Managing Early Aspects Interaction

Boubendir Amel

Abstract__ **Despite, the Aspect-oriented Software Development (AOSD) allows the separation of crosscutting concerns throughout the software life cycle and improve the modularity of software system artifacts, the complexity and diversity of interactions between aspects remind an important problem that AOSD community has to deal with, and has to support their manage by surely and fully catch the right conditions that involve their interaction in consistent way.**

In this paper, we present our approach which deal with many type of aspect interaction at requirement phase and follows a step by step strategy: from the concern level to the requirements level and analysis artifact level. Through this work we concentrate on issues of gathering and capturing clearly conditions necessary to be satisfied for a proper composition of the system.

Index terms__ AOSD, Aspect, Aspects Conflict type,Concerns, Interaction between Aspects,

I. Introduction

The Aspect-oriented Software Development is an emerging paradigm that complete and improve the current modern development approaches such as object-oriented and component-oriented approaches by providing a new mechanism for the separation of crosscutting concerns throughout the software life cycle in order to improve the modularity and maintainability of software system artifacts. The transversal concerns are then encapsulated in new modular units called "aspect"[13], and it provides as well a new composition technique for combining aspects and bases modular units named weaving. However, the complexity and diversity of interactions among aspects, and between aspects and base modules can reduce the value of the aspects oriented approach. So, it is essential to manage aspect interaction.

Up to now, existing aspect-oriented software development approaches have basically focused on dealing with aspect and their interactions at the programming level, such as works presented in [29]-[30]-[32]-[33] . Just lately, there have a few attentions that have been taken on managing their interaction at the early phases: analysis and design level [3]. Nevertheless, some works exist in the literature which undertake this subject in Aspect Oriented Requirement Engineering (AORE) such as in [12], where the authors try to diminish the time complexity of interactions among aspects, between aspects and base modules using Hamiltonian path techniques by locate the match point, order conflicting aspect, dominant aspect and generate the composition rules.

Manuscript received June 2019, revised June 2019.
Boubendir amel , Department of Computer Science
University of 20 Aout 1955 Skikda , Skikda, Algeria,
a.boubendir@univ-skikda.dz

In [34] S. Mohite et al. suggested analyze interaction and potential inconsistencies in requirement modeling. Through an use case driven approach , where the use case are refined by activities and are the join points , graph transformation system is used to weave aspect use case and to provide an analysis support for detecting order conflicts and dependencies. In the same way; in [5] Whittle et al. present an approach called MATA, based on model transformation. They provide support for conflict and dependency detection, based on critical pair analysis of weaved system. The objective of this detection phase is to order composition. More recent work by Chitchyan et al. [4] shifts the highlight towards a semantic analysis of requirements. Requirements here are annotated and composition rules can be expressed using semantic queries. This approach aim of removing some order conflicts automatically detected. In [19] the authors propose a FTS approach witch deal with aspects interactions and dependencies in analysis phase. The approach identifies aspects using Colored Petri nets and exploit dependencies resulted of using operators such as Before, After, and Replace which is consumed by their framework, for generating a composition rule for every match point using the specification of aspect. The approach incorporates a feedback edge set, topological ordering, and second valid ordering. In [6] an AORE approach called MDSOCRE based on XML syntax, it allows the organization of requirements into concerns. Concerns can be connected by means of compositions rules which express crosscutting relationships of aspects in requirement level granularity. However in [20] A. Alberto et al. concentrate rather on identifying and resolving conflicting dependencies between aspects in textual requirement, they present an automated tool EA_analyser to detect conflict on requirement specified in natural language.

Even so, from our literature review we have observed so far, a considerable lack of works that deal with different issues of aspect interaction analysis and management. From these important issues, which are less investigated, we underline the importance to deal with different aspect interaction type more than their classification and the issue of formulating sure conditions that has to be satisfied in implementation phase.

Therefore, through this paper we refine and extend our ideas on dealing with aspect interactions. We will present an extended multi-level interactions approach to deal with interaction between aspects in the requirement engineering phase, from the concern level (use case) through the requirements level (scenario and step of scenarios of use cases) to arriving at the analysis diagrams specification. We will concentrate on aspect condition identification issue and initiate also to a certain treatment process of the conflicts and multi-level interactions analysis during the early phases.

The rest of the paper is organized as follow: In section 2, we give a background about aspects interaction and conflict problems; we give some classifications as proposed in some research works. In section 3, we discuss how dealing with aspects interaction problem and we discuss issue of capturing aspects conditions to motivate our new approach. Our approach is described in section4. Once, the section 5 gives some related works, and finally concluding remarks are drawn from the work and perspectives.

II. ASPECTS INTERACTION AND CONFLICTS PROBLEM

The problem of interaction and conflicts between aspects is a very serious problem that occurs throughout all software development phases. An interaction has been recognized as the application of multiple aspects at the same join point which can yield to undesired effects during program execution [6]. However, a conflict captures the situation of semantically interference: one aspect that works correct in isolation does not work correctly anymore when it is composed with other aspects [5].

There are some works which cover in an explicit way these problems, most of those give classifications of them: According to [7], an aspect is able to interact with other components by altering mainly the static structure of both code or system or the control flow or by the state of the objects .In [9], the authors distinguish two types of conflict, control and data related conflicts. Where the first models the effect of advice on the control flow and the latter captures conflicts that occur due to shared data. According to [8] Bakre et al. classify interactions among aspects and base program into Spectative aspects that observe the state of the system at join points, but they do not control the execution flow. Regulative aspects which observe the state at join points and control the flow at join points based on the state, and Invasive aspects, which in addition to be regulative modify the system state at join points. Globally, all the previous classifications are too much oriented to aspect programs. However, the following classification is more complete than the previous ones: according to [10], we distinguish four principals' categories for interactions:

The first one is the transverse specifications in which the current use of join points for specifying aspects and theirs locations where they are to be inserted, can lead to two problems: the accidental join points and accidental recursion. The first problem where the capture behavior aspect is accidentally inserted bad and undesirable locations (Join point). However, accidental recursion refers to the situation where the behavior of the aspect itself corresponds to a join point specification leading to recursion. The second one is aspect-aspect conflicts: also called interaction aspects, this problem occurs when multiple aspects coexist in a system, and here they identify five types of interaction: Conditional execution, Mutual exclusion, Conflict of order, Conflict of a nature depend on the dynamic context and negotiation of the conflicts on the requirements and architecture level (tradeoff). Conditional execution is the case when the application of an aspect depends on another aspect that must be applied for its correct functioning. The two last ones are Base-aspect type conflicts and concern type conflicts. Conflicts Type Concern Occur when concerns affect the comportment or status of other concerns. After all, there is a last classification which is more general. According to [2], Aspect interaction occurs when several aspects coexist in a system. The authors distinguish between four different types of aspect interactions: mutual exclusion, dependency, reinforcement and conflict.

A. How dealing with aspects interaction problem?

From the foregoing, we can assure that the aspect interaction treatment process is very difficult, and that problem can be occurring and discussing in tree essential points:

- The aspect itself can occur in complex state.
- There are different types of conflicts and interactions.
- The conflicts and interaction problem affects all aspect oriented software process.

From the first point, an aspect itself may be the locus of further complexity. Aspects can be homogeneous or heterogeneous [11], it is heterogeneous if it implements different advices for many join point's specification. And then, we need a solid analysis for defining theirs behaviors and theirs interactions, and we need to formulate a strict constraints and conditions to be satisfied.

When from the second point, the different types of conflicts require to establish a clear process for treating different types of interactions in order to lead to formulate conditions and constraints list that must be never contradictory later.

However, from the third point, we can recognize that the aspect and the aspect interaction and conflict problems are occurred in all life cycle process. Certainly, it is favorable to define aspects early in the development process during the early stages of software development, requirements analysis, domain analysis and design phase architecture; that improves the aspect-oriented development.

Thus, it is very important to formulate conditions and constraints which must be satisfied later in the following stage of life cycle.

Other more, we think that's important to adopt a multi-level identification and definition of constraints and conditions which have to be satisfied in all following stages owing to the diversity and complexity of the problem itself.

B. Aspects interaction conditions identification issue.

E. Pulvermuller et al in [27] argued that there were several issues to be considered with respect to conditions that had to be satisfied: The classification, Detection, collection, expression, storage and evaluation of conditions and each was an area of research by itself [27]. We are interesting here to detect and collect conditions. We will first explain the relationship between aspect requirement (early aspect) and aspect implementation level as like as it has been described in several works like [17]-[18]-[26]-[27]. Then we will discuss some advantages and problems.

It is well known that aspect in requirements level will be dispersed in set of implementation unit in implementation level such as classes and methods and then the implementation of aspect requirement may interact to gather because an implementation unit may implement many aspect requirements.

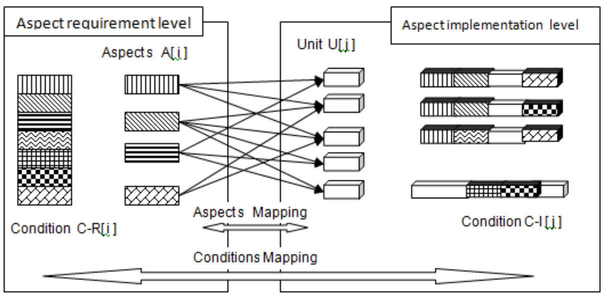

Fig.1 Relationship between aspect requirement and implementation level

It is important to capture relevant dependencies and conditions in the aspect requirement level, because the aspect implementation must satisfied all specified conditions connected to its specified requirement.

We agree with authors of [27] while they suggested that due to the fact that it is impossible to capture all relevant dependencies and conditions from beginning. Additional conditions were added as needed or detected in growth manner. Furthermore, we think that is a multi level analysis of interaction leads to capture sure and precise conditions which have to be satisfied in implementation level and we think that is the definition of condition in analysis artifacts is more suitable since it is the closest level to conception and implementation level and then the mapping of specified condition at this level to conception and implementation level is easier and more interpreted. However, this level is too finer that it is impossible to identify and deal with aspect interaction correctly owing to the diversity and complexity of the problem itself. Consequently we believe that a multi level analysis which adopts a step by step constraints specification strategy can help us to deal with this complex situation.

III. OUR APPROACH

In our work, we concentrate on the requirements analysis phase, from de requirement definition until requirement analysis since it is the earliest stage in life cycle process and the aspect oriented requirements engineering (AORE) proclaims the advantage of the early dealing with aspect interaction and conflicts for all development process phases. We focus on managing aspects interactions and dealing with some types of aspects conflicts which are:

- tradeoff of the conflicts on the requirements and architecture level
- accidental recursive conflicts
- identification of order conflicts
- dependency or conditional execution

That permits us to capture and formulate sure conditions in well established way. Our proposal is an extension of the approach presented in [28], However in this work we expand our treated type aspect conflict list to deal with tradeoff of the conflicts on the requirements and architecture level and dependency (or Conditional execution). Likewise, we initiate to manage interaction through the analysis artifacts.

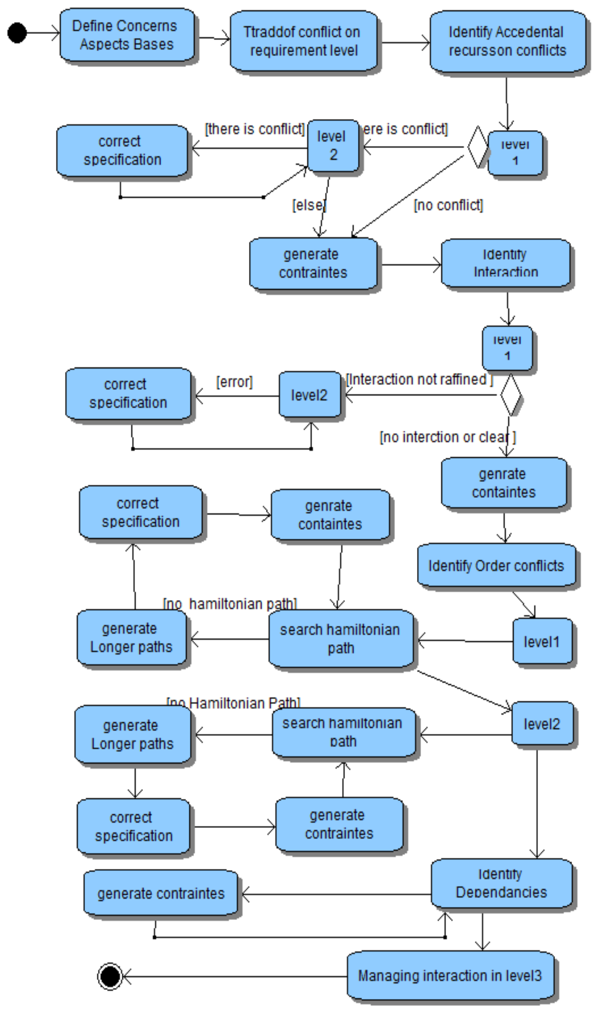

Fig 2. Overall process of our approach

The process of our proposal is illustrated by figure2. It comprises six stages, namely:

- Concerns, bases and aspects definition.
- negotiation of the conflicts on the requirements and architecture level (tradeoff) : level1
- Identification of accidental recursive conflicts: level 1 and level 2.
- Identification of interactions: level 1 and level 2.
- Identification of order conflicts: levels 1 and 2.
- Dependency and Conditional execution : level 1 and level2
- Managing interaction on analysis artifact : level3

This process is resumed in two principal activities: concerns definition and interactions analysis.

A. *Concerns definition*

During this activity, we define the concerns, aspects and bases as it was proposed in many aspect requirement approaches like [17]-[21]-[14] for us we adopt a generic use case template to specify theme. An aspect is a concern which could be either a base or an aspect, and this last may be a base concern for other aspects. It could crosscut many bases concerns and the base concern could be affected by many heterogeneous aspects. Furthermore, an aspect concern could have many behaviors which are applied in different localizations. It is the most generic and most difficult case, which mainly maintains more liberty and more aptitude to specify aspect oriented software. Concerns (aspects and bases) definitions are separated in separate template, but this is not enough to keep them really separated. It is well-known they influence each other either positively or negatively. And thus their influence relationship must be described and well managed

We exploit the matrix (tab1) described in [22] to link concerns and represent the symmetric influence relationship between them.

Table I
Matrix of link concern-concern

	Concern1	Concern2	…	Concern n
Concern1	▨	√		√
concern2	√	▨		√
			▨	
Concern n	√	√		▨

And we use also our proposal matrix (tab2) which is a specialization of matrix (tab1) that describes the crosscutting relationship between aspect and base in a high level of abstraction and which could be also viewed as an improved version of the matrix described in [21] .

Table II
Matrix aspects - bases

	Aspect1	Aspect2	……….	Aspect $_N$
Base1	▨			√
Base2	√	▨		√
			▨	
Base $_N$	√	√		▨

These matrixes (tab1 and tab2) are very useful; we can exploit them for an early detection of conflicts. The matrix (tab1) is very suitable to identify tradeoff of conflicts on the requirements and architecture level; in contrast our matrix (tab2) is suitable to capture aspects bases conflicts and interactions.

B. *Interactions Analysis*

In this activity, we focus on managing and analysis interaction between aspects and some types of conflicts between them.

First we start with negotiation of the conflicts on the requirements and architecture level, next we deal with accidental recursion conflicts type identification then we identify interaction between aspects and we deal with order conflicts and dependency or conditional execution. Finally we deal with interaction in analysis artifact.

C. *negotiation of the conflicts on the requirements and architecture level*

Negotiation of the conflicts in the requirements and architecture level are namely the most treated conflict type in requirement analysis phase. It is through positive and negative influences that almost early aspect approaches manage and reason about conflicts such as in works [21]-[22]-[23]-[24]-[25].

In our approach we propose that this type of interaction should have managed in first step and in only concern level. We take support of matrix tab1 which is more suitable to represent in an early step any identified influence between concerns. And we employ also the matrix of contribution (see tabIII) which point out positive and negative influences between concerns since it is the most technique currently used. When there is a negative influence, we usually use concern assigned weights and negotiation to resolve conflicts such as weight expresses priorities [21]-[22].

Table III
Matrix of contribution

	Concern1	Concern2	…	Concern n
Concern1	▨	.+		-
concern2	+	▨		+
			▨	
Concern n	-	-		▨

Nevertheless, we assume that priority negotiation is a hard task in which we can update priorities incorrectly. And so we propose carry out Tradeoffs shift our analysis to deal with other conflict types and manage them in several levels, in view of the fact that assigned weights are usually useful in aspect order conflicts identification and resolution. Consequently, Assigned priorities are fixed in gradual way and priority constraints are specified.

D. *Accidental recursion conflicts types identification:*

The figure 3 presented in [28] shows an example of recursion problem situation even as there is not a really recursion. The example is about three use cases where the first one crosscut principal scenario of the second with a principal scenario, the second crosscut the principal scenario of the third with another principal scenario and the third one crosscut the principal scenario of the second use case with an alternative scenario [28]. In concern level we detect one cycle between use case 2 and use case 3, in contrast in the requirements level there is not a real cycle seeing as we do not weave the same behaviors of use case 2 and use case 3.

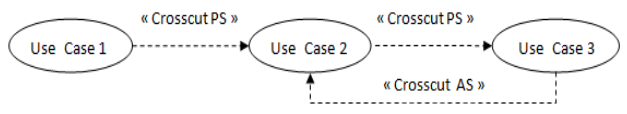

Fig 3. The recursion problem example

To deal with recursion problem we propose keep all crosscut relationship on a crosscut graph Cros=(X,U) which its X set of nodes includes the list of aspects and bases concerns and its U set of arc holds all the existing relation of crosscut and then looking for all cycles in this graph .

This matrix is employed in both concern and requirement levels. In contrast of the concern level; the last level detects recursive conflicts in a more detailed granularity level (scenario and step of use case scenario). If there are no recursive conflicts in the first level of concern, we should specify constraints to be respected; otherwise we shift on to the next level to check whether the cycle problem found in the previous level is real or not. In the case of conflicts the specification has to be corrected otherwise we should generate constraint.

E. Interaction identification:

We have to identify the problem of interaction between aspects, in each join point. We suggested using the match point matrix described in [14] provided that it identifies interactions between aspects into the two levels concern and requirement. In instance where there is one aspect in any match point, it means there is no conflict or aspect-aspect interaction, knowing that the problem of aspect-base interaction has been already resolved in the accidental recursive conflict detection phase. Otherwise, if there are several aspects in a one match point, it is a problem of interaction. The interaction can be positive or negative: the dependencies are positive interactions and conflicts are negative ones. There are several sorts of conflict that may occur and which needs to be handled by developers, such as order conflict, accidental join points and conflicts type concern. This task could be obviously critical and hard. In spite of that we believe that our step by step strategy allows us to reduce trouble of mission and carry out a good mastery of aspects interactions. We have to generate conditions to be respected. If not, we pass to the next level where others refined conditions can be formulated and corrections on the specifications can be done. And anyway we expect that all identified interaction condition should be made and extended with other conditions based on deal with join point class type interaction. So far we would concentrate in the next sections only on two type of interaction: aspect order conflict and aspect dependency.

F. Order conflicts identification:

The purpose here is to identify multi-level order conflicts. To pull off this goal, we pass through a step by step strategy: from the concern level at the requirements level. Similarly to [1, 12], we use the generic technique described in [1] which diminish the time complexity analysis of conflict and interactions among aspects

This technique exploits the initial generated dependencies graph and the look for Hamiltonian paths in this graph, if not the generation of longer paths. This technique has to be used for the two levels.

G. Identification of dependencies:

The identification of dependencies or conditional execution is very close to order conflicts handling ever since its resolution specifies order conditions which catch clear dependencies. In our approach dependencies are managed through generation of the fulfilled dependencies graph presented in [1]-[12].

However, the construction of such graph is a critical task. The identification and the reasoning about correct dependencies, is not easy. We classify dependencies to dependencies inferred from weaving operators and resolution dependencies introduced incrementally to resolve order conflict. The dependencies generated from weaving operators reflect the aspect oriented concepts and therefore, they can provide an ideal starting point for identifying convinced dependencies and allows a guided incremental dependencies analysis process. We can prohibit the introduction of a wrong dependency, as we can force the system to respond with the appropriate behaviour by identifying and specifying the sure dependencies.

H. Managing interaction on the analysis artifact:

In this section we outline a transition from an aspect-oriented use case model towards an aspect oriented analysis model, and initiate to deal with aspect interaction in this fin level. For dealing with aspects which have internal structure and complex behavior and dealing with their interaction, we support a use case driven approach which encapsulates these aspects as use cases as like as [26]. Here, the transformation between requirements, analysis, and design can be made perfectly. In requirements modeling we concentrate on defining rigorous use case models. In contrast, in analysis modeling we ponder to look for objects witch collaborate to realize the specified use case scenarios, and to model them with UML diagrams.

We assume that is in analysis artifact the more aspects are available the more interaction analysis is complex and the more errors on their composition are produced. Beyond the definition and modeling of aspects, a further problem arises: How could be an aspect specification verified and how could the correctness of their mutual interactions be proved? [26]-[27] And especially witch conditions must be expressed to be satisfied.

So we can reason about concerns (base and aspect) as a use case that each of them specifies a set of condition. Then requirement of any concern (scenario, step of scenario) has to satisfy their first level conditions (concern level). And could refine their concern conditions or could add specific requirement conditions that relate to specific requirements. Finally, each of analysis unit inherits all conditions of concerns and requirement that implement. Conditions specified on analysis artifact then have to refine the previous level constraints and specify them based on the units involved to each concern and requirement. For example let suppose that A1, A2 two aspect concern that crosscut the Base concern (B). in concern level it is specified that A1 and A2 has to be weaved before the base B a constraint of priority has be identified (A1 prior than A2). Then we can refine in requirement level to specify for example all requirement off aspect A1 are prior to all requirement of aspect A2 and in last in analysis artifact we specify that method invocation which start A1 execution (is captured as advice) is prior than the method invocation that start execution of A 2 . This is the simpler case of interaction other complex case can occur.

IV. RELATED WORKS

A. The aspect-oriented development approach with use cases (AOSD/ UC):

AOSD/UC is one of the most important aspect oriented methods; it is not only an analysis aspect-oriented requirements method, but an aspect-oriented development method that covers all the development process. In principle, this approach suggests that the use cases are crosscutting concerns, in consequence their realization affects several classes [15]-[17]. The AOSD/UC approach process in requirements engineering is very similar to the traditional use cases oriented approaches process. About the treatment of conflict; the method does not support the treatment of interactions. Identification and resolution of conflict between basic and feature use case, it is left to developer who must identify conditions to meet and write a rule specification that satisfies composition [15]. And as the approach assumes that the use cases are aspects that cut crosscut classes, the problem of interaction is only observed in the implementation level and does not consider the transverse effect of concerns during requirements analysis [15]. These concerns not only would be certainly crosscutting at the implementation stage but they will surely impose conditions and constraints in all phases owing to their interactions. In our work we take into account the transverse manifestation of concerns during the requirements phase. The disregard of their interaction handling yields to very hard specification of composition rules.

B. The ARCADE approach :

The objective of arcade approach described in [17] is the modularization and composition of crosscutting concerns at the level of requirements. The approach is in essence based on the point views. The concrete realization of the approach is carried out through the exploit of well-defined XML-based language models which let the definition of composition rules and enable the analysis of the composed specification to establish possible points of compromises. The approach deals with aspects conflicts. The identification of conflicts between candidates aspects is based on the build of the contribution matrix which indicates how (positively or negatively) an aspect contributes to others aspects, and also by assigning weight to the cells of the matrix. The assigned weights are employed to resolve conflicts. If two aspects contribute negatively to each other and have the same weight, a negotiation is necessary among stakeholders, to solve the problem [17]. So; although this approach addresses early the interactions problems, we can perceive several points of difference compared to our approach. The approach anticipate the advantage of a fine analysis (the requirement level) but surly not by step by step strategy analysis, it is just for avoiding unnecessary negotiations. However the purpose of our step by step analysis strategy is to gather and specify mandatory constraints to be satisfied in the following phases. Moreover, the approach focuses on a single type of conflict and does not treat different types of conflict. In our approach we anticipate the dealing of different types of problems.

C. Theme/UML approach:

The Theme/UML approach is an aspect-oriented analysis and design approach that tends to identify and model a wide range of aspects early in the life cycle of software [16]. It provides support for aspects oriented development, in requirements through the different views of Theme/Doc and crosscutting behaviors and design level via Theme / UML [16]. About the interactions treatment, as the approach does not compose Aspects at the requirement phase, the approach does not provide explicit support for the early treatment of aspects interaction problem during this phase [16]. Moreover the approach does not consider other types of conflict problem that may occur. Our work can enhance this approach. We take into account the treatment of several types of conflicts and emphasize their early treatment. We believe this not only helps maintain aspects traceability but also maintain traceability of solutions.

D. Version Model for Aspect Dependency Management:

The approach presented in [27] is a version model that concentrated on aspect dependency problem. This work was motivated by the fact that it is impossible to capture all relevant dependencies and conditions from beginning additional conditions have to be added during composition process as needed. May be the approach highlight the importance of capturing true condition, we can perceive several points of difference compared to our approach. This approach do not precise how to capture condition, they namely indicate that they appeared during the problem analysis and design of a system [27] . Yet, in our work we start to a well established process that captures conditions, we deal too with several type of conflict by step by step strategy and before any composition of system.

V. CONCLUSION

Despite, the Aspect-oriented Software Development (AOSD) allows the separation of crosscutting concerns throughout the software life cycle ,the complexity and diversity of interactions between aspects remind an important problem that AOSD community has to deal with, and has to support their manage by surely and fully catch the right conditions that involve their interaction in consistent way.

Just lately, there have a few attentions that have been taken on managing their interaction at the early phases [3]. In this paper we have refined our multi-level approach to deal with interaction between aspects in requirement engineering phase. The aim of our work was both the contribution to the built of multi-level interactions and conflicts identification process and to handle many types of conflict identification. By the way, we could by step by step strategy from the concern level (use case) through the requirements level to analysis artefacts, capture in clearly the sure constraints and conditions necessary to be satisfied for a proper composition of the system in the implementation level and offer a good mastery of interactions too.

This work is not yet the last step towards a multi-level management of interactions between aspects. Our future work will focus on developing a sophisticated support for this approach, and improving it by treating other aspect conflicts type and refined the third more detailed level of granularity (level of analysis class and objects) to present how identifying more interaction problems, and dealing with them . And we will focus on constraints formulation and how specifying them.

Reference

[1] A. Boubendir, A. Chaoui," Towards a generic technique for analysing interactions between aspects at requirement phase". ICDIM 2010 : 507-512.

[2] F. Sanen, E. Truyen, W. Joosen, A. Jackson, A. Nedos, S. Clarke, N. Loughran, A. Rashid, "Classifying and documenting aspect interactions".workshop on Aspectscomponents and patterns for infrastructure software at AOSD,2006.

[3] The Aspect-oriented Software Architecture Design portal: Http://trese.cs.Utwente.nl/taosad/aosd.htm

[4] R. Chitchyan, A. Rashid, P. Rayson, R.Waters. "Semantics-based composition for aspect-oriented requirements engineering". Proceedings of the 6th international conference on Aspect-oriented software development, pages 36{48, New York, USA, 2007.ACM.

[5] J. Whittle and P. Jayaraman. "MATA: A tool for aspect-oriented modeling based on graph transformation". Workshops and Symposia at MoDELS 2007, volume 5002 of Lecture Notes in Computer Science, pages 16{27. Springer, Berlin / Heidelberg, 2008.

[6] A. Zambrano, '' Addressing Aspect Interactions in an Industrial Setting: Experiences, Problems and Solutions'', Tesis presentada para obtener el grado de Doctor en Ciencias Informaticas Facultad de Informatica - Universidad Nacional de La Plata, Marzo de 2013.

[7] M. Bernardi, G. Di Lucca, "A Taxonomy of Interactions Introduced by Aspects", IEEE International Computer Software and Applications Conference, Italy, 2008.

[8] S. Bakre, T.Elrad."Scenario based resolution of aspect interactions with aspect interaction charts". In Proceedings of the 10th international workshop on Aspect-oriented modeling, AOM '07, pages 1-6, New York, USA, 2007. ACM.

[9] P. Durr, L. Bergmans, M. Aksit "Static and Dynamic Detection of Behavioral Conflicts Between Aspects", in sokolsky o, Tasirans, RV, 2007, Lncs,vol 4839,pp 38-50, springer .

[10] J.Hannemann, R.Chitchyan, R.Awais, "Analysis of Aspect-Oriented software", AAOS, Darmstadt, Almagne, 2003.

[11] C. Chavez et al,"Crosscutting interfaces for aspect-oriented modeling", journal of Brazilian Computer Society, 2006 volume12.

[12] K. Santhi, G.Zayaraz, V. Vijayalakshmi, ''Diminution of Interactions among Aspects at Requirement Phase using Hamiltonian Path Technique'', Research Scholar/CSE, Department of CSE, Department of ECE Pondicherry Engineering College, Puducherry, 2012.

[13] AOSD homepage, HTTP://WWW.AOSD.net

[14] I. Brito, A. Moreira, ''Towards a composition process for aspects-oriented requirements'',in EA workshop, Boston,USA,2003.

[15] R.Chitchyan, A.Rashid, P.Sawyer, A. Garcia, M.P. Alarcon , J. Bakker, B.Tekinerdogan, S.Clarke, A.Jackson ,"Survey of Analysis and Design Approaches", Report of the AOSD-Europe -Network of Excellence on AOSD , 18 May 2005.

[16] E. Baniassad , S. Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design," International Conference on Software Engineering, 2004.

[17] I. Jacobson , "use case and aspect-working seamlessly together", journal of objecttechnology, vol 2: 7-28, 2003.

[18] J. Araujo, E.Baniassad, P.Clements, A.Moriera, A.Rachid, B.Tekinerdogan, "Early aspect: the current landscape", Technical Report , Lancaster university February, 2005 .

[19] K.Shanthi,G.Zayarar,V.Vijayalakshmi,"Resolving Aspect Dependencies for Composition of Aspects", Arab J SCi Eng (2015) 40: 475.doi;10.1007/s13369-0141454-3

[20] A.Shardinha, A,Christchyan, J.araujo, A.Morira, A.rachid, (2013), "conflict identification with EA-analyzer", aspect oriented requirement engineering, springer, berlin, heidbelberg.

[21] A. Rachid, A.Moreira and J.Araujo "Modulaisation and composition of Aspectual Requirements". In 2and International conference on Aspect Oriented Software Development (AOSD). 2003 Bostan, USA: ACM.

[22] A. Moreira, J. Araujo, and A. Rashid, "A Concern-Oriented Requirements Engineering Model," presented at Conference on Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal, 2005.

[23] A.Amirat, "Towards a requirements model for crosscutting concern", information tecnnology journal Vol.6(3):332-337, asian network for scientific information , 2007.

[24] A. Moreira, J. Araujo, and A. Rashid, "Multi-Dimensional Separation of Concerns in Requirements Engineering," presented at Requirements Engineering Conference (RE 05), 2005, France,.

[25] I.Brito, A. Moreira," Integrating the NFR framework in a RE model", In proceedings of the 3rd Workshop on Early Aspects, 3rd international conference on Aspect-Oriented Software Development, March 2004.

[26] S. Herrmann, C. Hundt, K. Mehner," Mapping Use Case Level Aspects to ObjectTeams/Java", in workshop on early aspects,2004: aspect-oriented requirements engineering and architecture design, in conjunction with OOPSLA conference, Canada,2004 .

[27] E. Pulvermuller, A. Speck, J. O. Coplien, "A Version Model for Aspect Dependency Management" , In Proc. of 3rd International Conference on Generative and Component-based Software-Engineering (GCSE 2001), Springer, LNCS 2186, pages 70-79, Erfurt, Germany, 2001

[28] A. Boubendir, A. Boudeffa, "A multi-level interaction dealing approach with aspects in requirement engineerib phase",in engenering &MIS(ICMIS), IEEE, Maroco, 2016

[29] N.Weston, F.Taiani, A. Rashid, "Interaction Analysis for Fault-Tolerance in Aspect-Oriented Programming", in procedeeng of 2007 workshop on methods, models, and tools for fault tolerance

[30] R.douance , P,Frader, " detection and resolution of aspect interactions" ,INRIA technical report N°RR 4435 April 2002

[31] S.Mohite, R.Phalnikar, M.Joshi, S.D.Joshi, S.Jaldav, "Requirement and interaction analysis using aspect-oriented modeling", Advence computing conference(IACC),2014.

[32] E. Katz, S Katz, W. Havinga, Tstaijen, N. Weston, F.Tiani,R.Awis, Dong Ha Nguyen , M. Südholt, " Detecting Interference among Aspects ", Report of the AOSD-Europe -Network of Excellence on AOSD 18 february 2007,

[33] W. Havinga , I.Nagy , L. Bergmans , M. Aksit: "A graph-based approach to modeling and detecting composition conflicts related to introductions". AOSD 2007 : 85-95