

# The Symbiosis between View and Aspect

Eduardo Barra  
ebarra@inf.uc3m.es

Anabel Fraga  
afraga@inf.uc3m.es

Juan Llorens  
llorens@inf.uc3m.es

Departamento de Informática  
Universidad Carlos III de Madrid  
Avenida de la Universidad, 30  
28911 Leganés – Madrid, Spain

## ABSTRACT

If we are planning to do a global architecture description of a system, diverse criteria and concept applications could be used. The problems increase applying a Aspect Oriented (AO) development.

A research has started in order to aid the Aspect Oriented Software Development (AOSD), as well as support possible investigation in the area of quality Software Architecture (SA) descriptions. The idea is to explore in the unification of concepts that have the same roots or starting point.

The main goal is to offer an unified criteria for the representation of a Software Architecture description in any kind of development. The need of symbiosis between two concepts View and Early Aspect is specified. View concept, it is in the Software Architecture description studies. On the other hand, Early Aspect concept is presented in the Aspect Oriented Software Development research area.

## Categories and Subject Descriptors

D.2.2 [Software Engineering]: Design Tool and Techniques – *Aspect Oriented Design Methods*. D.2.11 [Software Architectures]: Languages.

## General Terms

Aspect Oriented Software Development and Software Architecture.

## Keywords

Early Aspect, View, Viewpoint, Software Architecture.

## 1. INTRODUCTION

The Aspect-Oriented Software Development (AOSD) has achieved its greatest strength thanks to the management of the concerns of the software system components at the stages of the final solution domain, without taking care of the early stages. Stages as requirements engineering and the stage in which it is identified the first structure of the early components in a system with its initial relations, known as the global software architecture of a system. Though, one of the abstract concepts introduced by AOSD for the concerns management at the early stages in a software development process is the Early Aspect, more specifically called in the architecture stage as Architectural Aspect. The Early Aspects initiative is the same like the Aspect initiative concept coined by Kickzales [1], with the novelty that the Early Aspects focuses on the management of the crosscutting concept features at the software development early stages. That

concept is perhaps the supporting point for the concerns management strength in the problem domain and the early solution.

On the other hand, diverse proposals have been developed in the Software Architecture (SA) research flow in order to describe the global software architecture of a system, being mostly nowadays based in the Viewpoints separation and identification. To understand this valuable concept we must think about how in a software system production it is necessary to imagine over large concerns to consider, each one of them depending on the specially outstanding perspective to a special kind of person related to the system. The persons are identified as stakeholders, each one of them perceive the system from its own perspective, and for this reason they are usually the basics on which diverse Viewpoints are created for many architectural descriptions.

The publication of the IEEE 1471 standard “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems” in 2000 has set the basis for the unification of a software systems global architecture description. This standard proposes a modular separation where each part is called View, created from different perspectives known as we said above as Viewpoints.

Analyzing the previous concepts, it was the origin of the idea of this paper, which tries to provide the justification for the symbiosis between View and the Early Aspect concepts, joining them in just one stronger concept. Thereby, in a natural way, it is provided a key concept to the methodology reinforcement in the design modeling at the architectural description early stage in a software system to any development paradigm, as the Object-Oriented model, preparing the system for the continuity of an Aspects-Oriented (AO) development.

The reminder of this paper is structured as follows: in Section 2 we propose a new casual definition of Aspect trying to support the natural evolution that has suffered in every software development stages. In Section 3, we analyze the primal concepts of the IEEE 1471 standard, like the View. Section 4 provides an approach of the symbiosis justification between the View and the Early aspect concepts. Section 5 presents some conclusions and future works.

## 2. ASPECT CONCEPT EVOLUTION

The AOSD was initially born from the Aspect-Oriented Programming (AOP) [1] and the Aspect concept was defined to carry out with this novel ideology. For this reason, it seems logical that the Aspect concept definition evolves to cover every of the stages of an AO software development.

## 2.1 Aspects in AOP and AOSD

The research works about AO are based in the separation of concerns. One concern is a requirement or specific consideration that must be taken into account to satisfy the intent of any software system. Hence, we can say that a software system is the performance or achievement of a concerns set. Likewise, these concerns can be high or low level concerns, useful or not useful concerns, business rules, synchronization, delivery, etc. Many dissimilar kinds of concerns can be outstanding to different developers in diverse roles, or in diverse stages in the software life cycle. Moreover, the separation of concerns entails the software decomposition depending on one or more concerns dimensions [2].

*“An aspect, if it can not be cleanly encapsulated in a generalized procedure. Aspects tend not to be units of the system’s functional decomposition, but rather to be properties that affect the performance or semantics of the components in systemic ways”.*

The former definition belongs to the AOP creator, Gregor Kiczales. The AOP offers a suitable framework that clarifies and facilitates the Aspect definition. Though, this proposal focuses just on the final solution model, where modulating and composing crosscutting concerns has been specified with a high efficiency rate.

The AOP approach has provided techniques and methodologies to perform the advanced separation of concerns at the implementation stage. The base was to propose the group of the Object-Oriented and structural languages as development technologies to the basic functionality concerns (behavior or functions), and to attend that concerns foreign to the basic functionality, which traverse the whole code and have a common behavior encapsulated in a new first class entities called Aspects that applies new AO languages. This proposal separates the Aspects from the rest of Components which compounds the system. Then, we find that the elements in a system could take form of a Component or an Aspect; the peculiarity of the first one is that can be clearly encapsulated in a general process (an element is clearly encapsulated when it is right allocated, easily obtainable and easy to repair); the second one can not be clearly encapsulated in a general process. The first ideas from the Aspects used to be features about the Components semantics in a systematic way (for example: synchronization, memory management, delivery, etc). But, some other inclinations suggest that Components should be treated as entities from the same level than an Aspect. In contrast, the AOSD foment the aim of promoting the advanced separation of concerns at every life cycle stages in the Software Development Process (SDP).

The idea of attend every model on its diverse abstraction levels in a system as parts, without exception, that belongs to at least one Aspect, and that every Aspect has dissimilar conceptual levels at every stage in a software development. These are signs of the Aspect concept evolution.

With one idea of evolution, the research area known as AOSD. It actually achieves the label of Aspects Oriented Paradigm, unifying in just one name the whole AO techniques and methodologies for the software systems production with quality.

Some research groups have been working in order to put in practice the goal of the advanced concerns separation through the Aspect concept at the diverse stages in the software development

life cycle, proposing different abstraction levels and composition devices, discerning some nuances from the Aspect concept within those stages:

Aspect: the AOP proposed the Aspect at the lowest level (implementation Aspect) abstracting and composing Aspects in the solution domain.

Early Aspect [3]: the early aspects aim is the management of the Aspects created at the early development stages, abstracting and composing Aspects in the problem domain and the initial solution.

Architectural Aspect [4]: at the architectural design level there are concerns represented in crosscutting. The abstraction and composition of these concerns can not be easily located, and there are initial proposals for the specification of these components or individual architectural models.

It must be remarked how the Aspect concept has transcended to others stages in software development life cycle, and its based on the dimension of abstraction stages. Each approach is based on the general concept of Aspect, but curiously this concept is not the one initially proposed by the AOP which just attended the performing stage.

Aspect, Join Point, Crosscutting and Weaving are the concepts introduced by Kiczales on his Aspect-Oriented Programming paper, and nowadays they establish the Aspects-Oriented Paradigm core. Next, these concepts are described in a global sense in order to accept the application of the same criterion at every stage of the software development.

### 2.1.1 Join Point

It is clear that there is a relation between Aspects and the system components or models, and between the different Aspects involved on the same software development stage too. Therefore, they must act on each other somehow. In order to be woven they must have some common points, known as join points.

### 2.1.2 Crosscutting

The join points are a special interface type between the Aspects and the Components or Models, but the really woven atomic elements are the Crosscutting, which represent a concern feature scattered on a software system. The Crosscuttings are represented on an abstract way in the Aspects, and on a logical and physical way in the Components or Models at the diverse software development stages.

### 2.1.3 Weaving

The weaving concept establishes the devices for abstraction and composition, as static as dynamic, between Aspects and Components or Models, always at just one software development stage.

## 2.2 The New Aspect

The preceding review helps to understand the reason why the Aspect concept nowadays needs a new definition, in order to offer to the developer a general context that allows separating clearly the Aspects in its diverse roles, at the different Aspect-Oriented software life cycle stages, to produce quality software.

*“An Aspect is a modular unit grouping concerns, tested and selected from specific points of view, scattered through the diverse models created at the different software development life cycle stages. The Aspects will change its abstraction level at the software development life cycle stages, and it will exist or not a trace between every different transformations of an Aspect”.*

The goal of this new casual approach to the Aspect definition, it is converging the investigation trails within the AOSD.

On the search of AO essential abstract concepts acceptance, along every software development stages, it is been established that the abstraction level is the foundation.

### 2.2.1 Abstraction and instance

Human beings make abstractions from the understanding of particular things. However, converting an abstraction in something concrete requires an almost religious conviction.

In order to identify the differences between the diverse AO concepts at every software development stages, it is important to think about the abstraction and instance concepts. The essential system features are represented at one of the stages of the software system modeling process development. This is achieved testing and selecting concerns from some Viewpoints, omitting the irrelevant or less substantial details.

The abstraction is a common mental process, as usual that is absolutely transparent to the human mind. Hence, it is remarkable that the same entity from the real world can be modeled in different abstraction levels. Within the Object-Oriented Paradigm [5], it is described that an instance is a concrete abstraction indication. So at the same time this abstraction is a specification of an instances group, where this abstraction could be treated as a concrete instance at the moment when it will be related to some higher abstraction level specification.

In a similar way, the Aspects are specifications at a stage previous to that stage on which it is been modeling; and the stage on which it is working, it is a concrete instance, when exists a trace between them. However, the transformation of a stage model into the next stage model is not easy, but this is not something treated on this paper.

## 3. VIEWPOINT AND VIEW

Nowadays, the software engineering community disagreed in how it must be done the architectural description of a system, and in the contents and structure that the description should have. However, the basis proposed by the IEEE 1471 standard are been adopted for a great part of the community.

The IEEE 1471 standard [6] uses a system modular separation in different prospects, called Viewpoints. Each system owns an architecture described by models collection, for its documentation, according to the rule. This architectural description is organized by Views which are composed for one or some models, where each one of this Views represents one or more of those system concerns relevant to the stakeholders group. So, from the stakeholder group perspective, a Viewpoints is defined to perform the View models. A Viewpoints allows concentrating on some concrete concerns of the system, omitting the rest. Each Viewpoints provides a sequence of individual concepts and structural rules in order to obtain the View abstraction, and enables represent properly the outstanding ideas for the treated concerns.

The standard considers the requirement of treating the Views consistency and completeness, demonstrating that one architectural description is coherent if none of its Views imposes contradictory requirements about one of the others Views, and it is complete if contains every details of the system stakeholders (See Figure 1).

The IEEE 1471 standard sets up the information that any specification of whichever Viewpoints must contain:

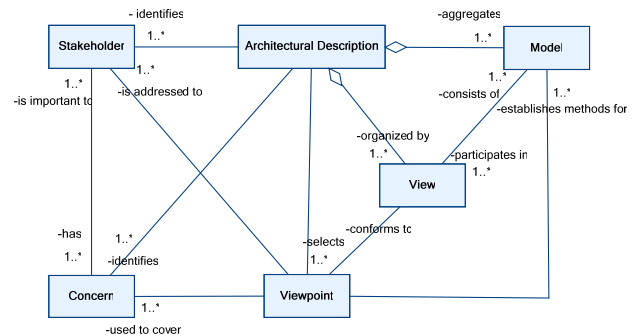


Figure 1 Conceptual Model IEEE Std 1471

- The Viewpoints name.
- The system stakeholders that considers outstanding the Viewpoints.
- The system concerns treated by the Viewpoints.
- The languages, notions, modeling techniques and methodologies to use in order to create Views associated to the Viewpoints.

Likewise, the standard says that each View will include:

- An identifier and other introductory information, as defined by the using organization
- Representation of the system constructed with the languages, methods, and modeling or analytical techniques of the associated viewpoint
- Configuration information, as defined by the using organization.

## 4. WHY A SYMBIOSIS?

The symbiosis between View and Aspect concepts facilitate the Aspect Oriented Software Development and Software Architecture. It provides common benefits to the Software Engineering community.

### 4.1 Architecture Key concepts

In general, no known definition of Software Architecture is acknowledged for the whole community. However, it is present a convergence in the definition of the standard IEEE 1471, it says that Software Architecture must be understood as the fundamental organization of a system, embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.

Following that definition, some key concepts could be extracted:

- The organization, as a qualitative and structural concept.
- The components, as early elements of the solution.
- The relationships between components, as means to interact.
- The restrictions, as result of the influence of the environment, as well as non functional requirements of the starting software.

That key concepts are not new, in the earliest attempts to search patterns in the Software Architecture, the architectural styles definitions took advantage of them. Pery and Wolf [7], defined architectural styles as abstractions of element types and formal aspects of diverse specific architectures. An style encapsulates important decisions of architectural elements and give emphasis to relevant restrictions in the related elements. Quite the opposite,

Shaw and Garlan [8], Garlan et al. [9], and Shaw and Clements [10] identify styles in function of interacting patterns between typed components.

In the search of Organization, the IEEE 1471 standard used the concept View Point. In diverse proposals of development for an architectural description, it is natural to structure the system using View Points, even without manifesting it openly. For example, in Web applications development, three Viewpoints are proposed: navigation, presentation, and business logic. In the MDA model suggested by the OMG (Object Management Group) [11], three Viewpoints are suggested: computer independent, platform independent, and platform specific. One of the proposals for use views is the Kruchten model [12], where five views are recognized, but the considered Viewpoints for the organization of the architecture are not explained there.

The first model that took openly the IEEE 1471 standard is the RM-ODP (Reference Model –Open Distributed Processing) [13], it established some key concepts for the open and distributed application development. It describes five general view points: business, information, computer, engineering and technology. This last model is the best one and it describes the Viewpoints to use, that is why it is used more often in the community.

On the other hand, each View Point use a series of own concepts, constructions and structural rules. An abstraction form that helps concentrate in some concrete concerns of the system. It helps to abstract it from the others, creating a View. The standard manifest that a View could be formed of one or more models. But it is not clear the difference between View and Model.

## 4.2 View vs. Aspect

The View Point concept is the greatest contribution of the proposal for architectonic description: IEEE 1471. Even though, the specified View in that proposal has no abstract concepts for its modeling.

On the other hand, since the publication of the thesis of Aspects, it gave the key concepts Crosscutting and Joint Point that makes it strength.

In the implementation stage of a OO or AO system. The Aspects describe advice to the objects behavior, referencing the Object Classes, defining the Joint Point where the advice should be located. In the early design stage of a software architecture the key concepts of Aspects are valuated but using the name of Early Aspect or Architectonic Aspect.

In the approach to find the symbiosis, it is necessary in the article to provide four clarifications (See Figure 2):

1. Relating the simple definition of Aspect (An Aspect is a modular unit designed for the implementation of one or more concerns) with the definition of View in the IEEE 1471 standard (An Aspect is a representation of a system from the perspective of a related group of concerns). By one side, the Aspect concept seems to have the focal point in the composition and the View concept in the abstraction.
2. The nonexistence of properties that makes a clear difference between View and Model in the IEEE 1471 standard, where it is thought that a View could be created with one or more architectural models. And an architectural model could participate in more than one view. That lack is not

present in the difference between Aspect and Model. Thanks to the concept crosscutting the concerns of a system could be differentiated if they are clearly encapsulated in a general procedure and the concerns that could not be clearly encapsulated are in the Early Aspects. It must be clear that crosscutting is a concern property that gives abstraction and composition mechanism.

3. As mentioned, the standard considered the need of contemplate consistency and completeness of Views, it is required in the advance separation of multidimensional concerns.
4. The View Point that must establish conventions about why a View is created, represented and analyzed, gives strength to AO for clarifying in early stages the concerns that must be taken for the development of a software system. Absorbing the manage of Joint Points as a mechanism of weave between Models and Early Aspects or Views.

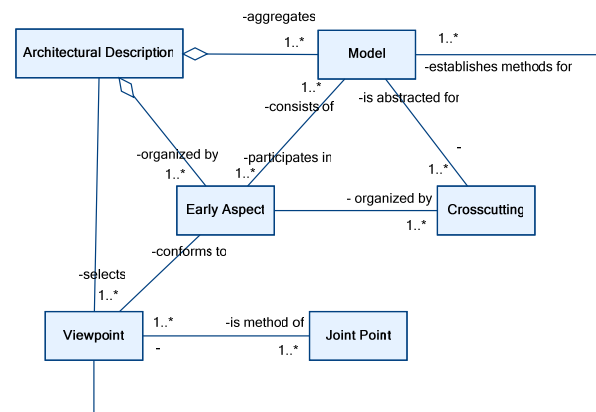


Figure 2 Conceptual Model Proposal

## 5. CONCLUSIONES

It is obvious that DSOA is an Aspect Oriented Paradigm. It is included in all stages, even before the concept Aspect came to be used. Maybe, it is the motivation why researchers feels that they worked in an AO but with a different label.

In the development of a software system in the stage of architectural description, we had presented a general survey about AO and SA. We found some justifications to accept the symbiosis between View and Early or Architectonic Aspect with the implication that one has over the other, the benefit is mutual. It does not mean that one of the concepts should disappear, just to be accepted as one and use the name considered appropriate in the context.

Right now, a lot of systems with huge quantity of software installed some time ago are getting going to apply AO techniques and methodologies for extensibility and maintainability. However, the organization of concerns is in the architectonic description and specifically with the concepts View and View Point. So, it is of high interest to unify vocabulary between the development team of a system, as well as seek techniques and methodologies based on the same abstract concepts.

A proposal suggested in the article is pending to develop as a future work. Using the new proposal definition of Aspect as an starting point, then a consensus must be established in the DSOA community for getting a standard in the level of abstraction of Aspects in the whole software development process. Therefore, it could be possible to search for modeling rules, and the traceability in the software development process or life cycle.

## 6. REFERENCES

- [1] Kickzales, G. Lamping, J. Mendhekar, A. Maeda, C. Videira Lopes, C. Loingtier, J. M. Irwin, J.. Aspect-Oriented Programming. *In Proceedings of the European Conference on Object-Oriented Programming ECOOP'97.* (Jyväskylä, Finland, June 1997) Springer-Verlag, Berlin Heidelberg, 1997.
- [2] Tarr, P., Ossher, H., Harrison, W. and Sutton Jr., S. M. N Degrees of Separation: Multidimensional Separation of Concerns. *In Proceedings of the 21st International Conference on Software Engineering.* (Los Angeles, CA, USA May 16-22, 1999) ACM, New York, 1999, pp.107-119.
- [3] Rashid, A., Sawyer, P., Moreira, A. and Araújo, J. Early Aspects: a Model for Aspect-Oriented Requirements Engineering. *IEEE Joint Conference on Requirements Engineering.*(Essen, Germany, September 2002), pp.199-202.
- [4] Tekinerdogan, B. ASAAM: Aspectual Software Architecture Analysis Method. *In WICSA 4th Working IEEE/IFIP Conference on Software Architecture.* (Norway, 2004) pp.5-14
- [5] Booch G. Object-Oriented Design with Applications. The benjamin Cummings Publishing Company, Inc, 1991.
- [6] IEEE-1471 Standard. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society. 2000.
- [7] D. E. Perry, A. L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4), Oct. 1992, pp. 40-52.
- [8] D. Garlan and M. Shaw. An introduction to software architecture. Ambriola & Tortola (eds.), *Advances in Software Engineering & Knowledge Engineering*, vol. II, World Scientific Pub Co., Singapore, 1993, pp. 1-39.
- [9] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch, or, Why it's hard to build systems out of existing parts. *In Proceedings of the 17th International Conference on Software Engineering*, (Seattle, WA, 1995). Also appears as: Architectural mismatch: Why reuse is so hard. *IEEE Software*, 12(6), Nov. 1995, pp. 17-26.
- [10] M. Shaw and P. Clements. A field guide to boxology: Preliminary classification of architectural styles for software systems. *In Proceedings of the Twenty-First Annual International Computer Software and Applications Conference (COMPSAC'97)*, (Washington, D.C., Aug. 1997), pp. 6-13
- [11] MDA Guide Revision Draft, Version 00.03. *Document ormsc/05-11-03*, November 2005. <http://www.omg.org/>
- [12] Kruchten, P. *The Rational Unified Process: An Introduction*, 2nd ed. Reading, MA: Addison Wesley Longman. 2000.
- [13] ITU-T Recommendation X.901 ISO/IEC 10746-2. Reference Model - Open Distributed Processing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AOSD'07, March 12–16, 2007, Vancouver, British Columbia, Canada.  
Copyright 2007 ACM 1-58113-000-0/00/0004...\$5.00.