A Preliminary Comparison Framework for Aspect-Oriented Modeling Approaches

José Uetanabara Júnior[1] and Valter Vieira de Camargo[2]

[1]Fundação de Ensino "Eurípides Soares da Rocha"
Centro Universitário "Eurípides de Marília" – UNIVEM
Marília, São Paulo, Brasil - CEP 17525-901

[2]Departamento de Computação da Universidade Federal de Lavras (DCC-UFLA)
Campus Universitário, Lavras, Minas Gerais, Brasil - CEP 37.200-000
miklotovx@gmail.com, valtercamargo@hotmail.com

Abstract. Several proposals to model aspect-oriented systems can be found in literature. However, there is not an approach which emerges as "the most suitable" for any kind of projects. In this context, it is interesting to provide tools for the software engineers which guide them for the choice of a modeling approach that best fit with their project requirements. So, a preliminary comparison framework is presented on this paper. This framework contains four criteria and allows the assignment of weights for each criterion in order to customize the framework. The weights provide flexibility to the framework since the software engineer can assign values according to the project requirements. Four modeling approaches were compared using the framework: Clarke and Baniassad, Stein *et al*., Pawlat *et al*. and Groher and Baumgarth. The framework pointed out Stein´s approach as the most appropriate when all the criteria have the same weight.

Keywords: Aspect-Oriented Modeling, Comparison among Aspect-Oriented Modeling Approaches.

1 Introduction

Unlike from other aspect-oriented languages which AspectJ [7][10] emerges as the most disseminated, the same does not happen with the aspect-oriented modeling approaches. Currently, there is not a modeling approach for aspect-oriented programming which arises as the most appropriated. So any initiative to support the software engineer in selecting a modeling approach is a great contribution. Thus, in this paper a preliminary comparison

[1] This research is supported by CAPES.
[2] This research is partially supported by CNPq: Projeto Latin-AOSD - PROSUL-Proc. no. 490478/2006-9.

framework is presented. The aim is to provide criteria for conducting a comparison among modeling approaches. The proposed framework allows the software engineer to assign weights to criteria according to the characteristics of a certain project. Also, the comparison framework can also act as base for software engineers to be able to compare other approaches that are not being mentioned in this paper.

To exemplify the framework application, we show a comparison among four modeling approaches that were applied to the same order system. The unique crosscutting concern employed to this system was Trace. We have adopted this crosscutting concern because it is simple and much known in literature. Also, since the comparison framework is in its initial version, we believe that a more complex concern could obscure the analysis. The approaches used to exemplify the framework application were proposed by the following authors: Pawlak *et al.,* [11], Clarke and Baniassad [5], Groher and Baumgarth [8] and Stein *et al.* [12]. There are several other modeling proposals [1][2][3][6][9][13] available in the literature, but only those four were used in the comparison.

The framework consists in four criteria; two of them were obtained from the AOSD-Europe-ULANC-9 report [4] and the other two were proposed by the authors of this paper. It is highlighted that the comparison framework does not point out an approach as the most appropriate for all cases, once the choice of a certain approach is completely dependent on characteristics of the project in which the approach will be employed.

This paper is structured as follows: in Section 2 the comparison framework with its criteria is presented; in Section 3 the approaches and the case study are presented; in Section 4 the framework application is made and in Section 5 conclusions and future workers are presented.

2 Comparison Framework

In this section the comparison framework is shown. Four criterion where used, which two of them were obtained from the AOSD-Europe-ULANC-9 report [4]. They are "traceability" and the "scalability". We argue that the two criteria obtained from the AOSD-Europe-ULANC-9 report [4] are not enough to guide the developer in selecting a modeling approach. Thus, two other criteria are being proposed to verify the notation representativity, which are "aspect representativity" and "crosscutting relationship representativity". All criteria can be classified as "low", "medium" or "high".

The criterion "traceability" determines if the traceability level of the approach. This criterion determines if the artifacts can be mapped/traced during the development phases: analysis, design and implementation. The larger the capacity to maintain a mapping among the artifacts that are created during the development, the higher the traceability will be; characterizing it as "high".

The criterion "scalability" determines the scalability level of the approach. The "scalability" of an approach is related to its legibility as the number of components/classes of the system grows. The approach should be appropriate for both large and small

systems. The classification of an approach regarding its "scalability" can be low (easy to model a system with few components and difficult to model systems with many components), medium or high (easy to model systems with few or many components).

The "aspect representativity" measures the clearness degree in which is possible to identify the basic components of an aspectual module and its own presence. This criterion aims to determine if: 1) it is easy to visualize and to identify the presence of one or more aspects; 2) it is easy to visualize and to identify the presence of one or more pointcuts and 3) it is easy to visualize and to identify the presence of one or more advices. If the three items can be identified, the criterion is classified as "high". If only two of them can be identified the criterion is "medium" and if only one is identified the criterion is "low".

The "crosscutting representativity" measures the clearness degree that is possible to visualize/identify the crosscutting relationship details. This criterion aims to determine if it is possible to visualize/identify: 1) that an aspect affects one or more classes; 2) exactly the methods affected by an aspect; 3) if the method is affected by the aspect before or after its execution/calling and 4) if the method is affected on the call or execution context. If the four items are satisfied, the criterion receives a "high" score. If only three or two are satisfied the criterion receives a "medium" score, otherwise it is classified as "low".

In our framework, the values "high", "medium" or "low" are associated with numbers to make the comparison based on a numeric base. Therefore, to "high" was assigned the value 3; to "medium" was assigned the value 2 and "low" received the value 1. These grades were assigned to ease the comparison, but they do not harm it, once all of the approaches are compared in the same way when the comparison framework is used.

For a software engineer to be able to select an approach, the characteristics of the project where the modeling approach will be applied must be taken in consideration. So, depending on project characteristics, a criterion may be more important than others. Thus, it is suggested to use an average weight of the grades, where a weight is provided for each criterion. Note that the sum of the weights must totalize 1.

Table 1 shows the comparison framework. The weight of each criterion is defined as 0,25 only for illustrative purpose, but should be changed according to the characteristics of a certain project by the software engineer. The inferior part of the table shows how the average weight must be calculated for a certain approach.

Table 1. Comparison Framework and its respective weights and possible grades.

	Traceability	Scalability	Crosscutting Representativity	Aspect Representativity
Weight	0,25	0,25	0,25	0,25
Grade	1, 2 or 3	1, 2 or 3	1, 2 or 3	1, 2 or 3
The average weight would be calculated by the following formula: M = (gradeTraceab. * weight + gradeScalabi. * weight + gradeAspectRepresenta * weight + gradeCrosscuttingRepresenta * weight)				

3 Modeling Approaches and Case Study

In this section, the four modeling approaches used for the comparison and the system used as case study are presented. The system is a simple application of order processing whose class diagram can be seen on the diagram highlighted with the letter (a) on Figure 2. This class diagram is not shown in details again, only shown illustratively, for example, at the higher part of Figure 3. In this application, a customer (`Customer class`) can request an order (`Order class`) to be made. An order is composed of several items (`Item class`) where each item is a number of a specific product (`Product class`).

On Figure 2, the system modeled with the Pawlak *et al.* approach is shown [11]. These authors use the stereotype `<<aspect>>` in a class to represent the aspects, the stereotype `<<pointcut>>` in the relationships to represent the point that an aspect affect, the stereotypes `<<before>>` and `<<after>>` in the compartments of the methods to represent respectively the advices that the aspect will execute before and after the execution of the crosscut methods, and the tagged values `?method(..)` or `!method(..)` also in the relationships. The character "?" represents that the method is affected by the aspect in the execution context, and the character "!" represents that the method is affected by the aspect in the call context.

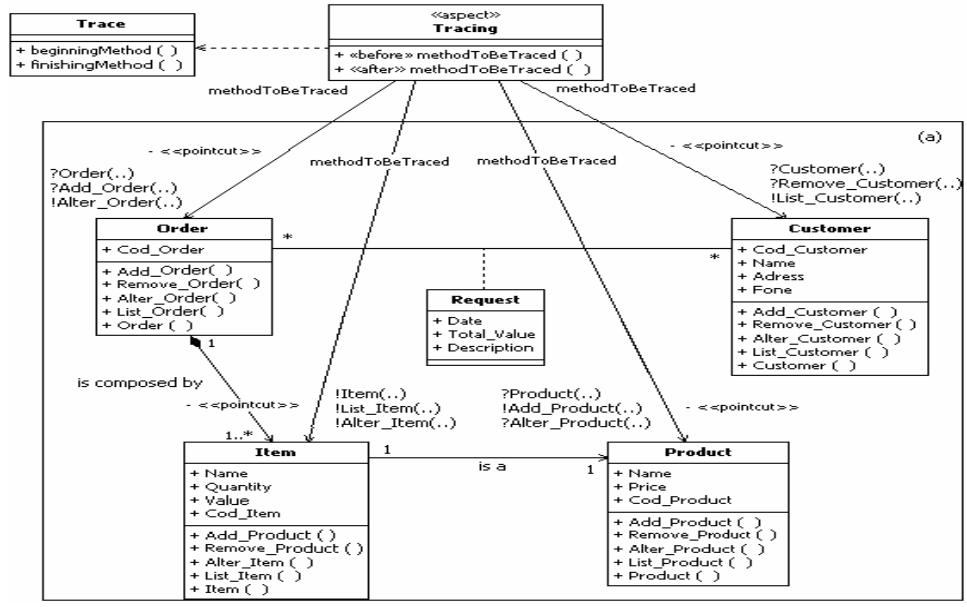

Fig. 1. Pawlak *et al.* approach [11].

On Figure 2, the system modeled with the Clarke and Baniassad approach is shown [5]. These authors proposed an approach of aspect-oriented software development called Theme, which works with two types of themes: the base and the crosscutting. The Base-Theme has the functionalities of the problem domain and the Crosscutting-Theme encapsulates the crosscutting concerns that affect the Base-Theme. A crosscutting-theme is graphically represented by UML templates. These templates encapsulate crosscutting concerns in a generic way, i.e., without creating dependence with concepts and terms used in a certain base-code. After the system is set up with base and crosscutting-theme, it is necessary to make the composition of them with a relationship called "bind". It is through this relationship that it is possible to know which methods of the Base-Theme will be affected by the Crosscutting-Theme. After the bind relationship is done, it is possible to generate the composed model that will not be shown because of the lack of space.

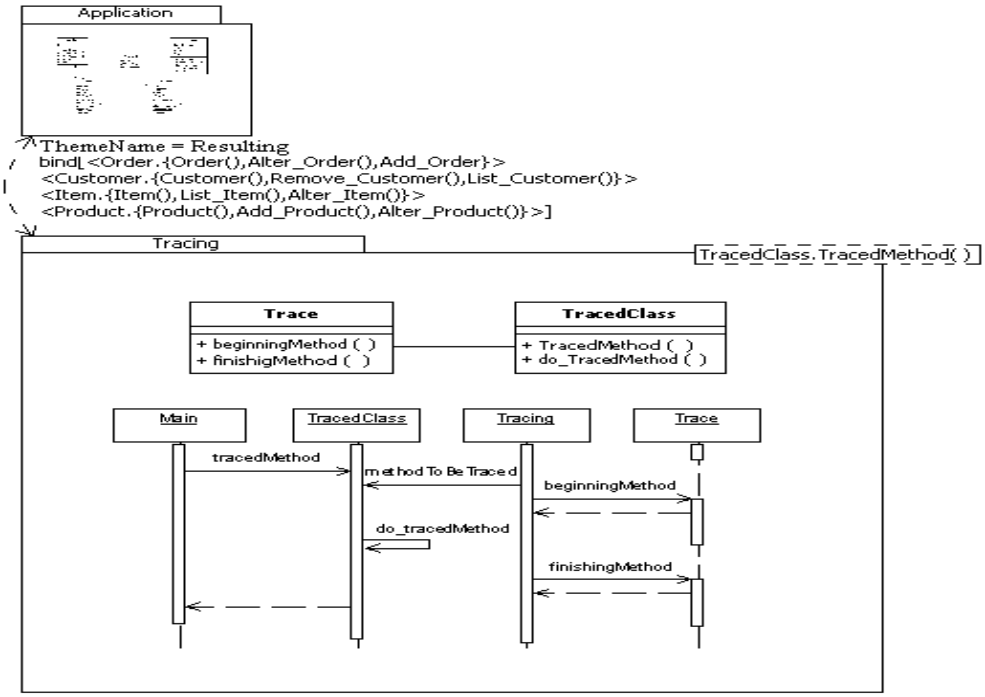

Fig. 2. Clarke and Baniassad approach [5].

On Figure 3 the system modeled using the Groher and Baumgarth approach is shown [8]. The approach use three packages: `base`, `tracing` and `connector`. The package `Tracing` encapsulates the crosscutting concern, the package `Base` has the application that is affected by the crosscutting concern and the package `connector` is responsible for linking the packages `base` and `tracing`.

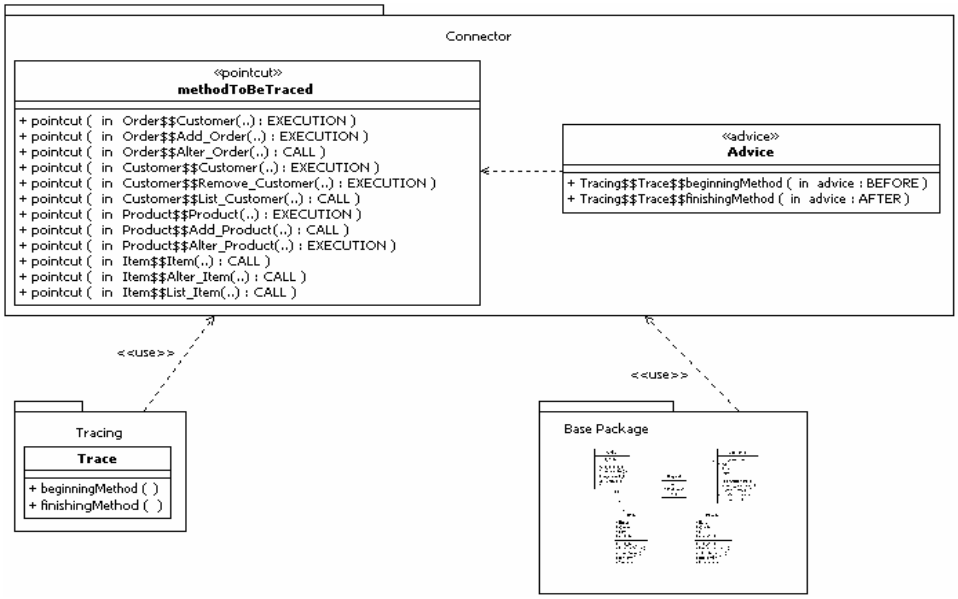

Fig. 3. Groher and Baumgarth approach [8].

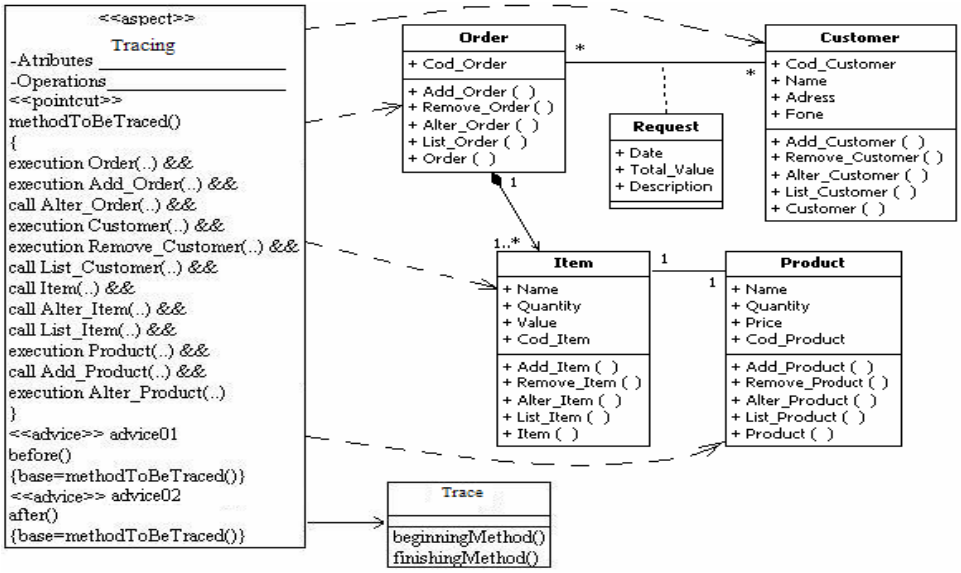

Fig. 4. Stein *et al.* approach [12].

On Figure 4, the order system modeled using the Stein *et al.* approach is shown [12]. In this approach the authors use the stereotype `<<aspect>>` to represent the aspects existing in the system, the stereotype `<<pointcut>>` to represent the pointcut and the stereotype `<<advice>>` for the advices. There are other stereotypes, however they are not presented in this case study. Also, the authors use the delimiters `-Atributes` and `-Operations`.

4 Framework Application

In Table 2, the approaches used in the comparison with all of the comparison criteria and their respective grades and weights are shown. The grade assignment for the criteria "representativity" (aspect representativity and crosscutting representativity) was made comparing the models to each other. The model that allows identifying the aspects and the crosscutting relationships in an easier way got higher grades. The grade assignments for the "traceability" criterion were made by analyzing the difficulty of tracing artifacts on the stages of the development process. The grades of the criterion "scalability" were assigned through changes on the model. The four models were increased with other aspects and other crosscutting relationships. The model that maintained better legibility, even with a high number of aspects, was assigned with the highest grade.

Regarding Table 2, for the Clarke and Baniassad approach [5], grade 1 was assigned for the criterion "aspect representativity", once the aspect is not explicit in the model; it is necessary a deeper analysis to understand how the aspect is represented. Regarding the "crosscutting representativity" the grade 2 was assigned because it is possible to see which methods are crosscut through the `bind` relationship, but it is not possible to know if the crosscut happens in the call context or in the execution context. In the criterion "traceability", the grade 3 was assigned by allowing the artifacts to be traced during the analysis, design and implementation phases. The criterion "scalability" was assigned with grade 1 because the case study grows a lot when compared to other approaches, once twelve sequence diagrams (one for each affected method) were generated.

For the Groher and Baumgarth approach [8] the grade 1 was assigned in the criterion "aspect representativity", once the aspect is not explicit in the model. Regarding the "crosscutting representativity" the grade 2 was assigned because it is possible to identify the methods that are crosscut by the aspect, through the package `Connector`. In the criterion "traceability" the grade 1 was assigned by not allowing the artifacts to be traced in the development phases. In the criterion "scalability" the grade 2 was assigned because in a big system, the model would not be as large as the Clarke and Baniassad model [8] nor as polluted as the Pawlak *et al.* model [11].

For the Pawlak *et al.* approach [11] the grade 3 was assigned for the criterion "aspect representativity", once the aspect is explicit in the model by the stereotype `<<aspect>>`. Regarding the "crosscutting representativity", the grade 2 was assigned, since it is possible to see which methods are crosscut by the aspect, because of the tagged values that are in the bind relationship and also to know if the methods are affected in the call or in the execution context. In the criterion "traceability", the grade 1 was assigned by not

allowing the artifacts to be traced in the development phases. In the criterion "scalability", the grade 1 was assigned because tests showed that the number of crosscutting relationships harms the model legibility when the number of aspects increases.

For the Stein *et al.* approach [12] the grade 3 was assigned in the criterion "aspect representativity", for the same reason of the Pawlak *et al.* approach [11]. Regarding the "crosscutting representativity", the grade 2 was assigned, as it is possible to see which methods are crosscut by the aspect by the stereotype `<<pointcut>>` and, to know if the methods are affected in the call context or in the execution context. In the criterion "traceability", the grade 2 was assigned because the model allows the traceability during the class model and the implementation. In the criterion "scalability", the grade 2 was assigned because of the great number of crosscutting relationships (associations) found in large models. The methods that are affected are located inside the aspect instead of being represented as tagged values, as in the Pawlak *et al.* approach [11].

Table 2. Comparison Framework Application

Criterion / **Approaches**	**Representativity**				**Traceability**		**Scalability**		**Avg Wgt**
	Aspect		**Crosscutting**						
	grd	wgt	grd	wgt	grd	wgt	grd	wgt	
Pawlak *et al.*	3		2		1		1		1,75
Clarke and Baniassad	1	0,25	2	0,25	3	0,25	1	0,25	1,75
Groher and Baumgarth	1		2		1		2		1,5
Stein *et al.*	3		2		2		1		2
Note: grd = grade, wgt = weigt.									

According to the comparison, the Stein *et al.* [12] approach got the highest weighted average considering the four comparative criteria with the same weight. Thus, if in a certain project the "traceability", "scalability" and "representativity" criteria are equally important factors, it is interesting to use this modeling approach. However, we must consider the restrictions of this comparison. We believe in the sentence which describe in a more precise way the comparison shown in Table 2, that is: "When all of the four criteria have the same importance inside a project and only the Trace crosscutting concern is used, Stein *et al.* [12], seems to be the most suitable approach".

It is possible to observe by the grades that there is no best approach for any type of project. In the case of a system in which the priority is to maintain the mapping on the artifacts during the analysis, design and implementation, it would be interesting to use the Clarke and Baniassad approach [5] for being the only one that obtained a high grade for the criterion "traceability". Regarding the "aspect representativity", the Pawlak *et al.*

approach [11] and the Stein *et al.* approach [12] are the ones that obtained the best grade, if compared to the others, for showing the aspect very explicitly in the model. Yet, regarding the "scalability" the Groher and Baumgarth approach [8] is the one that would be the best choice, by the fact of the final model not being very big nor very polluted.

5 Final Remarks

The main contribution of this paper is the supplying of a preliminary comparison framework that can be used by software engineers to compare modeling approaches in order to point out one of them as the most appropriated under specific constraints. Besides, another contribution is an initial comparison among four modeling approaches which pointed out one of them as the most suitable when the four criteria have the same weight and the Trace crosscutting concern is used. As the comparison framework is still very preliminary, we cannot say that Stein *et al.* [12] approach is the most proper approach for all cases.

Through the grades and the weights, it is possible to obtain an average weight for each one of the four modeling approaches, and with this, it was concluded that none of the four proposals is optimal, but all of them present their strong and weak points. Thus, the effectiveness of an approach depends on the characteristics of the project being developed.

It is important to note that our framework was designed to be used for researchers interested in build a catalogue of modeling approaches that best fit for certain situations. Of course our framework is not useful under time constraints, as for example in industrial software production. In this situation a software engineering have no time to model several systems with different modeling approaches only for selecting one of them. So, it is interesting to develop a comparison framework that can indicate a specific modeling approach before to model a system. However, this is a very difficult task because the criteria should be applied in the meta-model of the approach, but a lot of them are not available or do not exists.

Regarding the limitations of this work, we can mention that the result of a comparison must not be still considered the final decision when choosing a specific approach. We only have four criteria, and in a real software development process other criteria may be necessary. We intend to extend the criteria set by conducting experiments to validate the comparison framework in order to verify if a approach pointed out by it, is really is the best fit within some contexts. A less trivial crosscutting concern also is needed: the Trace is concern is a classical concern in the aspect oriented programming. This concern was used because a more complex concern could obscure the analysis. Indeed it is necessary to make a better case study with more non trivial crosscutting concerns for better results. This will be done in future work.

References

1. Aldawud, O., Elrad, T. and Bader, A**.** 2003. UML Profile for Aspect-Oriented Software Development. In: Proceedings of Workshop of Aspect Oriented Modeling with UML of Aspect Oriented Software Development Conference (AOSD), 2003.
2. Barra, E. Génova, G., Llorens, J. 2004. An approach to aspect modeling with UML 2.0. In Proc. 5th Int. Workshop on Aspect-Oriented Modeling, October 2004.
3. Chavez, C.F.G., Lucena, C.J.P. 2001. Design Support for Aspect Oriented Software Development. Doctoral Symposium at OOPSLA'2001 and Poster Session at OOPSLA'2001, Tampa Bay, Florida, USA, October 14 – 18, 2001.
4. Chitchyan, R., Rashid, A., Sawyer, P., Garcia, A., Pinto, M., Bakker, J., Tekinerdogan, B., Clarke S., Jackson, A.. AOSD-Europe-ULANC-9 Technical Report, Lancaster University, 2005.
5. Clarke, Siobhán and Baniassad, Elisa. 2005. Aspect Oriented Analysis and Design – The Theme Approach. Editora Addison-Wesley, First Edition. 2005.
6. Evermann, J. 2007. A Meta-Level Specification and Profile for AspectJ in UML. Victoria University Wellington, Wellington, New Zealand. AOSD 2007.
7. Gradecki, J.D. and Lesiecki. 2003. N. Mastering AspectJ – Aspect Oriented Programming in Java. Wiley Publishing. 2003.
8. Groher, Iris, Baumgarth, Thomas. 2004. Paper: Aspect-Orientation from Design to Code. Munich, Alemanha. 2004.
9. Harrinson, W., Tarr, P., Ossher, H. 2002. A Position on Considerations in UML Design of Aspects. In: Proceedings of Workshop of Aspect Oriented Modeling with UML of Aspect Oriented Software Development Conference. (AOSD). 2002.
10. Kiczales, Gregor, Lamping, John, Mendheckar, Anurag, Maeda, Chris, Lopes, Cristina, Loingtier, Jean, Irwin, John 1997. Aspect-oriented programming. In Proc. of ECOOP (European Conference on Object-Oriented Programming), Springer-Verlag. 1997.
11. Pawlak, Renaud, Duchien, Laurence, Florin, Gerard, Legond-Aubry, Fabrice, Senturier, Lionel, Martelli, Laurent. 2002. Artigo: A UML Notation for Aspect Oriented Software Design. Workshop AOSD 2002. France. 2002.
12. Stein, D., Hanenberg, S., Unland, R. Designing Aspect-Oriented Crosscutting in UML. In: Workshop Aspect-Oriented Modeling with UML, AOSD, Enschede, April, 2002.
13. Zakaria, A.A., Hosny, H., Zeid, A. 2002. A UML Extension for Modelling Aspect-Oriented Systems. In: Proceedings of Workshop of Aspect Oriented Modeling with UML of Aspect Oriented Software Development Conference (AOSD). 2002.