

Early Aspects: Some Analysis, Trends and Perspectives

Antônio Maria P. de Resende
Department of Computer Science
Federal University of Lavras
POB 3037, zip code 37.200-000
Lavras – MG – Brazil
+55 (35) 3829-1545
tonio@dcc.ufla.br

Fábio Fagundes Silveira
Department of Computer Science
Technological Instit. of Aeronautics
Pça Marechal Eduardo Gomes, 50
São José dos Campos – SP – Brazil
+55 (12) 3947-2636
ffs@ita.br

Adilson Marques da Cunha
Department of Computer Science
Technological Instit. of Aeronautics
Pça Marechal Eduardo Gomes, 50
São José dos Campos – SP – Brazil
+55 (12) 3947-2636
cunha@ita.br

ABSTRACT

Despite generated benefits, Object-Oriented (OO) paradigm seems reaching its limits, regarding complexity reduction of current systems. In this context, the Aspect-Oriented (AO) comes up as an alternative to reduce software development complexity while keeping OO advantages. Needs for investigating methodologies of Aspect-Oriented Software Development (AOSD) has emerged along with AO. As an example, Early Aspects (EA) aim to identify aspects on the early stages of software development, such as domain analysis, requirements specification and architectural project. Being one of the newest software engineering paradigms, AO emphasizes that new studies and experiments should be carefully carry out, in order to establish improved methods, techniques and tools applicable to this new way of development. This paper presents the state of the art and some open issues about EA to be investigated by the scientific and technological community, aiming to develop, improve as well as discuss the EA's methodology, in order to reach a better use of AO potentialities. It also presents a categorization of current works and aspects, besides suggestions to expand boundaries of Software Engineering. It represents the initial investigation effort of a doctoral thesis. The major contribution of this paper is the need to investigate a new third dimension still unexplored by AO researchers.

Categories and Subject Descriptors

D.3.1 [Software Engineering]: Requirements/Specifications – Elicitation methods, Methodologies.

Keywords

Early Aspects, requirements engineering, aspects methodologies.

1. INTRODUCTION

The emergence of Object Oriented (OO) paradigm shows up several benefits to the software engineering field, such as the software development complexity reduction, as well as facilities to maintain, modularize and reuse software. Despite of such contributions throughout previous years, OO seems to achieve its limits for reducing systems complexity nowadays [1] [2] [3] [4]. The AO has appeared in this context, being able to reduce the

software development complexity and keeping benefits achieved by OO [3].

Along with AO, needs of figuring out a methodology of Aspect Oriented Software Development (AOSD) has emerged, enabling to identify, separate, design and compose aspects, as known as crosscutting concerns. This methodology shall support software engineering phases such as analysis, design, implementation, testing and maintenance.

Early Aspects (EA) represent the subset of activities belonging to the AOSD, aiming to identify aspects from initial phases of software development as domain analysis, requirements specification and architectural design, as shown in Figure 1.

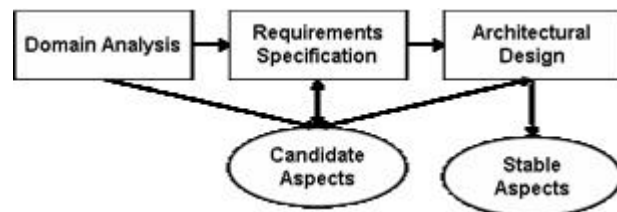


Figure 1 – Early Aspect Scope

This paper presents the EA *status quo*, a set of open issues and suggestions about how to develop a methodology aiming to take advantages of AO paradigm potentialities.

The rest of this paper is organized as follows. Section 2 introduces some background information related to this research. Section 3 briefly summarizes related works on Early Aspects. Next, in Section 4, the main ideas of the paper are discussed. Section 5 shows the current work, and concluding remarks are given in the section 6.

2. EARLY ASPECTS

Like other paradigms, such as Procedure Oriented (PO) and OO, and Modular Programming (MP), AO is based upon the separation of concerns (SoC) principle, presented by Dijkstra [5]. Modular programming researchers, as Maynard [6], have used this principle since the end of 60's to define how many and which are the modules that compose the whole system.

The SoC is equivalent to the functional and systems decomposition practices, which comes from the system theory.

The aiming of such techniques is to divide the system into smaller parts and treat them in a separated way. Regarding SoC, these concerns after separated, might be implemented as one or more aspects. The EA area lacks a methodology that identify, model and handle aspects on the early phases of a software development process. Building this methodology is the doctoral thesis major contribution of the first author of this paper. Parnas [7] has written one of the most important papers on system decomposition, where criteria based on projects decisions are applied.

The EA methodology shall overcome difficulties of applying SoC principle in the requirement specification phase, especially in the specification of Non-Functional Requirements (NFR) which naturally crosscut each other [38] and are scattered and tangled into Functional Requirements (FR).

Among concerns involved in software development, crosscutting concerns can be emphasized, which are hard, due to traditional deficiency, to be modeled and implemented by means of scattering and tangling. One of the most important differences of AO deals with that deficiency, aiming to uncouple and encapsulate concerns as security, distribution, synchronization, persistence, mobility, interface, performance, application domain and so on.

3. RELATED WORKS

Nowadays, EA investigation results would be categorized as templates and methodological proposals.

1. **Templates** - represent investigation results that outline structures, taxonomies, schemas, patterns, standards and others, used as references or formats to guide software modeling. Those formats accept end users attachments to adapt templates to the reality of each work environment. Hence, they can be called flexible templates.
2. **Methodological** - represent investigation results of: a method, systematic, processes, techniques, frameworks, approaches, and so on. Methodological articles usually propose or apply templates. However, such templates are not the focal point.

Templates show a kind of format or references that when used by developers become suitable to fit and adapt systems to it. On the other hand, methodological results present methods, techniques and criteria to break down the system, and do not show necessarily a template. In practice, templates and methodological results complement each other, guide developers on the decomposition activity and turn less hard the identification and mapping of concerns into aspects.

The categorization above outlines how researchers are looking for solutions to identify and treat early aspects. Would these two categories be the only existing ones? Would be other alternatives not seen due to the fact of researchers are tied to previous paradigms?

3.1 Templates

Sutton [8] presents a schema called Cosmos to aim the multidimensional concern spaces modeling. That schema consists of a classification structure used like a template to decompose a system according to the taxonomy proposed. Some elements of that taxonomy are: a) logical and physical concerns; b) categorical interpretive, mapping and physical relationships; and c) predicates.

Moreira and Brito [9] and Moreira and Araújo [10] separate the crosscutting concerns into requirements specification phase. The approach uses a template adapted from Mylopoulos et al. [11] and Malan and Bredemeyer [12] [13] to identify and describe: a) quality attributes, e.g., high level NFR; and b) FR applying use cases. Such articles present use cases and sequence diagram extensions to represent composition of crosscutting quality attributes with functional requirements. Difficulties such as composition rules and conflict resolution do not get an adequate support.

Lopes et al. [14] demonstrate a technique of Design Structure Matrix (DSM), proposed by Eppinger et al. [15], which has the capability to model dependencies inside system, including Aspects.

Monteiro [16] discusses some situations related to the system development, focusing the development of good practices applicable to AOSD.

Monteiro and Fernandes [17] point out to the community traps that occur when using AO paradigm to implement design patterns presented by Gamma et al. [18]. An accurate understanding appearance reason of traps could help to create antipatterns rules.

Monteiro and Fernandes [19] take out aspects from OO legacy codes, organize them and create a set of AO refactoring. That article might be considered an essay for developing patterns catalog.

Hannemann and Kiczales [20] discuss and present proposals to implement AO design patterns. Nowadays, it is very important to spend time developing an AO design pattern catalog helping the SoC activity.

3.2 Methodological

Grundy [23] presents a model called Aspect-Oriented Component Requirements Engineering (AOCRE) to develop components-based software using OO and AO. Main concerns are categorized such as persistence and distribution which have to become available for other components and end users. AOCRE is limited to a specific domain of Components-Based Software Engineering (CBSE), restricting this way the model applicability.

Dingwall and Finkelstein [24] show the development of a system for runtime monitoring of system goals using the KAOS approach. Composition rules are defined using HyperJ, and KAOS is limited to a specific domain, restricting also the applicability.

Rashid et al. [25] show an Aspect-Oriented Requirements Engineering (AORE) enclosing six steps: a) identify concerns; b) specify concerns; c) identify viewpoints, discover requirements and relate to concerns; d) identify candidate aspects; e) specify and prioritize aspects; and f) specify aspects dimensions.

There is a necessity of developing techniques aiming to accomplish the steps above. The dimensions specification is related to SoC. Similarly, dimensions are related to NFR and FR, and their influences in other development phases as well as identification of mapping from requirements to function, aspects and decisions are presented.

Rashid et al. [26] refine the approach proposed in [25], showing detailed rules of aspectual requirements composition based on constraints and operators with a well defined semantic. Rules are applied to specify how an aspectual requirement influences or restricts the behavior of non-aspectual requirements. The paper comments about a scheme of conflict resolution, where the approach is supported by a tool called ARCADE.

Katz and Rashid [27] propose the PROBE framework which implements additional support for traceability on the AORE model. PROBE generates proof of obligations based upon temporal logic that should be hold by an AO system from the initial aspectual requirements and their related trade-offs. Temporal logic inferences can be used as input to tools of formal methods, such as model-checkers or as a basis to generate test cases.

Araújo and Coutinho [28] present a merge of viewpoint-oriented aspects and requirements. The approach with UML is extended to include aspect-oriented concepts during the definition of use case and NFR. NFR that crosscut viewpoints or use cases are non-functional candidates. A use case labeled <<included>> or <<extended>> that is used more than once is called aspectual use case. Use cases that crosscut several viewpoints represent also aspectual use cases.

Whittle et al. [29] and Whittle and Araújo [30] propose an approach to model aspect-oriented scenarios using UML into the requirements specification phase. Such scenarios are modeled and composed with non-aspectual requirements. The approach generates state machine of composed scenarios, designed by sequence diagrams, and used to create prototypes for requirements checking of stakeholders. To generate such state machine, the Whittle and Schumann [31] Synthesis approach is used. A variation of this approach is presented by Araújo, Whittle and Kim [40], where the composition is done in the state machine level.

Baniassad and Clarke [32] propose an approach called Theme to support the aspect-oriented analysis. Initially, a set of domain's sensible verbs are searched from the requirements document, identifying crosscutting concerns. From this point, actions or themes, extracted from the requirements document, are modeled as collections of structures and behaviors that represent a system's

feature. Then, a tool creates relationship graphs between concerns and requirements. Directions are given in [33] to deal with large scale systems.

Yuy, Leite and Mylapoulos [34] show up an approach to find out aspects from relationships between functional and non-functional, using goals-oriented models, where FR and NFR are represented by tasks, goals and soft goals. Finally, the resulting graph is used to identify aspects.

Brito and Moreira [35] present a model to separate concerns during the requirements engineering applying catalogs as well as adopted on OO, helping to identify and specify concerns. This approach gives a composition process introducing match points, dominant aspects and composition rules based on LOTOS operators. Brito [36] extends this approach by using a framework catalog of NFR, besides exploring the composition refinement called LOTOS.

Moreira, Rashid and Araújo [37] show up considerations and discussions about SoC and an approach based on criteria of trade-offs. To define an adequate set of system's concerns, this approach apply trade-offs criteria on concerns compositions, in order to define the better arrangement to the system.

Souza and Castro [38] proposed a goal-oriented requirements methodology founded on SoC (GREMSoC), which aims to improve reusability, maintainability and comprehensibility of requirements specification by means of the SoC application.

Sousa et al. [39] present an approach to apply SoC at several abstraction levels of the software development, enclosing from the requirements specification to design phases. Such approach consist of adapting some use cases driven activities of the unified software development process in requirements, analysis and design workflows.

3.3 To Learn More About

Papers below contain good references to learn more about EA and get information detailed:

a) Araújo et al. [21] discuss EA in initial phases of software development process such as domain analysis, requirements specifications and architectural design. Some related works are grouped and discussed considering each phase separately. This paper highlights the needs of developing EA methodologies

b) Chitchyan et al. [22] discuss AO analysis and design. They describe basic concepts related to phases from requirements specification to design, besides presenting some related works.

4. CONSIDERATIONS AND DISCUSSIONS

4.1 Aspects Categorization

Considering the AO theory, it is possible to categorize aspects as follows:

- Aspects of Structural Dynamic - consist of aspects that have the capability to change classes' attributes and methods. This can be achieved either by inserting or deleting nominally attributes and methods or changing the hierarchical structure of classes;
- Aspects of Functional or Behavioral Dynamic - consist of aspects that have capability to change system behavior by means of the weaving process. This can be achieved in three distinct ways: a) override - substitutes the functional requirements behavior; b) overlap - adds a new behavior; and c) wrap - when a behavior encapsulates other [10]; and
- Aspects of Data Dynamic - consist of aspects that have capability to changing parameters and methods returns.

Resende and Silva [4] state that OO models the world in a static way whereas AO offers the opportunity to model in a dynamic and non-intrusive way. That is one of the most important differences between these paradigms. The OO allows to encapsulate attributes and entities responsibilities, while AO allows encapsulating such entities relationships as well as rules that determine their structural and behavioral changing.

Thus, AO permits to change both the object structure as well as their relationships by means of the system dynamic modeling.

Aspects can be applied in a recursively fashion over several superposed layers, aiding: a) aspects act over objects aiming to change the system behavior; b) aspects act over aspects aiming to change the system behavior; c) aspects act over aspects aiming to take over the system behavior changing layer, building a meta-layer, meta-aspects or a meta-AO.

Considering AO allows modeling the dynamic of a system, the overlapping of aspects layers turns possible to model the dynamic of dynamic, allowing another abstraction level of control, equivalent of computational reflection. However, OA makes easier build that overlapping using AO than reflection techniques e.g. Meta-Object Protocols - MOP.

The flexibility of AO to control dynamic of systems permit to devise the expansion of Software Engineering in two ways: a) approach of Software Engineering and Artificial Intelligence; and b) incorporate automatically laws, agreements, rules and others, in software systems, adapting its changes adjusting the dynamic of systems using AO.

In this paper, it is considered that meta-layer is responsible for a natural intersection between models of software engineering and Artificial Intelligence (IA). Thus, it is possible to make decision and take action on the system application, changing its dynamic by AO.

Figure 2, similar to an inverted sequence diagram, initially depicts a two dimension: a) horizontal dimension – which corresponds to the architectural dimension of developers; and b) vertical

dimension – which corresponds to the functional dimension of end users. It is believed that AO shall be used over the two dimensions, so that a better control over the system could be achieved.

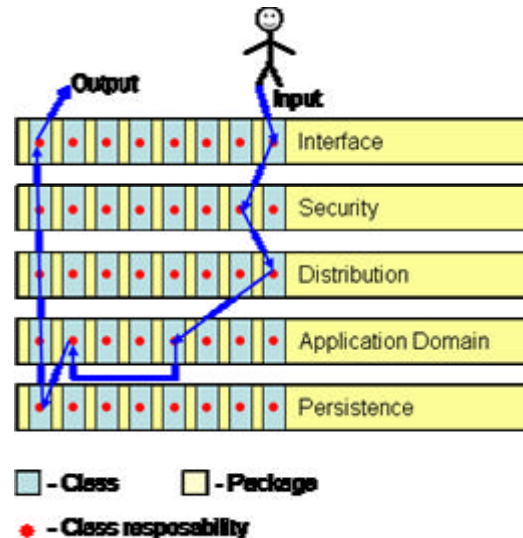


Figure 2 – Dimensions and Concerns Coupling

On the horizontal dimension, AO shall be able to separate functional and non-functional concerns at the initial phases of software development. This favors the uncoupling of concern components, relationship encapsulating and improving the organization system architecture.

On the vertical dimension, AO shall be able to encapsulate functionalities required by users, implementation of functionalities dynamic, and linking the traditional software engineering and IA.

A new third dimension, still unexplored by AO, suggested by the authors, representing the main contribution of this paper is based upon changes of business rules through out the time. Business rules can be stable with predictable or unexpected changes. Therefore, developers can build systems focusing to make easy to arrange the dynamic of systems allowing accommodate of business rules modifications. This dimension seems to be promising and worthy of investigation.

In

Figure 2, arrows can be seen as path to be followed in solution of required functionality, which could be and can be implemented nowadays static or dynamically. Such map can modify itself throughout the time to accommodate changes of business rules, additionally being dynamically implemented using AO. It's necessary to connect business rules and dynamic systems so that modifications in business rules automatically reflect modifications in dynamic systems.

In order to take advantage of AO capabilities to change the structure of objects and their relationships, it is crucial raising the current abstraction level of system models, connecting them to the

organizational enterprise structure. Such connection would provide that making decisions described in official organizational documents could be incorporated by means of applying changing into the system dynamics.

Documents containing organizational decisions should be structured in a way to facilitate their readings by systems, which could automatically alter the system dynamics, adapting to previous decisions. In this case, business rules could be implemented after reading enterprise documents automatically by the system.

It is important to develop ontologies and apply them on public and private sectors so that changes in laws could automatically be interpreted and converted into enterprise rule business changings. The AO, applied to model relationships among entities, might facilitate rules tailoring between government and organizations, as depicted in Figure 3.

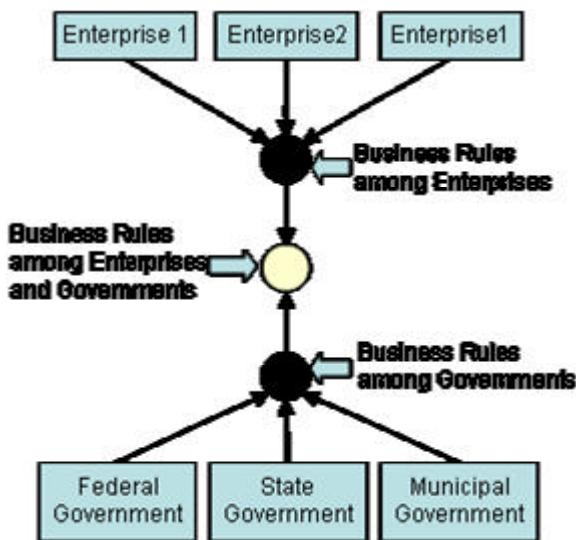


Figure 3 – Relationships among real entities

Business rules which frequently come from legislations or laws, agreements, accords, partnerships, and others, conduct relationships among entities of the real world and therefore must be mapped to systems using new AO potentialities.

From this point of view, two new skills must be considered. The first one consists of extrapolating the AO reasoning and/or abstraction to a meta-AO. In this case, the focus is to capture business rules that are in charge of changing the behavior of object relationships. The later one consists in connecting the system to the sources of decisions which affect the system behavior.

4.2 Technology reuse on the AOSD

EA process should be based on previous well-succeeded technologies such as OO. Among the main applied technologies, the following can be mentioned:

a) Traceability matrix - similar to the Design Structure Matrix (DSM), it can be used to check the dependencies among system

elements. Its use aids an identification of tangling concerns points [14];

b) Catalog - contains templates like structures, taxonomies, schemes and patterns. Some of them can help to separate FR from NFR as well as their subdivisions (e.g. ISO 9126), aiming to improve the catalog with previous lessons learned. The lack of an AO catalog can be noticed; and

c) AO Heuristics - nowadays there are heuristics enabling the identification of objects from requirements. However, this can not be noticed on the AO, where empirical activities and decisions about how to separate concerns must be taken.

Throughout the software development, the mapping activity from concerns and entities to aspects must occur after a serious analysis. A premature mapping can induce to project's decision mistakes and hence to undesirable reworkings. Thus, it is crucial to investigate criteria that aim the mapping activity and validation.

Investigating and establishing criteria standards constitute a hard task, because some elements which influence decisions of project to their decomposition are: a) staff training; b) problem domain; c) solution domain; d) available technologies; e) budget; and f) schedule. Thus, it is possible to model and implement the same system in different manners.

The necessity of criteria to decomposition was already highlighted by Parnas [7] in 1972. Therefore, it is believed that the success of software development has never been so coupled to decomposition criteria as nowadays, due to the complexity of software and the use of concepts as SoC, Crosscutting Concerns and AO.

4.3 Final Remarks

The major contribution of this paper is the AO software dynamic modeling, as a third unexplored dimension suggested by the authors, base upon business rules variations along with time. It will only emerge after researchers have been untied from OO dogmas and axioms, still considered ingrained into the scientific and technological community.

During the history of science, successful techniques were reused. Recently, computing researchers are applying OO and Modular Programming techniques into areas of AO techniques, e.g. EA.

Even though technological and scientific communities are spending time to reapply old techniques into AO, they need to expend more time identifying, acquiring dynamic as the soul of AO and in the future, build techniques to expand current frontiers of Software Engineering by:

a) Providing dynamic in system models to improve their representation of the real world. For example, classes lack skills to gain and lose attributes, responsibilities, relationships among entities, features inherited and son on. AO can provide that;

b) Expanding frontiers of software engineering towards organization structure, allowing automatic capturing of business

rules from laws, agreements, accords, and other official documents, besides to adapt system using AO to alter dynamics;

c) Expanding software engineering towards Artificial Intelligence using some techniques like Planning and applying AO to alter the dynamics of the system; and

d) Investigating the latent power of overlapping multiple AO layers and benefits to control the dynamics of dynamics.

5. CURRENT WORK

Specific issues about AOSD are in process of investigation by the Software Engineering Group of the Technological Institute of Aeronautics (Instituto Tecnológico de Aeronáutica – ITA), in cooperation with Federal University of Lavras (Universidade Federal de Lavras – UFLA).

Some of the issues raised above constitute the focus of the first author research. There are two focal points in current investigation: a) developing a consistent AO methodology applicable to Analysis and Design phase; and b) integrating the model with enterprise documents that describes business rules and changes system behaviors.

6. CONCLUSIONS

Being one of the newest software engineering paradigms, AO emphasizes that new studies and experiments should be carefully carried out, in order to establish improved methods, techniques and tools applicable to this new way of development.

This paper has presented background information about Early Aspects (EA) and several related work summaries. Araújo et al. [21] and Chitchyan et al. [22] are considered significant references for a first contact as well as for further information about EA.

The major findings of this paper are: a) considering EA investigation, results of related works can be categorized as templates and methodological proposals; and b) considering the AO theory, it is possible to categorize aspects of Structural Dynamics, Behavioral Dynamics, and Data Dynamics.

The categorization aforementioned outlines *how* researchers are looking for solutions to identify and treat Early Aspects. Would these two categories be the only existing ones? Would be other alternatives not seen due to the fact of researchers are tied to previous paradigms?

Aspects categorization emphasizes mainly the dynamic modeling capability of AO to provide mechanisms to modify structure, behavior and data of systems. Considering dynamic like one of the best potentials of AO, authors believe that Early Aspects need to focus relationships among entities.

This paper has presented the state of the art on Early Aspects and some open issues to be investigated by the scientific and technological community. Aiming to develop or improve Early Aspects methodologies, to effectively supply evidences of quality and reliability for analysis and design phases. It presents some

important discussion to help the Early Aspects methodology development, providing better use of the potential of the Aspect Oriented paradigm.

It emphasizes the importance of developing AO methodologies by the scientific and technological community to handle crosscutting concerns, and mapping them from analysis to design phase, which is an important open issue in the AO area.

Most of current EA proposed methodologies adapt OO techniques to AO paradigm, such as catalogs, identification of candidate artifacts (e.g. Aspects), criteria to select candidates, templates and others.

The AO theory capability of modeling the system dynamics needs to be carefully investigated to transform its potential into concrete advantages. To achieve this goal, the scientific and technological community needs to get rid of OO archetypes.

The AO theory and technologies make possible to model system dynamics and, at the same time, facilitate to expand the boundaries of software engineering, enabling to incorporate decision making based upon enterprise documents that describe business rules.

Some open issues, as well as the development of an EA methodology are under investigation, being the motivation and the theme of current and further research works.

7. ACKNOWLEDGMENTS

Authors would like to thank Gian Ricardo Berkenbrock and Ana Rubélia M. L. Resende for their useful comments on this paper.

Thanks also to CAPES, ITA and the Brazilian Ministry of Education, for the financial support to Antônio M. P. de Resende and Fábio Fagundes Silveira.

8. REFERENCES

- [1] Ossher, H., Kaplan, M., Katz, A., Harrison, W., and Kruskal, V., "Specifying subject-oriented composition". TAPOS, 2(3):179–202. Special Issue on Subjectivity in OO Systems, 1992.
- [2] Ossher, H. and Tarr, P., "Using subject-oriented programming to overcome common problems in object-oriented software development/evolution". In International Conference on Software Engineering, ICSE'99, ACM, 1999.
- [3] Miller, S. K., "Aspect Oriented Programming Takes Aim at Software Complexity". Magazine IEEE's Computer, vol.34, no.4, pp.18-21, April 2001.
- [4] Resende, A. M. P. and Silva, C. C., "Programação Orientada a Aspectos em Java". Published by Brasport Livros e Multimídia Ltda, Rio de Janeiro, Março 2005.
- [5] Dijkstra, E.W., "A Discipline of Programming". Prentice-Hall, Englewood Cliffs, N.J., 1976

- [6] Maynard, J., "Programação Modular". Publisher Livros Técnicos e Científicos – LTC, 1976, translated from original version "Modular Programming". Petrocelli Books, New York, 1972.
- [7] PARNAS, D. L. Designing software for ease of extension and contraction. In: International Conference On Software Engineering 3, 1978, Atlanta. Proceedings...Piscataway: IEEE Press, 1978. p. 264–277.
- [8] Sutton Jr., S. M. and Rouvellou, I., "Modelling Software Concerns in Cosmos". Proceedings of the 1st International Conference on Aspect-Oriented Software Development (AOSD), ACM Press, pp.127-133, Enschede, The Netherlands April 2002.
- [9] Moreira, A., Araújo, J. and Brito, I., "Crosscutting Quality Attributes for Requirements Engineering". Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE), Ischia, Italy, ACM Press, pp. 167-174, July 2002.
- [10] Brito, I., Moreira, A. and Araújo, J., "A requirements model for quality attributes". In: Workshop Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design event of the Aspect Oriented Development Software – AOSD, University of Twente, Enschede, The Netherlands, 22-26 de april, 2002.
- [11] Mylopoulos, J., Chung, L., and Nixon, B., "Representing and Using Non-Functional Requirements: A Process-Oriented Approach". IEEE Transactions on Software Engineering, Special Issue on Knowledge Representation and Reasoning in Software Development, Vol. 18(6), pp. 482-497, 1992
- [12] Malan, R. and D. Bredemeyer, "Functional Requirements and Use Cases". White Paper at <http://www.bredemeyer.com/>, 1999.
- [13] Malan, R. and D. Bredemeyer, "Defining Non-Functional Requirements". White Paper at <http://www.bredemeyer.com/>, 2001.
- [14] Lopes, C.V., Bajracharya and S. K., "An analysis of modularity in aspect oriented design". Proceedings of the 4th international conference on Aspect-oriented software, Chicago, Illinois, USA, 2005.
- [15] Eppinger, S. D., Whitney, D. E. and Yassine, A. A., "The Design Structure Matrix Web Site". URL: <http://www.dsmweb.org/> visited at 22/08/2004 and supported by Massachusetts Institute of Technology-MIT and University of Illinois at Urbana Champaign - UIUC DSM Research Teams.
- [16] Monteiro, M. P., "Catalogue of Refactorings for AspectJ". Technical Report UM-DI-GECS-200402, Department of Informatic of Minho, Portugal, December 2004.
- [17] Monteiro, M. P. and Fernandes, J.M., "Pitfalls of AspectJ Implementations of Some of the Gang-of-Four Design Patterns". Proceedings of the Desarrollo de Software Orientado a Aspectos - DSOA'2004 workshop, at the VIII Jornadas de Ingeniería de Software y Bases de Datos - JISBD'2004, Málaga, Spain, November 2004.
- [18] Gamma, E., Helm, R., Johnson R. and Vlissides, J., "Design Patterns – Elements of Reusable Object-Oriented Software". Addison-Wesley, 1994.
- [19] Monteiro, M.P. and Fernandes, J.M., "Towards a catalog of aspect-oriented refactorings". Proceedings of the 4th international conference on Aspect-Oriented Software Development - AOSD, Chicago, Illinois, USA, 2005.
- [20] Hannemann, J. and Kiczales, G., "Design Pattern Implementation in Java and AspectJ". Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications - OOPSLA, Seattle, Washington, USA, Novembro de 2002.
- [21] Araújo, J., Baniassad, E., Clements, P., Moreira, A., Rashid, A. and Tekinerdogan, B., "Early Aspects: The Current Landscape". URL: <http://www.early-aspects.net>. Visited in May 27, 2005.
- [22] Chitchyan, R., Rashid, A., Sawyer, P., Garcia, A., Alarcon, M.P., Bakker, J., Tekinerdogan, B., Clarke, S. and Jackson, A., "Survey of Aspect-Oriented Analysis and Design Approaches". URL: <http://www.aosd-europe.net/documents>. Visited in June, 29, 2005.
- [23] Grundy, J., "Aspect-oriented Requirements Engineering for Component-based Software Systems". Proceedings of the 4th IEEE International Symposium on Requirements Engineering, IEEE CS Press, pp. 84-91, Limerick, Ireland, June 1999.
- [24] Dingwall, S. and Finkelstein A., "From Requirements to Monitors by way of Aspects". Early Aspects 2002: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop of the 1st International Conference on Aspect-Oriented Software Development (AOSD), Enschede, The Netherlands, April 2002.
- [25] Rashid, A., Sawyer, P., Moreira, A. and Araújo, J., "Early Aspects: A Model for Aspect-Oriented Requirements Engineering". Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE), IEEE CS Press, pp. 199-202, Essen, Germany, September 2002.
- [26] Rashid, A., Moreira, A. and Araújo, J., "Modularization and Composition of Aspectual Requirements". Proceedings of the 2nd International Conference on Aspect-Oriented Software Development (AOSD), ACM Press, pp. 11-20, Boston, USA, March 2003.
- [27] Katz, S. and Rashid, A., "From aspectual requirements to proof obligations for aspect-oriented systems". In 12th IEEE International Conference Requirements Engineering (RE), (Kyoto, Japan). IEEE, September 2004.

- [28] Araújo J., and Coutinho, P., "Identifying Aspectual Use Cases Using a Viewpoint-Oriented Requirements Method". Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop of the 2nd International Conference on Aspect-Oriented Software Development (AOSD), Boston, USA, March 2003.
- [29] Whittle, J., Araújo J. and Kim, D., "Modeling and Validating Interaction Aspects in UML". 4th AOSD Modeling With UML Workshop, workshop of the 6th International Conference on the Unified Modeling Language - <<UML>> 2003, San Francisco, USA, 20 October 2003.
- [30] Whittle, J. and Araújo J., "Scenario Modeling with Aspects". IEE Proceedings - Software, Special Issue on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design. Editors: A. Rashid, A. Moreira, B. Teknirdogan. August 2004.
- [31] Whittle, J. and Schumann, J., "Generating Statechart Designs from Scenarios". Proceedings of the 22nd International Conference on Software Engineering (ICSE), IEEE CS Press, pp.314-323, Limerick, Ireland, June 2000.
- [32] Baniassad, E., and Clarke, S., "Theme: An approach for aspect-oriented analysis and design". 26th International Conference on Software Engineering (ICSE), IEEE Press, Edinburgh, Scotland, May 2004.
- [33] Baniassad, E. and Clarke, S., "Investigating the Use of Clues for Scaling Document-Level Concern Graphs". Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design, workshop of the OOPSLA 2004, Vancouver, Canada, October 2004.
- [34] Yuy, Y., Leite, J. and Mylopoulos, J., "From goals to aspects: discovering aspects from requirements goal models". Proceedings of the 12th IEEE International Requirements Engineering Conference (RE), IEEE CS Press, Kyoto, Japan, September 2004.
- [35] Brito, I., and Moreira, A., "Advanced Separation of Concerns for Requirements Engineering". VIII Jornadas de Ingeniería de Software y Bases de Datos (JISBD), Alicante, Spain, 12-14 November 2003.
- [36] Brito, I., Aspect Oriented Requirements Engineering. In Doctoral Symposium 7th International Conference on the Unified Modeling Language - UML, Lisbon, PORTUGAL, 2004.
- [37] Moreira, A. Rashid and Araújo, J., "Multi-Dimensional Separation of Concerns in Requirements Engineering". The 13th International Conference on Requirements Engineering (RE'05), IEEE Computer Society, August 29, September 2, Paris, France, 2005.
- [38] Sousa, G. M. C. and Castro, J., "Towards a Goal-Oriented Requirements Methodology Based in the Separation of Concerns Principle". Proceedings of the 6th International Workshop on Requirements Engineering - WER'03, v. 1. p. 223-239, Piracicaba, Brazil, 2003.
- [39] Sousa, G., Soares, S., Borba, P. and Castro, J. "Separation of Crosscutting Concerns from Requirements to Design: Adapting an Use Case Driven Approach". Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design, workshop of the 3rd International Conference on Aspect-Oriented Software Development, Lancaster, UK, March 2004.
- [40] Araújo, J., Whittle J. and Kim, D., "Modeling and Composing Scenario-Based Requirements with Aspects". Proceedings of the 12th IEEE International Requirements Engineering Conference (RE), Kyoto, Japan, IEEE CS Press, September 2004.