domain analysis to the army movement control domain. Technical Report No. CMU/SEI-91-TR-028, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA.

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.

Dahiya, D. and R.K. Sachdeva, 2007. Design issues in aspect oriented programming. Inform. Technol. J., 6: 879-884.

Dennis, W. and L. Bergmans, 2002. Using a concept-based approach to aspect oriented software design. Proceedings of the 3rd International Workshop on Aspect Oriented Software Development, April 22-26, 2002, Enschede, The Netherlands.

Elminir, H.K., M.A. Elsoud and A.M. EL-Halawany, 2011. Uml-based web engineering framework for modeling web application. J. Software Eng., 5: 49-63.

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.

Fazal-e-Amin, A.K.M. and A. Oxley, 2010. A review on aspect oriented implementation of software product lines components. Inform. Technol. J., 9: 1262-1269.

Groher, I. and S. Schulze, 2003. Generating aspect code from UML models. Proceedings of 4th International Workshop on AOSD Modelling with UML, October 2003, San Francisco, CA., USA .

Groher, I. and T. Baumgarth, 2004. Aspect-orientation from design to code. Proceedings of the Workshop on Aspect-Oriented Requirements Engineering and Architecture Design, March 21, 2004, Lancaster, UK.

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.

Jayanthi, M.K., S.K. Srivatsa and T. Ramesh, 2007. Learning objects and e-learning system: A research review. Inform. Technol. J., 6: 1114-1119.

Jose, H., F. Sanchez, F. Lucio and M. Toro, 2000. Introducing separation of aspects at design time. Proceedings of the 14th European Conference on Object-Oriented Programming, June 11-12, 2000, NY., USA.

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., 2002. Superposing UML class diagram. Proceedings of the Workshop on Aspect-Oriented Modeling with UML Model-Driven Development, September 30, 2002, Germany.

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

Kaur, D., P. Kaur and H. Singh, 2012. Secure Spiral: A Secure Software Development Model J. Software Eng., 6: 10-15.

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.

Letherbirdge, T. and R. Laganiere, 2001. Object-Oriented Software Engineering: Practical Software development using UML and Java. McGraw-Hill, USA., ISBN-13: 9780077097615, Pages: 497.

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.

Marco, M., C. Anis, S. Jaroslav and W. Jan, 2008. Applying and evaluating AOM for platform independent behavioral UML models. Proceedings of the AOSD Workshop on Aspect-Oriented Modelling, March 31-April 04, 2008, Belgium, pp: 19-24.

Muley, K., U. Suman and M. Ingle, 2010. Representing join point in UML using pointcut. Proceedings of the International Conference on Computer and Communication Technology, September 2010, Kerala, India, pp: 557-561.

Muller, A., 2004. Reusing functional aspects: From composition to parameterization. Proceedings of 5th Aspect-Oriented Modeling Workshop (AOM) in Conjunction with the UML, October 11, 2004, Lisbon, Portugal.

Omar A., A. Bader and T. Elrad, 2002. Weaving with statecharts. Proceedings of the Workshop on Aspect-Oriented Modeling with UML, April 22-26, 2002, Ensehede, The Netherlands.

Ossher, H. and P. Tarr, 1999. Using subject-oriented programming to overcome common problems in object-oriented software development/evolution. Proceeding of the 21st International Conference on Software Engineering, May 16-22, 1999, Los Alamitos, CA., USA., pp: 687-688.

Parthasarathy, S. and N. Anbazhagan, 2006. Analyzing the software quality metrics for object oriented technology. Inform. Technol. J., 5: 1053-1057.

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.

Qureshi, K. and P. Manuel, 2006. A survey of concurrent object-oriented languages (Cools). Inform. Technol. J., 5: 601-611.

Ramey, J. and P. Rao, 2011. The systematic literature review as a research genre. Proceedings of the Conference Seminar on Professional Communication, April 27-29, 2011, Portsmouth, UK.

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.

Sanatnama, H. and F. Brahimi, 2010. Graph drawing algorithms: Using in software tools. J. Applied Sci., 10: 1894-1901.

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.

Shaw, M. and D. Garlan, 1996. Software Architecture: Perspectives on an Emerging Discipline. 1st Edn., Prentice-Hall, Inc. Upper Saddle River, NJ, USA., ISBN: 0131829572.

Shen, W.H., N.L. Hsueh and P.H. Chu, 2011. Measurement-based software process modeling. J. Software Eng., 5: 20-37.

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.

Suzuki, J. and Y. Yamamoto, 1999. Extending UML with aspects: Aspect support in the design phase. Proceedings of the Workshop on Object-Oriented Technology, June 14-18, 1999, Lisbon, Portugal, pp: 299-300.

Uetanabara J., P. Parreira, A. Lazanha, R. Camargo and R. Penteado, 2009. A preliminary comparative study using UML-AOF-A UML Profile for aspect-oriented frameworks. Proceedings of the 8th ACM on Aspect-Oriented Software Development, March 2-6, 2009, Charlottesville, Virginia.

Wimmer, M., A. Schauerhuber, G. Kappel, W. Retschitzegger, W. Schwinger and E. Kapsammer, 2011. A survey on UML-based aspect-oriented design modeling. ACM Comput. Surv., Vol. 43, 10.1145/1978802.1978807

Yusof, M.M., Z. Shukur and A.L. Abdullah, 2011. CuQuP: A hybrid approach for selecting suitable information systems development methodology. Inform. Technol. J., 10: 1031-1037.

Zhang, G., 2005. Towards aspect-oriented class diagrams. Proceedings of the 12th Asia-Pacific Software Conference on IEEE Computer Society Engineering, December 15, 2005, Washington, DC., USA., pp: 763-768.

Zhang, J., Y. Chen, G. Liu and H. Li, 2009. An aspectual state model and its realization based on AOP. Proceedings of the WRI World Congress on Software Engineering, vol 3, May 19-21, 2009, Xiamen, Fujian, China, pp: 163-166.