The Journal of Systems and Software 86 (2013) 905–933

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

The crosscutting impact of the AOSD Brazilian research community

Uirá Kulesza[a], Sérgio Soares[b,*], Christina Chavez[c], Fernando Castor[b], Paulo Borba[b], Carlos Lucena[d], Paulo Masiero[e], Claudio Sant'Anna[c], Fabiano Ferrari[f], Vander Alves[g], Roberta Coelho[a], Eduardo Figueiredo[h], Paulo F. Pires[i], Flávia Delicato[i], Eduardo Piveta[j], Carla Silva[b], Valter Camargo[f], Rosana Braga[e], Julio Leite[d], Otávio Lemos[k], Nabor Mendonça[l], Thais Batista[a], Rodrigo Bonifácio[g], Nélio Cacho[a], Lyrene Silva[a], Arndt von Staa[d], Fábio Silveira[k], Marco Túlio Valente[h], Fernanda Alencar[b], Jaelson Castro[b], Ricardo Ramos[m], Rosangela Penteado[f], Cecília Rubira[n]

[a] *Universidade Federal do Rio Grande do Norte, Brazil*
[b] *Universidade Federal de Pernambuco, Brazil*
[c] *Universidade Federal da Bahia, Brazil*
[d] *Pontifícia Universidade Católica do Rio de Janeiro, Brazil*
[e] *Universidade de São Paulo – São Carlos, Brazil*
[f] *Universidade Federal de São Carlos, Brazil*
[g] *Universidade de Brasília, Brazil*
[h] *Universidade Federal de Minas Gerais, Brazil*
[i] *Universidade Federal do Rio de Janeiro, Brazil*
[j] *Universidade Federal de Santa Maria, Brazil*
[k] *Universidade Federal de São Paulo – São José dos Campos, Brazil*
[l] *Universidade de Fortaleza, Brazil*
[m] *Universidade Federal do Vale do São Francisco, Brazil*
[n] *Universidade Estadual de Campinas, Brazil*

ARTICLE INFO

Article history:
Received 21 January 2012
Received in revised form 18 June 2012
Accepted 14 August 2012
Available online 25 August 2012

Keywords:
Aspect-Oriented Software Development
Modularity
Research impact

ABSTRACT

Background: Aspect-Oriented Software Development (AOSD) is a paradigm that promotes advanced separation of concerns and modularity throughout the software development lifecycle, with a distinctive emphasis on modular structures that cut across traditional abstraction boundaries. In the last 15 years, research on AOSD has boosted around the world. The AOSD-BR research community (AOSD-BR stands for AOSD in Brazil) emerged in the last decade, and has provided different contributions in a variety of topics. However, despite some evidence in terms of the number and quality of its outcomes, there is no organized characterization of the AOSD-BR community that positions it against the international AOSD Research community and the Software Engineering Research community in Brazil.
Aims: In this paper, our main goal is to characterize the AOSD-BR community with respect to the research developed in the last decade, confronting it with the AOSD international community and the Brazilian Software Engineering community.
Method: Data collection, validation and analysis were performed in collaboration with several researchers of the AOSD-BR community. The characterization was presented from three different perspectives: (i) a historical timeline of events and main milestones achieved by the community; (ii) an overview of the research developed by the community, in terms of key challenges, open issues and related work; and (iii) an analysis on the impact of the AOSD-BR community outcomes in terms of well-known indicators, such as number of papers and number of citations.
Results: Our analysis showed that the AOSD-BR community has impacted both the international AOSD Research community and the Software Engineering Research community in Brazil.

1. Introduction

The emergence of a new research area is often closely associated with problems and challenges faced by a relatively stable community of practitioners and practices. Researchers depart from well-established concepts and techniques to explore unknown

* Corresponding author.

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.08.031

territories and to discover new avenues that may lead to promising solutions to existing problems. This is certainly a valid common ground for research on Aspect-Oriented Software Development (AOSD) (Filman et al., 2005).

AOSD is a new development paradigm that aims to promote advanced separation of concerns and modularity throughout the software development lifecycle. AOSD has emerged on the shoulders of successive generations of software development paradigms, each of them adhering to fundamental principles such as separation of concerns (Dijkstra, 1976) and modularity (Parnas, 1972), and supported by programming languages, development methods, practices, tools, body of knowledge, and community. In scenarios of increasing complexity and unavoidable need for change, the general goal of these paradigms has been to support the development of software solutions to real-world problems, in a way that promotes internal quality attributes such as understandability, ease of change and reuse of software artifacts.

The *object-oriented* (OO) paradigm has played the role of the dominant development technology for the last two or three decades, with broadly recognized benefits. While the OO paradigm overcomes the limitations of previous paradigms to modularize concerns with respect to their support for encapsulation, information hiding (Parnas, 1972), and polymorphism (Cardelli and Wegner, 1985), it faces its own limitations for modularizing concerns that address global constraints and systemic properties, such as synchronization, persistence, error handling, logging mechanisms, among many others. In fact, these limitations and difficulties to modularize certain dimensions of concerns may not be specific to OO, but rather be part of a general problem that has been coined “The Tyranny of Dominant Decomposition” (Tarr et al., 1999).

Concerns that are difficult to modularize have been called *crosscutting concerns* (CCCs) since they naturally cut across the boundaries of modular units that implement other concerns. Without proper means for separation and modularization, crosscutting concerns tend to be scattered over a number of modular units and tangled up with other concerns, throughout software development activities and across different artifacts. The natural consequences are lower cohesion and stronger coupling between modular units, and reduced degrees of comprehensibility, evolvability, and reusability of software artifacts.

In the last 15 years, research on AOSD has boosted around the world, with initial focus on programming languages and tools (Tarr et al., 1999; Aksit et al., 1994; Bergmans and Aksit, 2001; Harrison and Ossher, 1993; Lieberherr, 1996; Lieberherr et al., 2001; Kiczales et al., 1997, 2001; Ossher and Tarr, 2001; Lopes, 2005). In the last decade, the focus has shifted towards different software development activities (Filman et al., 2005). Recently, research on AOSD has broadened its goals to address software modularity in general, yet with a distinctive emphasis on modular structures that cut across traditional abstraction boundaries (AOSD, 2011).

The AOSD-BR research community (AOSD-BR stands for AOSD in Brazil) emerged in the last decade, and has contributed to several Software Engineering (SE) research areas, including requirements engineering, analysis and design, languages, implementation methods and techniques, modeling, testing, tools, and assessment. In the last few years, this community has been involved in collaborative research networks, inside and outside Brazil, has published a number of papers in several top SE conferences,[1] and has educated a new generation of researchers.

Our previous work (Chavez et al., 2011)[2] provided a preliminary characterization of the research developed by the AOSD-BR community in the last decade and presented initial evidence about the impact of the AOSD-BR research outcomes. However, there was no comprehensive analysis of such outcomes against those of related counterparts, for instance, the international AOSD research community and the Brazilian SE research community. Moreover, the characterization of the overall contributions of the AOSD-BR research, with respect to existing research challenges, open issues and related work, could be enhanced.

In this paper, our main goal is to characterize the AOSD-BR community with respect to the research developed in the last decade, confronting it with the AOSD international community and the Brazilian Software Engineering community. This work extends the scope, refines and organizes our previous characterization (Chavez et al., 2011), providing improved analysis of the research outcomes and lessons learned from the AOSD-BR research community.

Three research questions were defined to drive our characterization of the AOSD-BR research community achievements and detailed comparisons:

(i) What is the impact of the research developed by the AOSD-BR community compared to that of the international AOSD research community ?
(ii) What is the impact of the research developed by the AOSD-BR community compared to that of the Brazilian SE community in the international context?
(iii) What is the impact of the research developed by the AOSD-BR community compared to that of the Brazilian SE community in the national context?

The expected contributions of this work are threefold. Firstly, the documented knowledge about the creation of the AOSD-BR research community and initiatives that promoted its evolution can be generalized to serve as a roadmap for researchers willing to foster new research areas and communities around the world. Second, the enhanced, general characterization of the state-of-the-art in some areas of AOSD can be useful for young researchers that are starting their research on AOSD, by unveiling, for instance, topics that have not been properly exploited yet. It can attract new researchers and fundings for the area as well. Finally, those interested in academic collaboration with Brazilian researchers can benefit from our characterization of the state-of-art of AOSD-BR research, to identify research groups, their main publications, and opportunities for collaboration.

This paper is structured as follows. Section 2 presents, in more detail, the AOSD-BR research community timeline, highlighting important events and initiatives, such as the inception and organization of the 1st Brazilian Workshop on AOSD (WASP), held in 2004, and the organization of the 10th International Conference on AOSD in Brazil, held in 2011, among others.

Section 3 presents an overview of the research work developed by AOSD-BR community in several prominent SE areas. For each research area, we highlight some key challenges, describe briefly how they have been addressed by the AOSD-BR community and related research conducted by the international community, and finally, point out some open issues.

Section 4 presents a discussion on the growth, impact, and quality of the AOSD-BR research outcomes in terms of several indicators, such as the number of publications in journals and conferences over the last decade, the number of papers published in top international

[1] By top SE conferences, we mean conferences broadly recognized as major SE venues. The conferences we considered are listed in Section 4.2.1.

[2] This paper was published at the Special Track “SBES is 25” – organized to celebrate the 25th Anniversary of the Brazilian Symposium on Software Engineering (SBES).

SE conferences, and the number of citations to these papers. Several original contributions are provided. A wide search was performed in the proceedings of six major conferences, looking for AOSD-related papers published since 1999 (year of the first Brazilian publications about AOSD) and the number of citations, in order to assess the AOSD-BR impact in the AOSD international context. The data gathered allowed us to provide several additional analyses.

Finally, Section 5 presents our concluding remarks, highlighting main results, presenting lessons learned and possible venues for future work in the AOSD field.

2. AOSD-BR timeline

In the last 10 years, the AOSD Brazilian (AOSD-BR) research community has emerged and matured, having developed high-quality research work in different AOSD-related research topics. This section presents the AOSD-BR community timeline, highlighting its main events and initiatives. Fig. 1 illustrates a timeline of these events. The main milestones of the AOSD-BR community are presented above the axis, along with some important milestones of the international AOSD community, which appear below the axis.

2.1. The first steps

Brazilian researchers have been working on AOSD-related topics since 1998. The first AOSD-BR papers were published at the Brazilian Symposium on Software Engineering (SBES) in 1999 (Sztajnberg et al., 1999b,a), just two years after the publication of the paper on aspect-oriented programming (AOP) (Kiczales et al., 1997) at ECOOP. In 2002, the first international conference on AOSD (2002) was held in Europe. In that same year, the first international high-quality paper from Brazilian researchers working on AOSD (Soares et al., 2002) was published at OOPSLA. The year of 2004 hosted the defenses of the first PhD theses on AOSD (Chavez, 2004; Garcia, 2004; Soares, 2004) in Brazil. Interaction among researchers was limited to messages exchanged by means of the AOSD-BR list (AOSD-BR, 2002), a mailing list created in 2002 to promote the discussion of research and technical issues.

2.2. Bringing together AOSD researchers from different Brazilian universities

The organization of the Brazilian Workshop on Aspect-Oriented Software Development (WASP), in 2004, is probably the starting milestone of the AOSD-BR community. That was definitely the first effort to bring together the AOSD researchers from different Brazilian universities in a single event. WASP (2004) was organized as a one-day workshop at SBES. The workshop was a great success, attracting a large number of participants (70) and submissions (40). The technical program of WASP 2004 included the presentation of technical papers and posters, and focused discussion groups. WASP also featured an international keynote speaker, Professor Awais Rashid from Lancaster University – UK, to foster the interaction of the AOSD-BR community with renowned international AOSD researchers.

WASP had a fundamental role in congregating the AOSD-BR community and motivating researchers from different Brazilian universities to create their respective research groups in this particular area of study. Another important outcome from WASP was the definition of a catalog of Brazilian Portuguese terms for the main AOSD concepts originally defined in English. Additional information and a complete report on WASP can be found elsewhere (WASP, 2004).

2.3. Encouraging the cooperation between Brazilian research groups

WASP 2004 provided a unique opportunity for several senior and young Brazilian researchers to meet each other and their respective research work by means of different discussion groups organized as part of its technical program. Two more WASP workshops (WASP, 2005, 2006) were held at SBES 2005 (Uberlândia – MG) and SBES 2006 (Florianópolis – SC), respectively. The invited speakers actively participated in the discussions held at the workshops, providing key feedback and insights to AOSD-BR researchers.

WASP 2005 and WASP 2006 contributed to the consolidation of new AOSD research groups in Brazil. In addition, they also encouraged the first collaborations between AOSD-BR research groups, which opened the way to the publication of a series of interesting research results in the following years. Examples of such results are: (i) the development of several empirical studies that focused on the quantitative comparison of aspect-oriented and object-oriented implementations (Figueiredo et al., 2008a; Greenwood et al., 2007), in a collaboration between PUC-Rio and UFPE researchers; (ii) the development of aspect-oriented software architecture languages and tools (Batista et al., 2006a,b; Garcia et al., 2006; Chavez et al., 2007, 2009; Molesini et al., 2010), in a joint effort involving PUC-Rio, UFBA and UFRN researchers; and (iii) the development of aspect-oriented approaches to modularize software frameworks and product lines (Kulesza et al., 2006b,d), in a collaboration between PUC-Rio and UFPE researchers. Finally, cooperation among AOSD-BR research groups also led to the organization, in 2005, of a special issue on AOSD in the Journal of the Brazilian Computer Society (JBCS), the main publication vehicle of the Brazilian Computer Science research community.

2.4. Crossing the borders

In 2005, AOSD-BR members published their first two papers (Garcia et al., 2005; Cole and Borba, 2005) at the AOSD conference. In 2006, the first volumes of the Transactions on Aspect-Oriented Software Development (TAOSD) journal (TAOSD, 2006) were published with the edition and organization of renowned researchers from the international community. It is interesting to notice that the first volume of TAOSD (TAOSD, 2006) already had contributions from the AOSD-BR community.

2.5. Promoting the cooperation between the AOSD-BR community and their South America neighbors

In 2006, the Latin American Research Network on AOSD (LatinAOSD, 2007) was created, involving several institutions from Brazil (PUC-Rio, UFBA, UFMG, UFPE, UFRGS, UFRJ, UFRN, USP-São Carlos, IME-RJ), Chile (UChile, UTFSM), Argentina (UNICEN, UBA) and Colombia (UniAndes). LatinAOSD was funded by the Brazilian National Council for Scientific and Technological Development (CNPq). The main goal of LatinAOSD was to establish a forum to promote the integration and cooperation between AOSD research groups from South America. The LatinAOSD network pushed WASP to a broader scope and, in 2007, WASP became LA-WASP, the Latin American Workshop on AOSD, organized as a two-day workshop at SBES 2007, held in João Pessoa – PB. LA-WASP was also a great success, receiving submissions from all the institutions that were part of the LatinAOSD network. It is also important to mention that many senior researchers from universities that were part of the LatinAOSD network (from Chile, Argentina and Colombia) were invited to join the technical and organizing committees of the workshop.

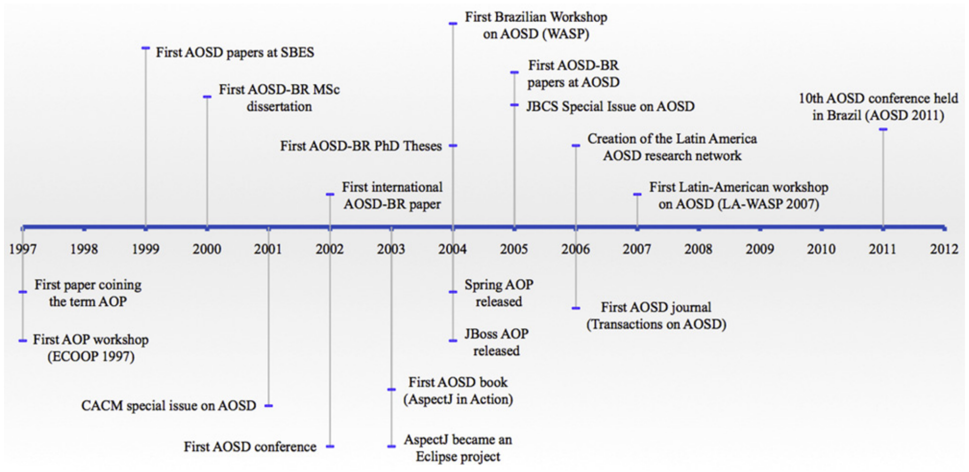
First Brazilian Workshop on AOSD (WASP)
First AOSD papers at SBES
First AOSD-BR papers at AOSD
First AOSD-BR MSc dissertation
JBCS Special Issue on AOSD
10th AOSD conference held in Brazil (AOSD 2011)
First AOSD-BR PhD Theses
Creation of the Latin America AOSD research network
First international AOSD-BR paper
First Latin-American workshop on AOSD (LA-WASP 2007)
1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
First paper coining the term AOP
Spring AOP released
First AOSD journal (Transactions on AOSD)
First AOP workshop (ECOOP 1997)
JBoss AOP released
First AOSD book (AspectJ in Action)
CACM special issue on AOSD
AspectJ became an Eclipse project
First AOSD conference

Fig. 1. AOSD and AOSD-BR timeline.

Table 1
Invited researchers to workshops organized by the AOSD-BR community.

Issue	Invited speaker
WASP 2004	Awais Rashid (Lancaster University, UK)
WASP 2005	Cristina Lopes (University of California, Irvine)
WASP 2006	Mira Mezini (Darmstadt University, Germany)
LA-WASP 2007	Theo D'Hondt (Vrije Universiteit Brussel, Belgium)
LA-WASP 2007	Paulo Borba (Federal University of Pernambuco, Brazil)
LA-WASP 2008	Kevin Sullivan (University of Virginia, USA)
LA-WASP 2008	Marco Tulio Valente (Federal University of Minas Gerais, Brazil)
LA-WASP 2009	Jon Whittle (Lancaster University, UK)
LA-WASP 2009	Alessandro Garcia (PUC-Rio, Brazil)
LA-WASP 2010	Éric Tanter (University of Chile, Chile)
LA-WASP 2010	Sergio Soares (Federal University of Pernambuco, Brazil)
LA-WASP 2011	Mehmet Aksit (University of Twente, Netherlands)

2.6. *Inviting international senior professors and researchers*

The subsequent editions of LA-WASP were also held in conjunction with SBES (2008, 2009, 2010 and 2011). These workshops followed the same philosophy of the previous WASP and LA-WASP workshops in promoting the interaction and cooperation between senior and young researchers from different universities in South America. In addition, one or two international keynote speakers were invited to present their latest research results and to actively participate in the workshop, which not only contributed to improve the quality of the discussions, but also helped to promote international cooperation with the AOSD-BR community. Table 1 shows the complete list of international and national speakers of LA-WASP and WASP along the years. Brazilian researchers were also invited to give keynote talks due to the increasing importance of their research in international settings.

Table 2
Submissions, acceptance rate and attendance at workshops organized by the AOSD-BR community.

Issue	Submissions	Acceptance	Participants
WASP 2004	41	41.46%	70
WASP 2005	21	42.86%	39
WASP 2006	26	53.85%	45
LA-WASP 2007	31	54.84 %	50
LA-WASP 2008	18	50.00 %	30
LA-WASP 2009	21	47.62 %	40
LA-WASP 2010	14	85.71 %	44
LA-WASP 2011	19	36.84 %	33

Table 2 shows the numbers of submissions, acceptance rate and participants of WASP and LA-WASP along the years.

2.7. *Promoting the cooperation between the AOSD-BR community and international research groups*

The invitation of international AOSD researchers to give talks at the LA-WASP and WASP series of workshops has contributed to motivate and promote the cooperation between the AOSD-BR community and international research groups. Many joint research efforts were developed collaboratively between Brazilian and international researchers, such as: (i) the development of empirical quantitative case studies of AO and OO implementations (Figueiredo et al., 2008a; Greenwood et al., 2007; Ferrari et al., 2010), in a collaboration between Lancaster University, PUC-Rio, UFPE and USP-SC researchers; (ii) the assessment of aspect-oriented implementations in AspectJ and CaesarJ in the modularization feature binding times (Andrade et al., 2011), in a collaboration between Darmstadt University and UFPE researchers; and (iii) the development of domain-specific languages to manage software variabilities (Zschaler et al., 2009; Alférez et al., 2009; Heidenreich et al., 2010), in a collaboration between Lancaster University, New University of Lisbon and UFRN researchers.

In addition, the close interaction with international senior researchers also promoted the active participation of many Brazilian researchers in relevant international projects, such as: (i) AOSD-Europe (AOSD-Europe Network of Excellence) (AOSD Europe, 2002), a large research project funded by the European Commission with the participation of nine European universities and two industrial organizations that aimed at harmonizing and integrating the research, training and dissemination activities of its members in order to address fragmentation of AOSD activities in Europe and strengthen innovation in areas such as aspect-oriented analysis and design, formal methods, languages and applications of AOSD techniques in ambient computing; and (ii) AMPLE (Aspect-Oriented Product Line Engineering) (AMPLE, 2006), a three-year research project funded by the European Commission that aimed at proposing a development methodology to modularize variations in software product lines through the integrated adoption of model-driven engineering and aspect-oriented software development.

Brazilian post-graduate students have also benefited from this cooperation, by joining international universities to develop their PhD thesis (either in full or partially) or to work as post-doc researchers after finishing their PhD thesis in Brazil.

Last but not least, AOSD-BR researchers now participate in program committees of important international conferences (AOSD and ICSE) and have organized and participated in the technical program committee of several workshops (Early Aspects, Assessment of Contemporary Modularization Techniques – ACoM, Workshop on Empirical Evaluation of Software Composition Techniques – ESCOT) in conjunction with top SE conferences.

2.8. Hosting the AOSD conference in Brazil

In 2011, the AOSD-BR community had another important milestone: the 10th International Conference on Aspect-Oriented Software Development (AOSD, 2011) was held in Brazil (Porto de Galinhas – PE). Several senior researchers from the AOSD-BR community were in charge of organizing AOSD 2011. The conference had 138 attendees (slightly above the average attendance), of which 65 (47%) were from Brazilian institutions. The success of the AOSD 2011 conference coupled with the relatively large number of Brazilian researchers in attendance are testimony to the important role played by the AOSD-BR community in the international context.

3. AOSD-BR research areas

This section presents an overview of the research work developed in the context of Aspect-Oriented Software Development in several prominent Software Engineering areas. The perspective taken is from the synergies and interplay between aspects and other SE areas, and thus it assumes that the interaction can be in both directions with AOSD helping to address key issues in other SE areas and vice-versa. In particular, this overview focuses on research topics that have been exploited by the AOSD-BR community in the last decade, organized in the following sections: Requirements and Architecture, Modeling and Model-based Techniques, Exception Handling, Refactoring, Testing, Metrics, and Software Product-lines. Each of the following sections presents a research topic, some of its associated challenges, a brief summary of research outcomes (from AOSD-BR and also from the international AOSD community) and open issues.

3.1. Early aspects

The term “Early Aspects” (EA) is used to describe issues related to crosscutting concerns in both software requirements and software architecture. Preliminary work on Aspect-Oriented Requirements Engineering (AORE) (Grundy, 1999; Rashid et al., 2002; Moreira et al., 2002) pointed out that several modeling approaches failed to explicitly handle the crosscutting nature of some requirements. These crosscutting concerns produce scattered and tangled representations that are hard to understand, maintain, and reuse. In this context, the Requirements Engineering community started to address modularization of crosscutting concerns, stressing the importance of its early awareness. Similarly, with the emergence of AOSD, the Early Aspect Workshop started in conjunction with the 1st International Conference on Aspect-Oriented Software Development, and its sub-title says: “Aspect-Oriented Requirements Engineering and Architecture Design”. The need for explicitly considering aspects in architecture was detailed in works such as (Kande and Strohmeier, 2000; Katara and Katz, 2003; Mens, 2002). It is important to represent crosscutting concerns at different abstraction levels of the software process, and at the architectural level this representation can help to avoid the architectural erosion (Perry and Wolf, 1992) problem over the lifetime of a system. Also the International Conference on Software Engineering (ICSE) promoted, during several years, an Early Aspect Workshop with the aim of fostering the maturation of Early Aspects as a discipline.

The main challenges, the AOSD-BR contributions and open issues related to early aspects are following summarized.

3.1.1. Key challenges

There are many challenges ahead of early aspects research, especially in the context of:

(i) *Crosscutting concerns identification and representation*: Which strategies should be used for the elicitation of crosscutting concerns? How we could evaluate these strategies? What are the representations that bridge the gap among requirements and architecture as well as how we could integrate them with existing representations? Finally, questions regarding symmetrical and asymmetrical approaches needed to be better evaluated.
(ii) *Trade-offs in crosscutting concerns composition*: Which methods should be used for determining, at requirements and architecture levels, the impacts of a given crosscutting concern over the whole system? Which design techniques could be used in order to avoid the preponderance of certain crosscutting concerns?
(iii) *Management of architectural concerns, specially crosscutting architectural concerns*: What is the impact of the crosscutting concerns with respect to the architectural evolution and degradation? How do the crosscutting concerns (and their representation at the earlier stages) influence the selection of variants of a system? Questions arising from configuration management of such representations do also need further study.
(vi) *Strategies to improve the traceability to other activities and related models*: Making aspects explicit in advance impacts the complexity of requirements and architectural descriptions. As such, which tools tools and automate support are necessary for managing those descriptions? Questions on how to integrate traceability features to such representations deserves special attention.

3.1.2. Addressing the challenges

To address the aforementioned challenges, the AOSD-BR community has been working in different research projects dealing with Early Aspects in two main areas: Aspect-Oriented Requirements Engineering (AORE) and Architectural Aspects. The research works on these areas have investigated and explored the challenges of identification, representation, composition and traceability of crosscutting concerns in different ways of requirements and architectural specifications. In addition, the AOSD-BR community has also contributed to address the challenge of management of architectural concerns. Next we present and discuss these research works.

Aspect-Oriented Requirements Engineering: Sousa et al.’s work Sousa et al. (2003) was the first one to address AORE in Brazil. The authors used the NFR-Framework (Chung et al., 2000) along with use cases to explicitly represent crosscutting requirements separately from non-crosscutting ones and to compose them in a non-invasive way.

Other researchers addressed the modeling (Silva and Leite, 2005a; Alencar et al., 2008) and the evaluation of AO requirements (Ramos et al., 2006), the mapping of crosscutting concerns between different modeling languages, the reuse of crosscutting requirements (do Prado Leite et al., 2005), and the elicitation of crosscutting requirements (Sampaio et al., 2005; Antonelli et al., 2010).

In the context of requirements modeling, an extension of the i* (iStar) modeling language (Yu, 1997), called Aspectual i*, was proposed to support the separation of crosscutting concerns (Alencar et al., 2008). i* models capture social and intentional characteristics, while Aspectual i* defines rules to systematically identify, modularize and compose crosscutting concerns, reducing the complexity of i* models and making their maintenance and evolution easier (Alencar et al., 2008, 2010). A tool was proposed to identify crosscutting concerns in i* and evaluated using the Health Watcher benchmark (Monteiro et al., 2007).

Aspectual i* was also used as the basis for the identification of aspectual use cases, as described by Rojas et al. (2009), and, in the Software Product Lines (SPL) context, to capture common and variable features and thus to facilitate the selection among product configurations (Silva et al., 2008).

In addition, Brazilian AORE researchers also contribute with approaches (as well as tool support) for modularizing features in both use case scenarios (Bonifácio and Borba, 2009) and business processes models (Machado et al., 2011). Regarding the representation of features in use case scenarios, case studies and controlled experiments revealed that, although the use of MSVCM (Modeling Scenario Variability as Crosscutting Mechanisms) (Bonifácio and Borba, 2009) improves the modularity of feature specifications, it increases the time required to extract these feature specifications from the requirements documents of existing products. Actually, the benefits of MSVCM arise when evolving the specifications of a product line, particularly when the analysts are experienced in aspect-orientation.

Also regarding modeling, a meta-model for AO requirements languages was defined (Silva and Leite, 2005a). Three modeling languages were defined upon this meta-model:

(i) AO-BPM (Cappelli et al., 2009, 2010), which is an extension of the BPM notation to describe business processes, to improve modularity, and to make business processes easier to understand and reuse;
(ii) AOV-graph (Silva and Leite, 2005b), which is an extension of V-graph (a goal-oriented language used to model early aspects (Yu et al., 2004)) to represent the relationships among requirements. AOV-graph added a new type of relationship, named crosscutting relationship, in order to separate concerns properly. AOV-graph has been used in empirical studies with the Health Watcher and Mobile Media benchmarks (Greenwood et al., 2007; Figueiredo et al., 2009c; Chitchyan et al., 2009); and
(iii) More recently, AOV-graph has been extended to PL-AOVgraph, in a strategy to automatically map feature models in the context of SPL – see also Section 3.7 – into PL-AOV-graph and vice versa (Santos et al., 2011).

Regarding requirements evaluation, an approach to evaluate the quality of AO requirements models was presented by Ramos et al. (2006). In addition, aspect concepts have been used to reduce duplication, tangling and scattering in requirements documents (Ramos et al., 2008).

Architectural aspects: The work from Batista et al. (2006a) pioneered the discussions on some fundamental issues about the integration of AOSD and architectural description languages (ADLs). The authors advocated that no new architectural abstractions were needed to represent aspects and that regular components could be used for this purpose.

They proposed a composition model based on existing architectural abstractions – connectors and configuration – to support the definition of crosscutting composition facilities and quantification mechanisms. The AspectualACME (Garcia et al., 2006; Batista et al., 2006b) AO ADL was developed following this approach. AspectualACME supports improved composability by means of aspectual connectors.

More recently, AspectualACME has been extended to PL-AspectualACME (Adachi et al., 2009; Barbosa et al., 2011), which proposes an architectural style for the definition of software product lines architectures and also a strategy for the instantiation of specific products derived from the architecture. PL-AspectualACME uses the original abstractions of ACME and Armani's formal constraints in order to guarantee well-formed descriptions. Specific product line architectures can be derived from the architectural style. PL-AspectualACME was evaluated with two realistic case studies: Mobile Media (Figueiredo et al., 2009c) and GingaForAll (Saraiva et al., 2010), the SPL architectural description of the Brazilian Terrestrial Digital TV System middleware (Ginga).

In the context of software architecture evolution, Chavez et al. (2009) proposed *style-based composition*, a new flavor of aspect composition at the architectural level based on architectural styles that addresses the pointcut fragility problem (Kellens et al., 2006).

The authors proposed style-based join point models and provided a pointcut language that supports the selection of join points based on style-constrained architectural models. Stability and reusability assessments of the proposed style-based composition model were carried out through three case studies involving different styles (Chavez et al., 2009). The interplay of style-based pointcuts and some style composition techniques was also discussed (Molesini et al., 2010) using four different architectural styles applied to an evolving multi-agent architecture. The works in Sande et al. (2006), Oliveira (2007) provided a notation to represent and visualize architectural aspects.

Regarding how crosscutting concerns impact architectural evolution and the selection of architecture alternatives, Sant'Anna et al. (2007), Sant'Anna (2008) defined a suite of metrics that were used to evaluate these two points. These metrics quantify attributes such as concern scattering, concern tangling, concern interaction, concern-based cohesion, and concern coupling based on information from architecture descriptions. These metrics were used in empirical studies which compared these attributes in different architecture alternatives, mainly AO and non-AO architectures, of the same systems (Sant'Anna et al., 2007; Sant'Anna, 2008). Also these empirical studies compared how the crosscutting degree of architectural concerns evolved as the systems evolved (Sant'Anna, 2008).

In the context of model transformations, a bidirectional mapping between requirements (in AOV-graph) and software architecture (in Aspectual-ACME) was defined by Medeiros et al. (2007) and Silva et al. (2007) and implemented using the MARISA tool (Medeiros et al., 2007).

In summary, the AOSD-BR community focused on applying the benefits of aspect-orientation in the context of architectural description languages, product line architecture, style-based composition, and the visualization of architectural aspects.

3.1.3. Comparison with research conducted by the international community

The AORE Brazilian community has been working to address the challenges related to the identification, representation and composition of crosscutting concerns at the requirements and architecture abstraction levels. Participation in international forums, as well as direct international collaboration has allowed a positive change of ideas. Many Brazilian approaches were influenced by the work of Rashid et al. (2002). In this work, the authors propose a general model that supports separation of crosscutting functional and non-functional requirements. Regarding the challenge (ii), we can highlight the work presented by Moreira et al. (2005), which proposes a technique to separate and compose requirements regardless of their functional or non-functional nature. In particular, this work proposes a guide to perform analysis of requirements-level trade-offs and provides insights into various architectural choices available to fulfill a particular requirement. The work presented by Rashid and Moreira (2006) is a step forward in relation to the challenge (i), since the authors argue that domain models have aspects which need to be identified and modularized effectively.

The drawbacks of syntax-based AO compositions have already been discussed (Chitchyan et al., 2007) from the perspective of AO requirements engineering. The similarities with our work are the discussions about the disadvantages of syntax-based compositions in contrast with the expressiveness of semantics-based composition in early aspects. Existing AO approaches at the architecture level do not support style-based pointcut specifications.

Kellens et al. (2006) proposed model-based pointcuts to address the fragile pointcut problem by replacing the intimate dependency of pointcut definitions on the base program by a more stable dependency on a conceptual model of the program. The authors argue that the model-based pointcut definitions are less likely to break upon evolution because they are no longer defined directly in terms of the actual base code structure. Therefore, the fragile pointcut problem is transferred to a more conceptual (and probably more stable) level and turned into a synchronization problem between the conceptual model and the base code whenever this code evolves.

The international AOSD community has proposed several AO-ADLs, but most of them are heavyweight solutions, in contrast with AspectualACME. An interesting survey on AO Analysis and Design Approaches (Chitchyan et al., 2005) was delivered by the AOSD-Europe network and cover such approaches. There are also several approaches to represent and visualize architectural aspects (Pinto and Fuentes, 2007; Krechetov et al., 2006) based on the UML (Rumbaugh et al., 2004).

3.1.4. *Open issues*

Given the challenges enumerated previously, there are some open issues in Early Aspects that deserve further discussion. The issue of proper elicitation of requirements is an open issue *per se*, and as such, it also reflects on eliciting requirements that are of crosscutting nature, which, as well pointed by Silva and Leite (2005a), are dependent on context, either of time or space. As a consequence, architectures modeled with aspect modularity in mind have to be prone to evolution, which again is an open issue in the large and also with respect to Early Aspects.

Within these two broad issues, research is necessary not only to have a better jump start, but as to maintain consistency along the alignment of requirements and architecture in the context of aspect-oriented modularization. Consistency maintenance for binding requirements and architecture is very much related to the recognition that the abstraction level of pointcuts must be lifted (Kellens et al., 2006; Stein et al., 2006). Syntax-based pointcuts are still typically used to select join points in different levels, exposing artifacts to pointcut fragility and reusability problems. New approaches should consider semantics-based solutions for composing aspects.

Another open issue is related to the provision of AO architecture modeling tools for the proposed methods and languages proposed by the Brazilian community. The lack of tools hampers the broad use and evaluation of the languages.

Last but not least, there is also a strong need for the assessment of the impact of the adoption of Early Aspects techniques at the requirements and architectural levels. To the best of our knowledge, there is no existing controlled experiment that quantifies if the adoption of Early Aspects approaches can really improve and contribute to the the quality of the final system produced.

3.2. *Modeling and model-based techniques*

Aspect-Oriented Software Development and Model Driven Development (MDD) (Schmidt, 2006) are useful and complementary techniques to achieve separation of concerns. Historically, AOSD has focused on *Aspect-Oriented Modeling* (AOM) and the management of crosscutting concerns at the same abstraction level (*horizontal separation*) whereas MDD has focused on the explicit separation of platform independent from platform specific concerns (*vertical separation*). As such, both AOSD and MDD can benefit from each other to tackle the challenges of developing large and complex software systems.

3.2.1. *Key challenges*

The main goal of research in *Aspect-Oriented Modeling* (AOM) has been to provide developers with general means to express aspects and their crosscutting relationships with other software artifacts. AOM has mostly focused on notation and techniques for specifying, representing, visualizing and communicating AO solutions at a more abstract level (i.e., programming language-independent).

There are several challenges concerning AOM (2011) and MBT in AOSD:

(i) *Developing conceptual models for AOSD*: Which are the essential characteristics of crosscutting concerns that need to be modeled? Are there core concepts and relationships that characterize "aspect-orientedness"?
(ii) *Specifying AO concepts in UML (Rumbaugh et al., 2004)*: How can aspects be specified and communicated using the UML?
(iii) *Developing tools*: Are there suitable tools for modeling and managing the relationship between business elements and crosscutting concerns? The lack of proper tools remains a hindrance to the widespread adoption of software modeling processes supporting AOSD.
(vi) *Minimizing model degradation*: How to minimize the negative impact of model evolution in AO systems ?
(v) *Promoting reuse*: How to improve the limited reuse (Gybels and Brichau, 2003) often achieved in AO systems?

3.2.2. *Addressing the challenges*

Aspect-Oriented Modeling: The AOSD-BR community has investigated different solutions for modeling aspects and their crosscutting relationships at the detailed design level. Such solutions have addressed the aforementioned challenges (i) and (ii). Furthermore, a novel approach for specifying and composing aspects at the architectural level has addressed challenges (iv) and (v).

Chavez (2004) proposed a *conceptual model* for AOSD. The main motivation of having such a conceptual framework was to support the characterization and comparison of existing AOSD approaches and the development of new methods, languages and tools based on unified terminology, concepts and properties. The proposed conceptual model was used to characterize four representative AOSD approaches (Chavez and Lucena, 2003) and to drive the design of `aSideML`, a new Aspect-Oriented Modeling language (Chavez, 2004).

The `aSideML` language (Chavez, 2004) was proposed to address challenge (ii). The `aSideML` is a modeling language for specifying and communicating AO designs. It provides notation, semantics, and rules with the main purpose of addressing the conceptual modeling and detailed design of a system in terms of aspects and crosscutting within the UML Object Model – for structural and behavioral models. The `aSideML` language has been used by different research groups from Brazil (Cole et al., 2004; Kulesza et al., 2005; Castor Filho et al., 2007) and abroad (Iborra et al., 2006).

The UML-AOF (Uetanabara et al., 2009) has also been proposed to address challenge (ii). UML-AOF is a profile that explicitly documents design characteristics of aspect-oriented frameworks (AOFs). UML-AOF provides stereotypes and tagged values to represent design and architectural information commonly found in AOFs, such as idioms (Hanenberg and Schmidmeir, 2003), patterns (Camargo and Masiero, 2008) and extension mechanisms.

Finally, Chavez et al. (2009) proposed style-based join point models and a new pointcut language that supports the selection of join points based on style-constrained architectural models. This work is part of the *style-based composition* approach, presented in Section , in which the stability (in face of change) and the reusability of the style-based composition models were main drivers (Chavez et al., 2009). Therefore, this work also addresses challenges (iv) and (v).

Model-based techniques in AOSD: In the context of MBT in AOSD, our community has investigated solutions for the aforementioned challenges (ii), (iii), (iv), and (v).

The lack of suitable tools for modeling and managing relationships between business elements and crosscutting concerns has been currently addressed by the Brazilian research community through the development of an infrastructure named CrossMDA (Alves et al., 2008a). CrossMDA encompasses a transformation process as well as a set of services and associated supporting tools. CrossMDA is based on the Model-Driven Architecture (MDA) (Mellor et al., 2004) and aims at: (i) raising the abstraction level of aspect modeling through the use of platform independent models (PIM) representing crosscutting concerns independent of the business models; (ii) promoting reuse of crosscutting concerns modeled as PIM elements; (iii) automating the process of mapping the relationship of crosscutting concerns and business models through automatic MDA transformations. CrossMDA encompasses a process that provides a clear separation between the modeling elements (aspects and business elements) and the different abstraction levels (MDA models) contributes to promote artifact reuse. Model based techniques are also being applied to handle problems that arise in the presence of software evolution. As in other software engineering approaches, AO systems are susceptible to software degradation due to the coupling between aspects and base code. One major source of degradation for AO systems is that the join points intercepted by the aspects may unexpectedly change as the system evolves. This may affect the pointcuts referring to such evolved points, which become fragile. The main reason for pointcut fragility is that, in most AOSD approaches, pointcut expressions (PCEs) are specified at a low abstraction level, referring to syntax-based join points specific to the system being developed (Koppen and Stoerzer, 2004; Kellens et al., 2006).

To address the pointcut fragility problem and decrease the coupling between aspects and base code, new research exploits the fact that implicit rules for aspect composition must be explicitly represented, for instance, by using model-based pointcuts (Kellens et al., 2006). The research community in Brazil has developed different solutions to tackle the pointcut fragility problem using a model-based approach. The model-based approach builds on the definition of a conceptual model that describes structural and behavioral concepts shared by an application domain and classifies elements of the base model according to these concepts. Such a model is used as an intermediary model that decouples the aspects from the base model. The main distinctive feature of the solutions proposed by the Brazilian AOSD community is their architecture-driven nature. These solutions leverage the ideas of the model-based approach, aligning it to the best practices of software development.

The first solution is an extension of the CrossMDA framework (Alves et al., 2008a), named CrossMDA2 (Fernandes et al., 2009), built as a new instantiation of the Kellens approach to model-driven pointcuts (Kellens et al., 2006). By integrating MDA with the conceptual models approach, the solution seeks to provide a development process that includes the specification and handling of the conceptual model in the entire software development life-cycle. The second solution (Pinto et al., 2009) also relies on the use of model-based pointcuts (Kellens et al., 2006) and in the definition of a conceptual model at the architectural level. However, in this solution three different layers in the definition of the conceptual model are defined: the system layer, the domain-specific layer and the application-specific layer. An MDD process drives the definition of conceptual and aspect models, and their instantiation and composition to generate the architectural base model. The solution is implemented using AO-ADL (Pinto and Fuentes, 2007), an aspect-oriented architecture description language. Since in both solutions the proposed conceptual models build on architectural patterns, which encapsulate total or partial architectural solutions, such approaches facilitate the reuse of concepts in different applications, minimizing the effort of specifying a conceptual model for each application compared to other model-based proposals.

3.2.3. Comparison with research conducted by the international community

Wimmer et al. (2011) proposes a conceptual reference model and uses it to evaluate several UML-based aspect-oriented design modeling approaches. Besides this one, a number of conceptual or reference models for AOSD concepts have been proposed in the AOSD literature, such as the AOSD-Europe ontology (Berg et al., 2005), Masuhara's conceptual framework (Masuhara and Kiczales, 2003), and Everman's (Evermann, 2007) reference model and profile for AspectJ programs.

Wimmer et al. (2011) work also provides a comprehensive and updated survey on UML-based aspect-oriented design modeling approaches. Among them are two well-known approaches for AOM and AOD (Chitchyan et al., 2005): Theme/UML (Clarke and Walker, 2005), a popular heavyweight approach used for aspect-oriented design (AOD), and Join Point Designation Diagrams (JPDDs) (Stein et al., 2006), a graphical visualization that expresses join point selections in UML in which each visualization can be examined with respect to the selection's underlying conceptual model (control flow-oriented, data flow-oriented, or state-oriented conceptual model).

The lack of suitable tools for modeling and managing relationships between business elements and crosscutting concerns has been currently addressed by international research that combines the concepts of AOM with MDA. There are several proposals for integrating crosscutting concerns in models by using MDA (Reina and Torres, 2005; Simmonds et al., 2005; Solberg, 2005; Wampler, 2003). In these pieces of research, aspects and their relationships with business elements are represented as first-class modeling elements.

There are mainly two different approaches that tackle the degradation problem of AO systems that occurs when they evolve: design-based pointcuts (Cazzola et al., 2006) and model-based pointcuts (Kellens et al., 2006). Design-based approaches rely on the premise that the problem to be solved is to provide pointcut expressions that support selecting join points based on their semantics (instead of their syntax), which must be explicitly represented by design information. Some researches represent design information at the modeling level using UML (Cazzola et al., 2006), while others do so at the coding level (Sullivan et al., 2005). Although coupling is minimized by defining explicit design rules, these rules are represented in a very low abstraction level (code), being more application-specific and dependent on the implementation and AO languages. Moreover, their management is hard in the presence of software evolution.

Kellens et al. (2006) proposed model-based pointcuts to address the fragile pointcut problem by replacing the intimate dependency of pointcut definitions on the base program by a more stable dependency on a conceptual model of the program. The authors argue that the model-based pointcut definitions are less likely to break upon evolution since they are no longer defined directly in terms of the actual base code structure. Therefore, the fragile pointcut problem is transferred to a more conceptual (and probably more stable) level and turned into a synchronization problem between the conceptual model and the base code whenever the latter code evolves.

3.2.4. *Open issues*

There are several open issues that deserve further investigation in the context of modeling and model-based techniques for AOSD. One such issue regards experimenting and assessing the existent approaches in order to answer questions such as: Are AOM approaches mature enough to be applied in real world scenarios? What kind of assessment is still necessary? The area still lacks experiments that evaluate practical aspects, such as the overhead introduced in the software development process by the proposed approaches.

For AOM, there is a need for a standard notation and approaches to support aspect composition and interference, and the management of relationships between aspects and business elements. AOSD-BR community initiatives such as CrossMDA focus on addressing this issue. However, tools for modeling and managing the relationships between business elements and crosscutting concerns remains as an obstacle to the wide-spread adoption of software modeling processes supporting AOSD.

Research in model-based techniques in AOSD can be further explored in different directions. One possible direction is to investigate the types of join points that can be suitably captured by architecture-oriented approaches and those that cannot be captured (or even do not make sense to be).

3.3. *Refactoring*

Refactoring (Opdyke, 1992) is the process of improving the design of software artifacts without changing their observable behavior. AO refactoring, in particular, combines refactoring and aspect-oriented programming techniques with the main aim of modularizing or improving the design of CCCs. These techniques can be used, for example, to restructure and improve the internal quality of OO systems by implementing CCCs with aspects. In addition, AO refactoring also refers to quality improvement of AO implementations, including behavior preserving changes to existing aspects and the study of extra conditions needed to assure that typical OO refactorings are correctly applied in the presence of aspects.

3.3.1. *Key challenges*

In this area, the key challenges are associated with each of the typical refactoring activities that should be supported by refactoring processes (Mens and Tourwe, 2004):

(i) *Identification of refactoring opportunities*: Which techniques can be used to identify opportunities to apply AO refactorings? How to structure and organize a catalog of AO refactorings?
(ii) *Assessment of refactoring effects on software quality*: What is the impact of AO refactorings on the internal quality of software systems? How can AO refactorings affect the stability and robustness of existing evolving software systems?
(iii) *Guaranteeing that behavior is preserved after refactoring*: How to guarantee behavior preservation during the application of AO refactorings? How can formal approaches help with the verification of behavior preservation in the context of aspect-oriented languages and other properties?

These are similar to OO refactoring challenges, but in addition have to consider a number of effects caused by the support of the more powerful language mechanisms provided by AO languages.

3.3.2. *Addressing the challenges*

With the aim of addressing these challenges, the Brazilian community has worked on a number of initiatives, as detailed next, separately for each challenge.

Identification of refactoring opportunities: Regarding the identification of refactoring opportunities, the Brazilian community has made several contributions, including the definition of catalogs of code smells (Piveta et al., 2005; Bertran et al., 2011), algorithms for code smell detection and suggestions of refactorings for their removal (Piveta et al., 2006), design guidelines to reduce code smells in AO software (Piveta et al., 2007), and a process for refactoring (including refactoring sequences) (Piveta, 2009; Piveta et al., 2008). In the same line of research, Bertran et al. (2011) present an analysis of code smells recurrently observed in a set of evolving aspect-oriented systems, analysing instances of code smells previously reported in the literature as well as describing new ones. Garcia et al. (2004) proposed several refactorings for the manipulation of CCCs, including Rename Pointcut, Collapse Aspect Hierarchy, and Collapse Pointcut Definition.

Refactoring is also used to provide mechanisms for aspect mining, in which code fragments related to CCCs are located and extracted to aspects. However, the original (base) program often has to be modified to fit these aspects properly. In this context, the Brazilian community has proposed a catalog of OO transformations (Valente et al., 2009) to associate crosscutting code with points of the base program that can be captured using the pointcut model of current AO languages. This catalog was evaluated on a case study with medium-sized Java systems that have been aspectized using AspectJ.

Recent studies have pointed out that not all kinds of crosscutting concerns (CCCs) are harmful to the system quality attributes, such as robustness (Ferrari et al., 2010) and stability (Figueiredo et al., 2008a). Therefore, only harmful CCCs should be factored out to aspects. Figueiredo et al. Figueiredo et al. (2008a) showed, for instance, that aspectizing some optional or alternative features of software product lines can produce mandatory features whose implementing modules are less stable. Brazilian researchers have documented (Figueiredo et al., 2009d) and proposed means to modularize several patterns of CCCs recurrently observed in software systems (Silva et al., 2009). One of the key findings of Figueiredo et al. (2009d) is that crosscutting concerns following some specific patterns may not be easily modularized, even with the use of AOP mechanisms. The patterns of harmful CCCs can be detected by means of a variety of resources, including visualization environments (Carneiro et al., 2010), metrics and heuristic strategies (Figueiredo et al., 2009a,b). Some of the heuristics defined by Figueiredo et al. (2009b) are explained in Section 3.6.2.

Assessment of the effect of refactoring on software quality: With respect to the challenge of assessing the effects of refactoring on quality, AOSD-BR researchers have been developing tools and concepts to improve the quality of AO software. One of these tools is ConcernMetrics (Valente et al., 2010), an Eclipse plugin that computes metrics commonly employed to evaluate the benefits of AOP, without requiring explicit extraction of CCCs to aspects. This tool was successfully used to evaluate the benefits of using aspects in three small-to-medium sized Java systems (Valente et al., 2010).

Valente et al. (2010) have also evaluated the implementation of logging using aspects, as supported by AspectJ. They showed that logging implementations require developers to call methods from the logging API in several parts of the system, which justifies the recurrent use of logging as an example of CCCs. However, such calls usually have different strings as arguments, i.e., different statements (one for each string argument) are needed to provide logging behavior throughout the system. As a consequence, AspectJ developers cannot extract and merge the logging calls in a single or in a small number of advices. In other words, the extracted advices have a reduced degree of quantification, i.e., they affect a small number of locations of the base program (in most cases, just one location). In view of such circumstances, Valente et al. argued that the refactored (aspectized) implementation of logging does not present clear advantages, in terms of comprehensibility, changeability, and independent development, when compared with the original code.

A method for ranking refactorings according to quality attributes was also proposed (Piveta et al., 2008). The relative importance of each quality attribute with respect to the others and the relative importance of the refactorings over quality attributes is quantitatively expressed using the AHP multi-criteria decision method (Saaty, 1990, 2003). Using this quantified information, a ranking of refactorings in terms of their contribution to ranked quality attributes can be automatically computed. The use of such ranking can optimize the search for refactoring opportunities, enabling the developer to focus on refactoring patterns that contribute most to improve the required quality attributes of a piece of software.

Guaranteeing that behavior is preserved after refactoring: The AOSD-BR community has also played a pioneering role in research focusing on guaranteeing that AO refactoring preserves behavior. The first work in this area uses AO programming laws for deriving refactorings for AspectJ (Cole and Borba, 2005; Cole et al., 2005), aiming at verifying if the transformations they define preserve behavior. These laws have been used to derive a catalog of AspectJ refactorings, which can be employed to restructure applications and to extract variations in software product lines. In both cases, the aspects were used as a modularization mechanism, either for features (Alves et al., 2007) or for common CCCs such as distribution and persistence. Most refactorings are implemented in an industrial-strength refactoring tool (Calheiros et al., 2007) and have been applied in the mobile game domain (Alves et al., 2008).

The AOSD-BR community has also extended the traditional notion of program refactoring to software product lines (SPLs), in which feature models (FMs) are refactored, in addition to source code elements (Alves et al., 2006). A FM transformation is a refactoring when the resulting FM improves (maintains or increases) the set of all possible configurations (products) of the initial FM. So, a SPL refactoring not only improves code structure, but also the quality of the FM by maintaining or increasing the SPL configurability in extractive or reactive scenarios (Krueger, 2001). We note that, in the SPL context, in contrast to the usual definition of program refactoring (whose scope involves only *one* program), semantics preservation refers to the *set* of SPL products and means that *previously existing* products are not affected after the transformation. Accordingly, semantics is preserved after FM refactoring because previously existing configurations and corresponding products still exist in the refactored FM and these have the same behavior. The possibly new configurations and corresponding products do not interfere with the semantics of the existing ones and simply indicate the enhanced configurability of the FM, which is regarded as a desirable property improved during refactoring motivated by the recurring occurrence of the extractive and reactive SPL adoption strategies.

3.3.3. *Comparison with research conducted by the international community*

The AOSD international community has also proposed catalogs of code smells for AO software. Monteiro and Fernandes (2006) describe three code smells for AO code, and Srivisut and Muenchaisri (2007) describe a set of additional code smells that can be found in AO software, along with algorithms to automate their detection and suggestions of refactorings for their removal. Additionally, several refactorings have been proposed to enable the manipulation of code elements in AO software (Hanenberg et al., 2003; Iwamoto and Zhao, 2003; Monteiro and Fernandes, 2004, 2005).

Iwamoto and Zhao (2003) discuss commonly used refactorings for OO software and claim that few can be used in AO software without adaptations. They propose refactorings for extracting and creating advice and pointcut. Hanenberg et al. (2003) propose a set of conflict resolution strategies in the context of refactoring AO code written in AspectJ, and a set of refactorings for aspect-extraction and reestructuring of AO code. Monteiro and Fernandes (2004, 2005, 2006) provide a comprehensive catalog of refactorings for AO software.

Instead of working directly on the behavior preservation property of AO refactorings, the international community focused so far on the underlying formal semantics of AO languages. Besides the operational semantics for method call interception (Lämmel, 2002) used by the Brazilian community to prove soundness of AO refactorings, a number of other approaches, based on different styles, were proposed for formally reasoning about aspect-oriented programs in different languages (Douence et al., 2001; Andrews, 2001; Wand et al., 2004).

3.3.4. *Open issues*

Future AO refactoring research work should focus on the analysis of behavior preservation when adopting other advanced modularization techniques, and making sure that refactoring tools correctly implement refactorings. In the context of software product lines, for example, a formal notion of refactoring (Borba et al., 2012) can help the development of analysis and testing tools that contribute to the verification of specific properties when applying successive refactorings. That kind of strategy can also be useful in the context of virtual separation of concerns approaches (Kästner et al., 2008). In addition, refactoring techniques should also be applied and investigated in early phases of software development in conjunction with visualization mechanisms or modeling techniques that provide support for the separation of concerns principle. The area also needs more empirical evidence of the benefits and drawbacks of applying AO refactoring in more realistic contexts.

3.4. *Testing*

The understanding and subsequent testing of AO programs is one of the main ways to make the approach less costly and more feasible in practice (Alexander, 2003). With this in mind, the AOSD-BR research community has contributed to establish testing approaches that address particularities of AO software.

3.4.1. *Key challenges*

We can enumerate five main challenges posed by AOP with respect to the testing activity. Each challenge comes together with associated questions. As described in the sequence, Brazilian researchers have been addressing these challenges with focus on some particular testing techniques. Note that, in a broader view, they represent typical challenges tackled by software testing researchers, however considering the specificities of AOP.

(i) *Identifying new potential problems*: What are the new sources of faults? Which AO-specific concepts and programming mechanisms impact the fault-proneness of the produced systems?
(ii) *Defining proper underlying models*: Which software artifacts can be used to build models from which test requirements are derived?
(iii) *Customizing existing test selection criteria and/or defining new ones*: Are existing test selection criteria able to reveal

AOP-specific types of faults? Do they require any adaptation? Should new criteria be defined?

(vi) *Providing adequate tool support*: Is it possible to use existing testing tools to support the adopted technique/criteria? What else, in terms of automated support, is necessary?

(v) *Experimenting and assessing the approaches*: How mature are AOP-related testing approaches? Are they ready to be applied in real world scenarios? What kind of assessment is still necessary?

3.4.2. Addressing the challenges

To address the aforementioned challenges, the Brazilian community has been investigating the three most widespread testing techniques: functional-based, structural-based and fault-based testing. Details are provided next.

Functional-based testing: Functional-based testing, particularly state-based testing, which derives test cases by modeling a class/aspect as a state machine, has been addressed by Silveira et al. (2005). The emphasis of this work is on the importance of developing a testing model for AO programs which can reveal a new potential fault caused by aspect composition in both static and dynamic weaving. METEORA (Silveira, 2007) is an extension of the *Aspectual State Model* (ASM) (Xu et al., 2005) and Flattened Regular Expression (FREE) (Binder, 1999) models. It is a new state-based testing method for AO software which includes a model to represent the dynamic behavior of aspect interactions, an extended strategy to derive testing sequences, and a testing tool prototype to support the method. This model introduces several extensions to the previous ones, in order to enable the representation of added, removed and/or changed states by aspects or aspects compositions woven into classes.

Structural-based testing: Zhao (2002) published the seminal piece of work related to structural testing of AO programs. Since then, relevant AOSD-BR contributions have emerged, resulting in a series of papers published in scientific events (Lemos et al., 2004; Lemos and Masiero, 2008) and in high quality journals (Lemos et al., 2007, 2009; Lemos and Masiero, 2011). Contributions address structural-based testing at the unit level (Lemos et al., 2004, 2007) and the integration level (Lemos and Masiero, 2008, 2011; Lemos et al., 2009). Control-flow and data-flow models have been formally defined, from which a range of test selection criteria have been devised. The control-flow criteria extended the all-nodes and all-edges criteria to incorporate AO-specific properties like join points and pointcuts. Similarly, the widely studied data-flow criteria, such as all-defs and all-uses, were adapted to take into consideration data that flows to and from aspectual elements like pieces of advice and introductions. The applications of all these criteria is supported by a tool named JaBUTi/AJ (Lemos et al., 2007, 2009; Lemos and Masiero, 2011), which has been continuously evolved in the last few years.

More recently, Lemos and Masiero (2011) have proposed stronger control and data-flow criteria with an underlying model. The model represents each advice integrated at each join point it may affect, and the criteria require test cases to cover all nodes (or statements), edges (or decisions), and def-use pairs of the advice at each integration point. Besides this, Cafeo and Masiero (2011) proposed an extension to the AO-based criteria that considers deeper integration levels, since the previous approach only tackled 1-level integrations.

Fault-based testing: Fault-based testing (Morell, 1990) requires well-characterized fault types – usually organized as *fault taxonomies* – to be applied effectively. Such a taxonomy, also called *fault model*, underlies the definition of fault-based testing approaches. In this context, Alexander et al. (2004) had the first initiative to define a candidate fault model for AO software. A few years later, Brazilian researchers performed a systematic survey in order to devise a comprehensive taxonomy (Ferrari et al., 2008). It includes four categories of faults, each related to the main programming mechanisms present in AO systems: pointcuts, introductions, advices and the base code itself. Their subsequent studies investigated the fault-proneness of evolving AO programs and revealed that the taxonomy has the ability to classify the observed variety of AOP-specific faults. These studies addressed, respectively, coarse-grained (Ferrari et al., 2010) and fine-grained fault classifications (Ferrari et al., 2010).

The taxonomy proposed by Brazilian researchers guided the definition of a fault-based testing approach for AspectJ-like programs (Ferrari et al., 2008). The approach explores the Mutant Analysis criterion (DeMillo et al., 1978) and encompasses a set of mutation operators that model several instances of fault types. Automation is achieved with the Proteum/AJ tool (Ferrari et al., 2010). It supports the full testing process, which starts with the derivation of test requirements (i.e., the mutants), the execution of both the application under test and the respective mutants, and the analysis of test results.

3.4.3. Comparison with research conducted by the international community

With respect to structural testing, the body of work produced by the Brazilian community can be considered relevant in comparison to the international context. For instance, to the best of our knowledge, the seminal work published by Zhao (2002, 2003) did not present implementations of the testing approaches. On the other hand, all model and criteria proposed by the Brazilian researchers were implemented in consecutive extensions of the JaBUTi tool. Moreover, in quantitative terms, the national group has shown to be highly active in comparison to other groups that have done research on the same topic. Evidence on this can be found in an extensive survey of testing approaches for AO software documented elsewhere (Ferrari, 2010).

The contributions of the AOSD-BR community with respect to the fault characterization for AO programs has inspired some initiatives by the international community. The fault taxonomy for AO programs (Ferrari et al., 2008, 2010) was applied to classify faults observed in a range of AO applications in a series of metrics-related studies (Burrows et al., 2010a,b, 2011). Furthermore, the mutation operators defined by Brazilian researchers were applied by Delamare et al. (2009) in a preliminary assessment of a novel test-driven approach for AO software. Delamare et al. also developed a tool named AjMutator (Delamare et al., 2009b) whose features overlap with Proteum/AJ's.

Regarding state-based testing, the Brazilian community has a major focus on aspect composition, rather than aspect-class composition. Abstractly, the idea of executing together software elements created separately is called *composition*. The term *aspect composition* refers to relationships among aspects woven into the target system. Thus, composition has a different meaning from the weaving and/or combination processes. The latter occur when one or more aspects are woven into an application without necessarily interacting with each other (Silveira et al., 2005).

3.4.4. Open issues

Experimentation and other types of evaluation comprise an essential part of research. With this in mind, a major limitation to be addressed by the community is the robust evaluation of the varied testing approaches proposed in the last decade. Another prominent research branch comprises hybrid approaches that combine different testing techniques (e.g., structural and mutation testing, or static and dynamic evaluation). Preliminary efforts have been reported (Lemos et al., 2006; Coelho et al., 2009) and shall be further revisited.

There are other research topics concerning software testing that have not yet been fully investigated in the AO context. For instance, the testing of AO programs considering exception-handling constructs is a topic to be further studied. To date, there is little research that deals with this problem, but still in a preliminary fashion (Coelho et al., 2009). Another topic yet to be adequately explored is the testing of AO programs considering the interaction of multiple aspects at coincident points in the system. Aspect interaction and composition is an important topic of AOSD that has not yet been considered in this context.

Other open issues, regarding state-based testing, include: (i) investigating the adaptability of current approaches to the exception model and to the response matrix, proposed by Binder (1999); (ii) carrying out experiments using the concept of sneak paths; (iii) extending current methods to deal with dynamic aspects; and (iv) investigating the possibility of developing specific mutant operators to generate aspects with composition faults.

3.5. Exception handling

Exception handling (Goodenough, 1975) (EH) mechanisms are one of the most frequently used techniques for modularizing the error recovery concern. In modern languages such as Java and AspectJ, the actions taken to recover from errors are encapsulated into handlers (`try-catch` blocks), and exceptions are represented as objects that are raised when an exceptional condition is detected. Raising an exception interrupts the normal control flow of the program; this is followed by a search for an appropriate exception handler that deals with the raised exception. Although one of the main goals of the EH mechanisms is to improve software modularity by promoting explicit separation between code representing normal behavior and EH code, the latter is a promising candidate for the use of (AOSD) techniques since, in most programming languages, it is a scattered and tangled concern (Kiczales et al., 1997).

3.5.1. Key challenges

The main challenge in this area is to understand how aspect-oriented techniques, both existing and new ones, can improve the quality of EH code. In addition, aspects themselves create new problems in terms of error recovery and circumventing these problems poses additional challenges.

(i) *Improving the modularity of error handling code*: Do aspects improve quality attributes such as cohesion, conciseness, and coupling? Are error handling aspects reusable? If there are benefits in using aspects for EH, how can we incorporate them into existing systems?
(ii) *Understanding the impact of aspects on program reliability*: Do error handling aspects introduce new bugs in a system? How can these bugs be characterized? What kind of tool support can tackle existing problems?
(iii) *New language constructs for error handling*: What are the shortcomings of AspectJ for modularizing EH code? Which one would be the best approach for EH: domain-specific or general-purpose aspects?

3.5.2. Addressing the challenges

To address the challenges presented in the previous section, we can divide research conducted by the Brazilian community in two groups: (i) empirical studies; and (ii) new languages, techniques, and tools. We explain both in the remainder of this section.

Empirical studies: Castor Filho et al. (2006) investigated the use of aspects to modularize EH code by comparing object-oriented and aspect-oriented versions of four different, non-trivial applications. The two versions of each system were compared in terms of coupling, cohesion, conciseness, and separation of concerns using a number of source code metrics and a qualitative evaluation. This study revealed that modularizing EH code into aspects, also called Error Handling Aspects (Castor Filho et al., 2006) (EH Aspects), and reusing these aspects is not straightforward. Instead, it depends on a set of factors, such as the type of exceptions being handled, what the handler does, and the amount of contextual information needed. In addition, it makes the systems bigger, in terms of number of lines of code. Later on, Castor Filho et al. (2007) elaborated a catalog of best and worst practices related to the aspectization of EH. This catalog aims to guide developers in deciding when it is beneficial to extract error handling to aspects and when it might have a negative impact on the quality attributes of the resulting system. Taveira et al. (2009) continued the investigation and found out that a considerable amount of error handling code can be reused within a single application by using aspects. Nonetheless, the programming overhead associated with using aspects to that end somewhat overshadows the reuse.

Coelho et al. (2008) performed an empirical study considering the error-proneness of AspectJ constructs for handling exceptions. The most important finding of the study was that EH Aspects, even when used with great care, can introduce a number of bugs that materialize as uncaught exceptions and unintended handler actions. The conducted study could observe that, in a multitude of scenarios where the EH Aspects were responsible for catching exceptions, the exceptions remained uncaught. As a consequence, they transparently propagate back to the program entry point, causing the Java virtual machine to terminate. Furthermore, aspects also cause exceptions to be mistakenly caught by an existing handler on the base code (an EH bug very difficult to detect known as Unintended Handler Action (Coelho et al., 2008)). One of the main reasons for these phenomena was the need to soften exceptions in AspectJ, in order to bypass the static checks for exception interfaces that the Java compiler performs. This led to the creation of a catalog of bug patterns and an approach for preventing and automatically detecting such bug patterns (Coelho et al., 2008).

New languages, techniques, and tools: Most of the findings of the aforementioned studies indicate that the limitations of AspectJ as a means to modularize EH code are not inherent to AOP. Taking that assumption as a starting point, Cacho et al. (2008) devised a new EH model named EFlow that uses join points and advice as means to generalize the concepts of EH context and exception handler. This model was implemented as a domain-specific extension to AspectJ named EJFlow and solves most of the problems pointed out by other studies. Later, Cacho et al. (2009) performed an exploratory study to evaluate this new model and observed that it was useful to foster the development of readable, weakly coupled, and reliable software systems.

Furthermore, Brazilian researchers have recently devised testing and static analysis approaches targeting the EH code of AO systems (Bernardo et al., 2011; Coelho et al., 2011). Bernardo et al. (2011) proposed an extension of the JUnit framework to check whether the exception flows that take place in OO and AO systems are the intended ones. The proposed tool relies on aspects to monitor the exception flows of a system. Coelho et al. (2011) presented a static analysis-based approach which allows AO developers to describe the EH behavior of a system and automatically check it. Both approaches target Java- and AspectJ-based systems.

3.5.3. Comparison with research conducted by the international community

Lippert and Lopes (2000) pioneered the investigation of the use of aspects to modularize EH code. In their work, they achieved a considerable reduction in the amount of EH code in the AO version of a reusable object-oriented framework. Moreover, they discovered that error handling aspects can be “plugged”, thus improving reusability. Since the seminal study, a substantial body of work has studied the effects of aspects on EH, from a number of different perspectives. AOSD-BR researchers have conducted a considerable part of this research.

Some researchers in the international community have addressed the problem of performing the actual extraction of an exception handler into an aspect. Most of this research is related to refactoring. Laddad presents the "Extract Exception Handling" refactoring. The refactoring centers around the effects of using it to extract trivial error handling code, but does not explain when it is useful (or possible) to apply it in practice. This refactoring was latter implemented in a tool developed by Binkley et al. (2006). While refactorings targeting error handling code concentrate on the mechanics of moving a `try-catch` block to an aspect, the research conducted by the Brazilian community identifies situations where this is beneficial and situations where it is not.

In terms of new language constructs, Hoffman and Eugster (2008) propose the concept of explicit join point (EJP) as a means to reduce the amount of obliviousness of aspect-oriented programs. The authors believe that this approach results in improved maintenance and understandability, at the costs of losing some of the textual separation yielded by AOP. Even though EJPs are a general-purpose concept that does not necessarily pertain to exceptions, their work took as a starting point our studies (Castor Filho et al., 2006, 2007) on EH and aspects. Moreover, the evaluation of their approach (Hoffman and Eugster, 2008) focuses mainly on EH. Therefore, it presents a counterpoint to the EJFlow approach, since it proposes a general purpose mechanism to solve some of the problems that AspectJ creates when employed to modularize error handling.

More recently, Figueroa and Tanter (2011) proposed the explicit use of Execution Levels (Tanter, 2010) as a means to avoid many of the bugs that error handling aspects introduce. These bugs were uncovered and cataloged by research conducted by the Brazilian research community (Castor Filho et al., 2006; Coelho et al., 2008).

3.5.4. Open issues

From the seminal work in 2000 to current research, we have achieved a good understanding of the limitations of current EH mechanisms when applied in the AOP context as well as the benefits and drawbacks of AOP, in particular when used to modularize the EH code. Hence, we can point out some issues that remain open and can be addressed by future work in the area of advanced mechanisms for software modularization.

First, almost all the previous studies on the use of aspects to modularize EH has targeted the AspectJ language. This is not surprising, since it was one of the first AOP languages and also the first one that had an acceptably mature toolset. Therefore, even though a number of new languages and techniques for aspect-oriented programming have been proposed throughout the years, we do not know whether they are useful to modularize error handling code. Therefore, empirical studies targeting these approaches are necessary.

In software product lines, separating normal and exceptional behavior can enable developers to design software variability related to different EH strategies. However, existing techniques that are employed to design and implement variability in software product lines do not consider the presence of exceptions and exception handlers. This "exceptional variability" depends on the resources available in each product, and is related to both the selection of proper handlers, and the existence of different exception control flows. Moreover, it involves guaranteeing that, despite the existence of variation points that might or might not be present in the final product, exceptions are handled where they are intended to be and no exception goes unhandled.

The Java language requires developers to either handle all the checked exceptions that a method encounters or indicate in the method's `throws` clause (the exception interface of the method) the ones that it does not. Some developers (Venners and Eckel, 2003) argue that `throws` clauses hinder maintainability and even reliability and, therefore, should be avoided as much as possible. Although there are proposals to solve some of the problems of Java's exception interfaces (van Dooren and Steegmans, 2005), they still require exception interfaces to be scattered throughout the methods of a system. AOP-like approaches for specifying exception interfaces seem to be a promising way of overcoming such challenges (Cacho et al., 2008, 2009). The aforementioned EJFlow model is an important step in this direction (Cacho et al., 2008), although there is much room for improvement. First, by decoupling the solution for exception interfaces from AspectJ. Second, by specifying simple and precise semantics.

Finally, recent advances in the area of mobile computing have enabled the development of a wide variety of event-driven applications, such as context-aware and mobile applications that are able to monitor and exploit dynamically changing contextual information from users and surrounding environments. The treatment of exceptional conditions in event-driven applications seems to indicate a number of open issues. Most of those issues are related to the fact that the design of EH mechanisms needs to be tightly integrated with the underlying module system. Therefore as the advent of specific middleware (Cugola and de Cote, 2005) and programming languages (Kamina et al., 2011; Costanza and Hirschfeld, 2005; Hirschfeld et al., 2011) is impacting on the way system modules are structured and interact with each other, suitable EH abstractions and mechanisms need to be investigated.

3.6. Metrics

As a new software development technique emerges, it is essential that empirical studies are carried out to investigate whether and in which cases the advocated benefits are real. Since the beginning of AOSD, it has been difficult to determine what is a good design or implementation based on AO mechanisms. It is not trivial to understand when to use aspects as architectural and design solutions, unless for implementing obvious crosscutting concerns, such as logging. The software engineering community agrees that empirical studies are necessary to evaluate the usefulness of AOSD and associated design practices. Software metrics provide a powerful means to make empirical studies more systematic and less subjective. Before the advent of AOSD, a number of software metrics were available in the literature, e.g., number of lines of code (Fenton and Pfleeger, 1998), McCabe's complexity metrics (Fenton and Pfleeger, 1998) and Chidamber and Kemerer object-oriented metrics (Chidamber and Kemerer, 1994). However, most of them required adaptations to be employed in the context of aspect-oriented software.

3.6.1. Key challenges

The AOSD paradigm proposes new abstractions and new composition mechanisms. As a consequence, most of the existing metrics were not suitable to be applied on AO software straight away. It was necessary to devise new metrics for evaluating AO software. Since AOSD promised improved separation of concerns, metrics for quantifying the modularization of concerns were also required, in order to verify this claim. In summary, the main challenges were threefold:

(i) *Developing modularity-related metrics suitable to AO software*: What kinds of metrics should be developed to deal with AO composition mechanisms? Can existing object-oriented metrics be adapted to be applied on AO software?
(ii) *Developing metrics for quantifying attributes related to separation of concerns*: How can we measure concern scattering

and tangling? Are there other attributes related to separation of concerns that should be measured? What kinds of design anomalies can be identified with this kind of metrics?

(iii) *Conducting empirical studies to assess AO designs aiming at better understanding the benefits and drawbacks of AOSD*: How can we use metrics to assess the impact of AOSD on the quality of software design? When does AOSD improve the quality of software design? When does AOSD hinder the quality of software design?

3.6.2. Addressing the challenges

To address the aforementioned challenges, the AOSD-BR community has been developing significant work on AO metrics and quantitative assessment of AO software.

Metrics for AO software: Regarding the development of AO metrics, Sant'Anna et al. developed one of the first metrics suites for quantifying modularity-related attributes in AO software, published in an SBES paper (Sant'anna et al., 2003). The suite included coupling, cohesion, and size metrics adapted from existing OO metrics to deal with AO abstractions and mechanisms. For instance, they adapted Chidamber and Kemerer's Coupling Between Objects (CBO) metric (Chidamber and Kemerer, 1994) in order to additionally take into account AO composition mechanisms that cause dependency between aspects and classes. For instance, the new coupling metric, named Coupling Between Components (CBC), considers that an aspect A is coupled to a class C when A defines a pointcut that refers to C.

Metrics for quantifying separation of concerns: The suite proposed by Sant'anna et al. (2003) also encompassed innovative concern-driven metrics, aimed at quantifying different facets of separation of concerns. Concern-driven metrics promote the notion of concern as a measurement abstraction. This kind of metric allows the identification of specific design flaws or design degeneration caused by the poor modularization of concerns. Most of the existing concern-driven metrics focus on quantifying the degree of concern scattering and tangling. Concern-driven metrics are based on a concern-to-design (or concern-to-code) mapping. The mapping consists of assigning a concern to the corresponding design elements that realize it. Therefore, before computing concern-driven metrics, it is necessary to identify and document the design elements or pieces of source code responsible for implementing each concern in the system. For instance, one of the most popular concern-driven metrics is Concern Diffusion over Components (CDC) (Sant'anna et al., 2003). CDC counts the number of components (classes and aspects) whose purpose is to totally or partially contribute to the implementation of a particular concern. In other words, CDC, for a given concern, counts the number of components in the system that contain attributes, methods, or lines of code to which the concern is mapped. It enables the designer to assess the scattering degree of a concern.

Subsequent work undertaken by the Brazilian community focused on further investigating concern-driven measurement. Sant'Anna et al. (2007) and Sant'Anna (2008) extended the metrics suite defined in Sant'anna et al. (2003) with new concern-driven metrics. Besides concern scattering and tangling, these new metrics also quantify concern interaction, concern-based cohesion, and concern-sensitive coupling. This work also adapted concern-driven metrics to be applied on architectural design. The architectural metrics were used in some empirical studies (Sant'Anna et al., 2007, 2008; Greenwood et al., 2007; Sant'Anna, 2008). The concern-driven metrics were later formalized by means of a conceptual framework (Figueiredo et al., 2008b) that supports not only the formalization but also the instantiation and comparison of concern measures. It subsumes the definition of terminology and criteria in order to foster the definition of meaningful and well-founded concern measures.

Figueiredo et al. (2009b) defined heuristic rules based on concern-driven metrics aiming at exploiting concerns as explicit abstractions in the holistic assessment process. The heuristic rules aim to support concern-driven analysis of software design. First, a basic set of heuristics identifies and classifies crosscutting concerns according to their primitive properties, such as their degrees of tangling and scattering. Then, a second set of heuristics identifies specific crosscutting patterns according to more sophisticated properties of crosscutting concerns, such as coupling and inheritance relationships. Finally, a third set of heuristics aims to detect classical bad smells (Fowler et al., 1999) that are often sensitive to the way concerns are realized in the source code, such as feature envies and god classes.

Empirical studies: Several empirical studies used the suite of AO metrics and heuristics (Sant'anna et al., 2003; Figueiredo et al., 2009b) to assess AO systems of different domains and nature. A first empirical study relied on the suite of metrics to compare Java and AspectJ implementations of a multi-agent system (Sant'anna et al., 2003; Garcia, 2004). Empirical studies (Sant'Anna et al., 2004; Garcia et al., 2005, 2006b) compared the modularity of Java and AspectJ solutions to implement the 23 design patterns proposed by the Gang of Four (Gamma et al., 1995). Another study systematically investigated how AOP scaled up to deal with modularization of design patterns in the presence of pattern interactions (Cacho et al., 2006). The authors made both quantitative and qualitative assessments of 62 pair-wise compositions of patterns taken from 3 medium-sized systems implemented in the Java and AspectJ programming languages. Cacho et al. (2006) assessed the use of AOP for improving the modularity of a reflective middleware platform. Kulesza et al. (2006a) conducted an empirical study in which they quantified the effects of AOP in the maintainability of a web-based information system. Figueiredo et al. (2008a) undertook an empirical study for evaluating whether AOP promotes better modularity and changeability of product lines than conventional variability mechanisms, such as conditional compilation. Silva et al. (2009b) performed an empirical study to assess whether AO solutions for agent-oriented design patterns improve the separation of pattern-related concerns. Piveta et al. (2012) provided rigorous definitions, usage guidelines, analytical evaluation, and empirical data from ten open source projects, determining the value of six metrics for aspect-oriented software. They also discussed how each of these selected metrics can be used to identify shortcomings in existing software systems.

3.6.3. Comparison with research conducted by the international community

Other researchers proposed AO modularity-related metrics. Zhao and Xu also proposed metrics for quantifying coupling and cohesion in AO software (Zhao, 2004; Zhao and Xu, 2004). Their metrics are based on a dependence model for AO software that consists of a group of dependence graphs. Each graph can be used to explicitly represent various dependence relationships at different levels of an AO program. Ceccato and Tonella (2004) also proposed a suite of AO metrics. For instance, their suite includes metrics such as Crosscutting Degree of an Aspect (CDA) and Coupling on Advice Execution (CAE). Their metrics are more specific since they consider certain constructs of some AO languages, such as pointcut, advice, and intertype declarations. Moreover, all these AO metrics can be seen as complementary to the metrics proposed by the AOSD-BR community and measure different facets of AO software modularity.

Researchers from the international community also proposed metrics for quantifying separation of concerns. Apart from the suite proposed by Sant'anna et al. (2003), one of the most significant works on this topic was conducted by Eaddy et al. (2008). They presented two concern-driven metrics that capture different facets of what they call concern concentration and component dedication: Degree of Scattering and Degree of Focus. These metrics can be employed at different granularity levels ranging from lines of code to architectural software components Eaddy et al. (2008) also carried out an experiment which involved their metrics and two of the metrics proposed by Sant'anna et al. (2003): Concern Diffusion over Components (CDC) and Concern Diffusion over Components (CDO). Their experiment aimed at testing the hypothesis that the more scattered a concern's implementation is, the more likely it is to have defects. They found a moderate to strong correlation between CDC and CDO metrics and defects, which suggested that scattering may cause or contribute to defects. In fact, they found a stronger correlation for CDC and CDO than for their own metrics.

3.6.4. *Open issues*

There are several opportunities for future work on concern-driven measurement. First, there is still a need for empirically validating concern-driven metrics in terms of their correlation with external quality attributes, such as change-proneness (Silva et al., 2011). The use of visualization techniques to support concern-driven measurement has already been exploited (Carneiro et al., 2010). For instance, a tool called SourceMiner provides four categories of code views for concern properties, namely: (i) concern's package-class-method structure, (ii) concern's inheritance-wise structure, (iii) concern dependency, and (iv) concern dependency weight. However, there is still much to be done in the area of concern visualization. For instance, additional enhanced views could be provided and studies could be conducted to verify whether visual support helps in software development and maintenance tasks.

3.7. *Product-lines and frameworks*

A Software Product Line (SPL) is a set of software intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission, and that are developed from a common set of core assets in a prescribed way (Clements and Northrop, 2001). Within SPL architectures, frameworks are a usual implementation technique used to modularize core assets. A framework is defined as a reusable, "semi-complete" application that can be specialized to produce custom applications (Fayad and Schmidt, 1997). Since the modularization of features in SPLs tends to have crosscutting and tangled implementation (Mezini and Ostermann, 2004) over code assets, AOSD has been explored as a possible implementation strategy for SPLs in order to provide modular feature implementation. Indeed, the AOSD-BR community has conducted research leveraging AO concepts, techniques, and tools in order to address key challenges in SPL and framework development. Conversely, this community has also explored how AOSD benefits from SPL and framework concepts, techniques, and tools.

3.7.1. *Key challenges*

Key research challenges in the interplay between SPLs, framework development, and AOSD are the following:

(i) *Variability management*: How can aspects help with the modularization, composition and customization of SPL core assets in terms of features to address different product configurations? How can aspect-based tools help to manage the inherent complexity of handling a potentially vast number of SPL variants?
(ii) *Adoption strategies*: If an organization decides to shift to SPL development, how should it proceed? Would it leverage existing products? If so, how should this be implemented and what is the role played by aspects? Once an SPL is in place, how should it evolve? How should aspects within this SPL evolve?
(iii) *Taxonomy and development of frameworks*: What are the other framework types in the AOSD context ? Are there specific guidelines for the design and implementation of these kinds of frameworks? How should an application framework developed with the support of AOP be characterized?

3.7.2. *Addressing the challenges*

In order to tackle the issues mentioned previously, the AOSD-BR community has provided: (i) aspect-oriented implementation guidelines and tool support for dealing with the variability management and SPL adoption strategies challenges; and (ii) a taxonomy and development approaches for addressing the challenges associated with the adoption of aspect-oriented techniques in framework implementation. Details about these research works are presented in the following.

SPL implementation guidelines: The AOSD-BR community has addressed portability issues in industrial-strength mobile game development (Alves et al., 2005, 2007). These works propose the use of aspects to modularize the implementation of device-specific features by presenting guidelines on how to systematically apply a catalog of code-level refactorings to bootstrap and evolve a product line. This catalog and the associated strategy to apply it were evaluated in case studies involving three mobile game SPLs and were later extended to feature model and configuration knowledge levels. Additional evaluation was performed on different domains such as media management on mobile devices and automatic test case generation tools (Alves et al., 2006; Neves et al., 2011).

Additionally, Pacios et al. (2006) proposed guidelines to use AOP to implement the functional features of an SPL in an incremental way. This work was further evolved into an approach named AIPLE-IS (Braga et al., 2007), with focus on the information systems domain. The basic idea is to start by developing the SPL core assets and then incrementally build optional and alternative features using AOP. A case study in the domain of systems for psychology clinics was performed and the resulting SPL has been built in three incremental steps, with a clear separation of several variant features using AOP. Further empirical work has assessed the benefits and drawbacks of AOP in evolving SPLs (Figueiredo et al., 2008a).

Tool support: Tools are essential to support the use of AOP in SPL development. Indeed, the task of instantiating SPLs is particularly complex and error-prone, especially when details about crosscutting aspects are required. Brazilian researchers have contributed to addressing this critical issue. Captor-AO is a configurable application generator that implements AOP concepts and is used to support the development of domain-specific systems (Pereira et al., 2008). Captor-AO is also used in an approach where aspects are employed to implement SPL crosscutting features with the goal of improved reuse not only within the SPL domain, but also across different domains (Braga et al., 2010). The idea is to isolate these crosscutting features into 'crosscutting domains', so that features scattered across different SPLs are easily located and reused. Captor-AO was evaluated through the instantiation of SPLs from several domains, such as: a business resource management SPL – an SPL that encompasses sub-domains such as sales, rental and repair of goods and services; and MobileMedia – an SPL that provides functionalities of media management on mobile devices (Figueiredo et al., 2008a).

Additionally, the FLiP tool was developed to implement the previously mentioned catalog of AO-based refactoring templates and the associated strategy to support the extractive and the reactive SPL adoption strategies. The tool provided semi-automated support for the task of modularizing portability-based features in three case studies involving industrial-strength mobile games (Alves et al., 2008; Soares et al., 2008).

Framework taxonomy: Addressing variability management implies properly organizing and classifying core and variant assets. In particular, the interplay of aspects and frameworks primarily demands an appropriate taxonomy. Despite the number of research works conducted exclusively on frameworks in the context of AOP, most of them refer to this kind of framework using different terms, such as “aspect-oriented frameworks”, “aspect frameworks”, “aspect-based frameworks”, “framework of aspects”, “reusable aspects” and “aspect-oriented application frameworks”. The lack of consensus about the real meaning of these terms leads researchers to use the same term to represent different things or different terms to represent the same thing. The AOSD-BR community has addressed this issue. In particular, Camargo and Masiero (2005) proposed a classification and terminology for frameworks developed within the AO context.

The root term of the terminology is “aspect-oriented framework”, which designates any framework that uses aspects in its internal structure. This category is broken down into two others: Aspect-oriented Application Frameworks (AOAFs) and Crosscutting Frameworks (CFs). AOAFs are frameworks whose instantiation process generates a complete application and use aspects in their architectures to modularize specific concerns. On the other hand, CFs encapsulate in a flexible and abstract way a specific crosscutting concern, such as persistence, distribution, concurrency, business rules, and design patterns. The idea is to make the reuse of this kind of concern a more systematic and controlled task (Camargo and Masiero, 2008). CFs are further divided into Context-Dependent CFs (CD-CF) and Context-Independent CFs (CI-CF) (Camargo and Masiero, 2005). The CI-CFs are those that do not need to capture data from the base code to work properly. In general, this kind of CF acts as an observer, as it simply crosscuts the base code without interfering or interacting with it. However, CD-CFs need to capture values or objects from specific points of the base code to work properly. Examples of CD-CFs are a Caching CF that needs to capture the objects from the base code which need to be registered and stored into the cache, and a Pooling CF that needs to capture Connection objects from the base code to populate a repository.

The AOSD-BR community has proposed two different approaches to the development of AOAFs and CFs, which are detailed next.

Framework modularization and development guidelines: In order to promote the modularization of AOAFs, a novel approach has been proposed to address the design and implementation of object-oriented frameworks with aspects (Kulesza et al., 2006d,b). It contributes to deal with the following framework modularization challenges: (i) difficulty in modularizing optional features in OO frameworks; (ii) crosscutting feature compositions in framework integration; and (iii) complexity of object collaboration. In the proposed approach, an OO framework is responsible for specifying and implementing not only its common and variable behavior using OO classes: it also exposes a set of extension join points (EJPs), which can be used to also extend its core functionality. These join points can be used with three different purposes: (i) to expose a set of framework events that can be used to notify or to facilitate a crosscutting integration with other software modules (such as frameworks or components); (ii) to offer predefined execution points spread and tangled in the framework that can be included in the implementation of optional features; and (iii) to expose a set of join points in the framework classes that can have different implementations of a crosscutting variable functionality. Explicit guidelines on how to implement EJPs in the AspectJ language have been provided (Kulesza et al., 2006b). EJPs enforce design rules between OO core and aspect-based assets and prevent architectural erosion in framework-based SPLs. Finally, Kulesza et al. (2006c, 2007) also propose a complementary model-based generative approach that enables the systematic derivation of aspect-oriented application frameworks or product lines by defining how to map crosscutting features to aspects across different artifacts.

Regarding the design of CFs, a set of guidelines (Camargo and Masiero, 2004) and a UML profile were also proposed (Uetanabara et al., 2009, 2010). Besides, a pattern called Data Catcher (Camargo and Masiero, 2008) was created specifically for designing CD-CFs. When this pattern is used, the framework architecture is explicitly separated into two parts: one that deals with the composition mechanisms and another that deals with the functional variabilities. In the composition part, there are several composition alternatives facilitating coupling with existing base code. The functional part can be designed using classical OO patterns. When using this pattern, the reuse process is facilitated because the application and the domain engineer can deal with both composition and functional parts separately.

3.7.3. Comparison with research conducted by the international community

Regarding guidelines, prior research also investigated the use of AOP for building SPLs (Anastasopoulos and Muthig, 2004). AOSD-BR research complements this work by considering the modularization of features in industrial-strength applications, explicitly specifying refactorings to build and evolve SPLs, and raising issues in AspectJ that need to be addressed to foster widespread application in the mobile games domain. Feature modularization with aspects has also been addressed elsewhere (Lee et al., 2006) with a focus on domain analysis and handling feature dependencies, but not addressing adoption strategies as Brazilian researches do. Another important observation is that other works that use AOP to modularize features, such as (Mezini and Ostermann, 2004), deal with a single SPL targeting a specific domain. On the other hand, the AOSD-BR community has dedicated effort to address multiple domains (multiple SPLs), easing the evolution of products in several domains at the same time (Pereira et al., 2008; Braga et al., 2010). Another approach concerning feature modularization is Feature Oriented Programming (FOP) (Prehofer, 1997), which essentially consists of non-quantifiable aspects. FOP and AOP were later systematically compared and a symbiosis proposed (Apel et al., 2005, 2008).

In terms of framework modularization, EJPs (Kulesza et al., 2006b) are a kind of specialization of the concept of crosscutting interfaces (XPIs) (Sullivan et al., 2005) that establishes extension contracts between the framework classes and a set of aspects extending the framework core functionality. Although there are several works that use frameworks in the AOP context (Pinto et al., 2002; Lobato et al., 2008; Batra and Dahiya, 2009; Mortensen and Ghosh, 2006; Kulesza et al., 2006d; Santos et al., 2007; Rashid and Chitchyan, 2003; Cunha et al., 2006; Bynens et al., 2010), most of them lack an accurate and consistent definition of AOP-based framework terminology. Besides, the guidelines (Camargo and Masiero, 2004), patterns (Camargo and Masiero, 2008) and UML profiles (Uetanabara et al., 2009, 2010) proposed to design CFs complement other research work found in the literature. For example, the profile presented by Uetanabara (Uetanabara et al., 2009, 2010) extends Evermann’s profile (Evermann, 2007) with intrinsic CFs characteristics.

Incremental development processes focused on AOP have also been proposed by the international community. Examples include the works of Loughran et al. (2004) and Apel et al. (2006). The first proposes an approach that joins framing and AO techniques to integrate new features in product lines. This solution provides parameterization and reconfiguration support for the feature aspects, although the framing technology has the disadvantage of being less intuitive, requiring previous knowledge about the

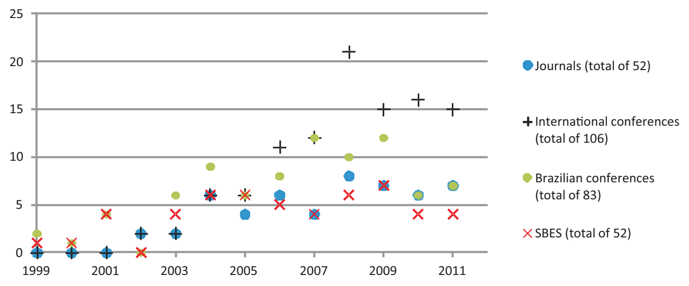

Fig. 2. AOSD-BR publications.

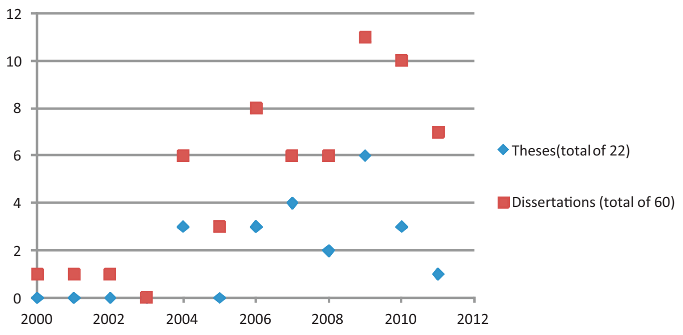

Fig. 3. AOSD-BR theses and dissertations.

subsequent features. An advantage is to parameterize aspects at runtime. The second one (Apel et al., 2006) proposes the integration of aspects and features at the architectural level through aspectual mixing layers (AMLs), also aiming at incremental software development. The approach proposed by AOSD-BR (AIPLE-IS) is more focused on the process itself rather than on the architecture, and there is no need to anticipate features, as every step revisits the previous model before integrating new features.

3.7.4. Open issues

Prospective research on the interaction between AOSD and SPL engineering includes providing sound and evidence-based guidelines and tools on how to combine different strategies for designing and implementing features. The main reasons for that are to reach feasible feature modularity, good stability of SPL assets, and low cost of usage and stability. In particular, the benefits of virtual separation of concerns (Kästner et al., 2008) are currently being investigated by an automated approach (Valente et al., in press). The idea is that, from code-level seeds specified by the developer or architect, the code related to a feature is colored. The feature is then assigned a particular color, which the developer may or not want to see, depending on his/her interest. Another promising research thread is exploring the use of formal methods to ensure in a scalable way the checking of properties, such as type safety and non-functional requirements of SPL artifacts at different abstraction levels (Apel et al., 2010; Siegmund et al., 2011; Ghezzi and Sharifloo, 2011). In this direction, there is a preliminary work from the AOSD-BR community addressing safety of SPL composition (Teixeira et al., 2011).

4. AOSD-BR research impact

In this section, we quantify and assess the efforts of the AOSD-BR community in several areas, such as publications and their impact, and training of human resources (Section 4.1). We also compare the scientific production of the AOSD-BR community with that of the international AOSD and the Brazilian software engineering research communities (Section 4.2). Data gathering and validation has been performed with the help of AOSD-BR researchers (AOSD-BR-Community, 2011).

4.1. Intellectual production, collaboration, and human resources training

Table 3 depicts raw/absolute numbers related to AOSD research in Brazil, including publications, theses, and international cooperation projects. The selected papers: (1) had at least one author working in a Brazilian institution when the paper was written; (2) brought contributions to AOSD or related topics; and (3) are indexed by Google Scholar. The growth of the AOSD community can be noticed by the number of international research projects developed in cooperation among Brazilian and foreign groups, as well as an increase in the number of papers resulting from such cooperative research. This is more evident when looking at the number of conference papers (Table 3) – more than 50% were accepted by international conferences.

Figs. 2 and 3 depict publications and theses growing from the AOSD-BR community over the years. It is interesting to notice: (i) the increase in the number of publications and theses since the formation of AOSD-BR community in 2004; and (ii) the high number of international publications that, since 2006, has been greater than the number of publications at Brazilian venues, showing once more *the internationalization process of this community*. Fig. 2 also shows the number of SBES papers from the AOSD-BR community. The growth of the community and its research can be clearly seen by the increase in the number of AOSD-related papers at SBES over the years. In fact, most of the national AOSD-BR papers were published at SBES (53 out of 86 papers), as shown in Table 3.

4.2. Research impact quantification

The research impact of the AOSD-BR community can be quantified in terms of: (i) the increasing number of accepted papers in top SE conferences around the world; (ii) the number of citations; and (iii) the number of best paper awards and nominations received in international and national conferences. In addition, other relevant results are reported, such as the adoption of AO applications as benchmarks by the AOSD research community, and the participation of Brazilian researchers in program and organizing committees of several international conferences and workshops.

In the next sections we perform an analytical analyses based on numbers collected from several top SE conferences that publish AOSD-related papers.

4.2.1. Analyzing the international impact of the AOSD-BR community

In order to allow the analysis of the scientific production of the AOSD-BR community in the international context, we investigated the following research question: *What is the impact of the AOSD-BR community compared to the AOSD international research work?* The answer for this question was developed by analyzing the number of papers and citations of the AOSD-related papers in top SE conferences from the AOSD-BR community and the other international AOSD research groups. This analysis helped us to evaluate the impact and quality of the AOSD-BR international papers.

When conducting our analysis, we looked for the number of citations of AOSD-related papers from 1999 to 2011 in six top SE conferences: International Conference on Software Engineering (ICSE), International Conference on Software Maintenance (ICSM), Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), European Conference on Object-Oriented Programming (ECOOP), Foundations on Software Engineering (FSE), and Aspect-Oriented Software Development (AOSD). We selected, from these conferences' proceedings, only papers that

Table 3
AOSD-BR in numbers.

Journal papers	Conference papers	SBES papers	Theses	Dissertations	Int'l cooperation projects
52	189*	53	22	60	12

* 103 at international conferences, 86 at national ones (including SBES).

address non-conventional modularization techniques.[3] For the AOSD conference, we considered all the papers because they are related to advanced modularization techniques. The complete list of the search result is available elsewhere (AOSD-BR-Community, 2011). With this data, we compared AOSD-BR publications against those from research groups from other countries, in terms of the numbers of papers and citations. The collected results show a high international impact of the AOSD-BR community in the international AOSD context.

Fig. 4 depicts the number of AOSD-related papers per country. In order to quantify such number, we considered the first author's affiliation country (AOSD-BR-Community, 2011). The figure shows that papers from USA and Canada institutions are hegemonic in these conferences, but there is a significant number of papers from Brazilian institutions. It is interesting to notice the high number of AOSD-BR papers at the AOSD and ICSM conferences. They are greater than several major European countries and always the greatest among Latin-American countries.

We also investigated the number of citations of the selected AOSD-related papers from the six top SE conferences in order to assess the impact of the AOSD-BR scientific production compared to the international community (AOSD-BR-Community, 2011). Figs. 5–9 depicts the top 10 cited papers, according to Google Scholar, at each of the conferences, except for FSE, where no AOSD-BR appears among the 10 most cited papers. The figures show that there is at least one AOSD-BR paper in each list.[4] This suggests that research conducted by the AOSD-BR community has given significant contributions to the AOSD international community. In addition, for the OOPSLA, ICSM and AOSD conferences, the AOSD-BR papers rank among the top 5 most cited AOSD-related papers, thus providing even more evidence of the aforementioned significance.

Finally, Fig. 10 shows the list of authors that have published more often at the AOSD conference. We show only the top-18 authors, which are the ones with 4 or more papers published. It is very exciting to note that the researcher that published the most papers at AOSD is Professor Alessandro Garcia from PUC-Rio, one of the most active researchers of the AOSD-BR community. The list was specified by considering all the authors of every paper from all editions of the AOSD conference (AOSD-BR-Community, 2011).

4.2.2. Analyzing the impact of the AOSD-BR community compared with the SE Brazilian community

In our analysis, we also collected data to investigate the results and impact of the AOSD-BR community when compared with the overall SE Brazilian community. Our analysis was guided by the following research questions: (i) *What is the impact of the AOSD-BR community when compared with the SE Brazilian community in the international context?* and (ii) *What is the impact of the AOSD-BR community when compared with the SE Brazilian community in the national context?* The next two subsections explore the analysis of the collected results for these questions, respectively.

[3] The ACM and IEEE digital libraries and DBLP bibliography were used to browse and to read the content of the papers in order to determine if they are AOSD-related papers or not. In addition, Google Scholar was used to quantify the number of citations of each paper.

[4] Such papers are indicated by the "(AOSD-BR)" label.

Table 4
Brazilian papers on top SE conferences.

Top SE conferences	Average paper acceptance	Papers from the Brazilian Community (1999–2011)	Papers (percentage) from AOSD-BR Community (1999–2011)
AOSD	24%	9	9 (100%)
ECOOP	17%	3	2 (67%)
FSE	19%	2	1 (50%)
ICSE	15%	4	2 (50%)
ICSM	34%	9	4 (44%)
OOPSLA	24%	2	1 (50%)

Impact of AOSD-BR and Brazilian SE communities in the international context

In order to analyze the impact and relevance of the AOSD-BR research results compared with the SE Brazilian community in the international context, we quantified the visible increase of accepted papers in top SE conferences over the last years. Table 4 shows the number of papers published by the Brazilian research community from 1999 to 2011 in top SE conferences.[5] In addition, it also illustrates the general acceptance rate of the conference, the number and percentage of the total number of these Brazilian papers that are a contribution from the AOSD-BR community.

In most cases, as we can see in Table 4, the contribution of our community is equal or superior to 44% of the total number of papers published by Brazilian researchers in these conferences. In this way, we can observe that AOSD-BR has significantly contributed to improve the acceptance rate of Brazilian papers in these conferences. We can also notice that the AOSD-BR community has gradually increased the number of accepted papers in the premier AOSD conference from 2002 to 2011. It is important to emphasize that all these conferences have a strict acceptance rate and the papers published there represent the state-of-the-art of outstanding and high-quality research developed worldwide. These publications also demonstrate the maturity and quality of research work developed by the AOSD-BR community.

Another view of such papers is presented by Table 5, which accounts their citation numbers (AOSD-BR-Community, 2011). We mark AOSD-BR papers with the label (AOSD-BR). It is interesting to note that, except for FSE, the most cited Brazilian paper in each conference is always a paper from the AOSD-BR community. Also note that we excluded AOSD papers from this table since all AOSD Brazilian papers are from the AOSD-BR community.

Impact of AOSD-BR and Brazilian SE communities in the national context

The analysis of the impact and relevance of the AOSD-BR research production compared with the SE Brazilian community in the national context was conducted using data collected from the Brazilian Symposium on Software Engineering (SBES), the main SE scientific conference in Brazil. The analysis was performed by quantifying: (i) the number of international citations for all papers published in SBES since its first edition in 1987; and (ii) the most prolific SBES authors in all editions.

Table 6 shows SBES top-10 cited papers, where AOSD-BR papers titles are labeled as (AOSD-BR). One of these papers, "*On the*

[5] Only papers with at least one author from a Brazilian institution were considered.

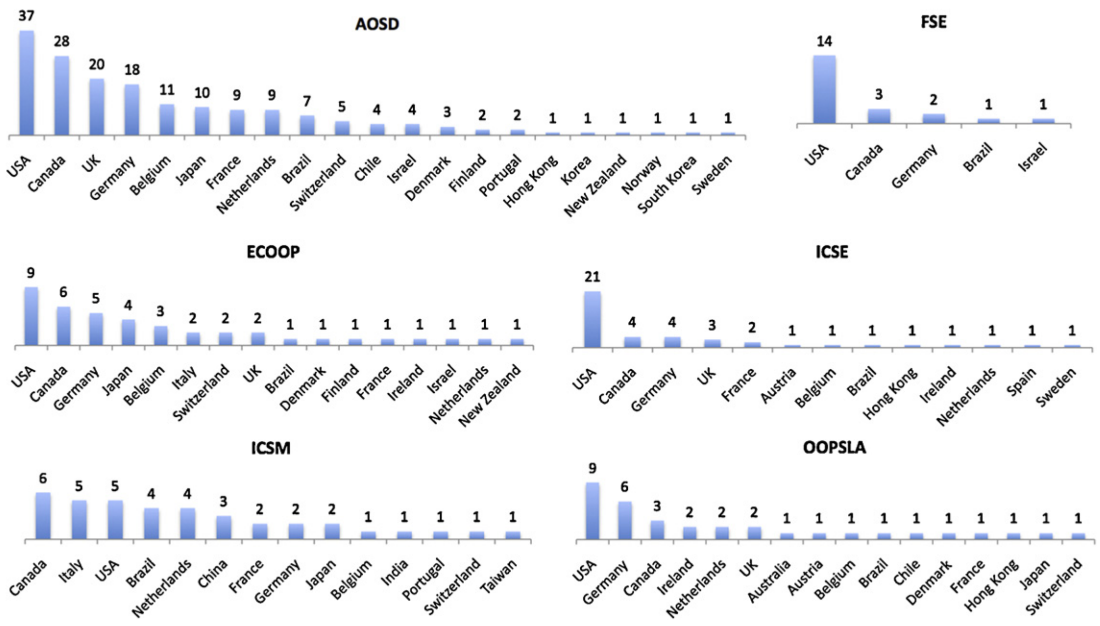

Fig. 4. AOSD-related papers at top SE conferences per country.

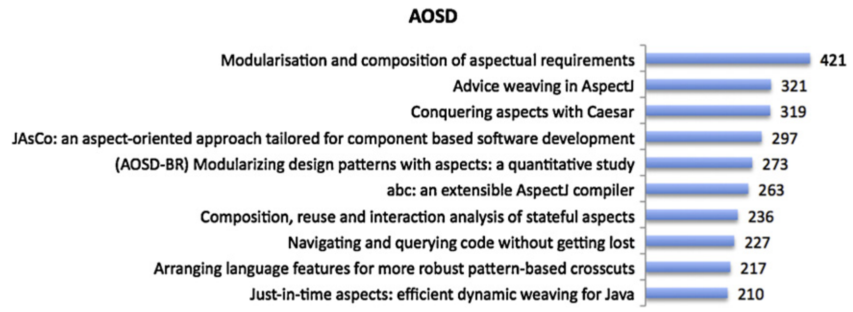

Fig. 5. Top 10 cited papers at AOSD, according to Google Scholar.

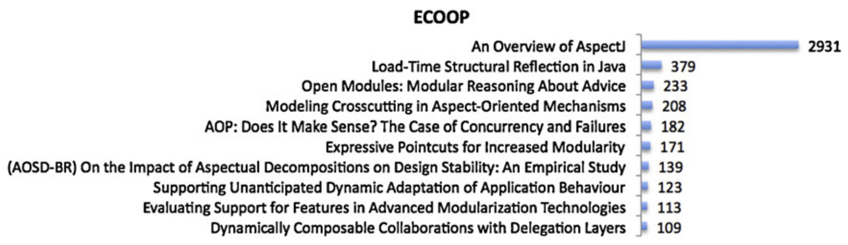

Fig. 6. Top 10 cited AOSD-related papers at ECOOP, according to Google Scholar.

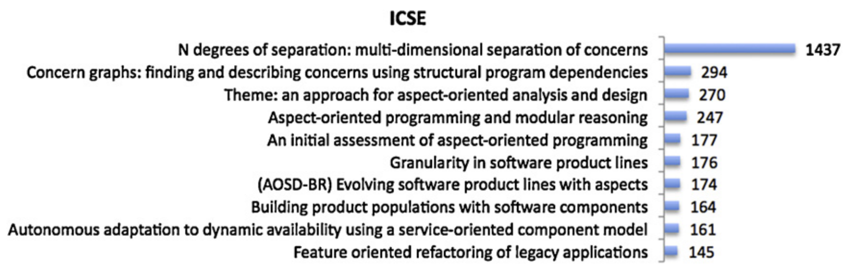

Fig. 7. Top 10 cited AOSD-related papers at ICSE, according to Google Scholar.

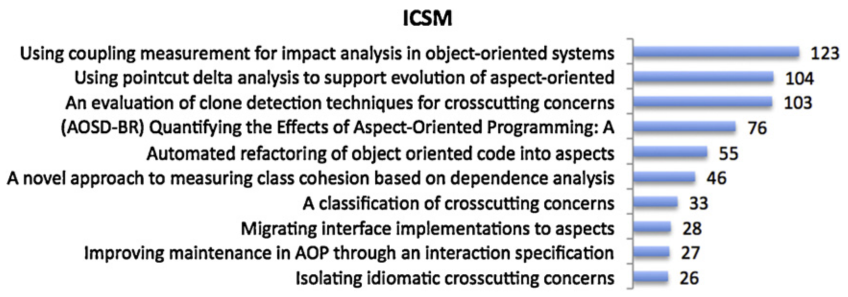

Fig. 8. Top 10 cited AOSD-related papers at ICSM, according to Google Scholar.

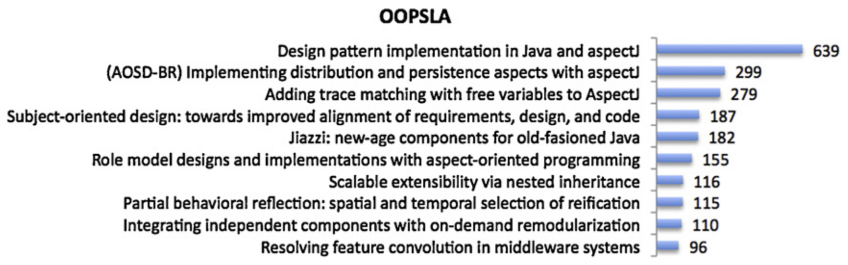

Fig. 9. Top 10 cited AOSD-related papers at OOPSLA, according to Google Scholar.

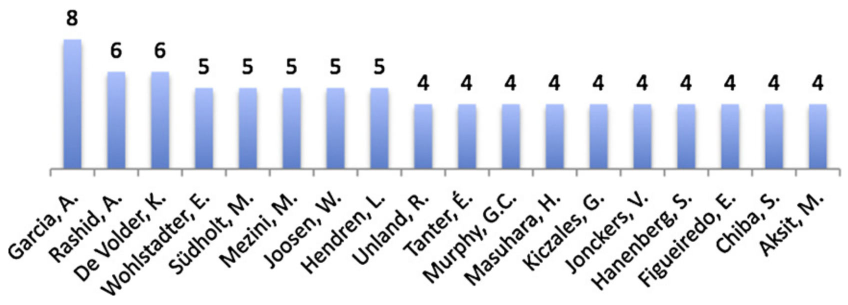

Fig. 10. Number of papers of the top-18 authors at the AOSD conference.

reuse and maintenance of AO software: An assessment framework" (Sant'anna et al., 2003), presents a metrics suite to compare AO and OO implementations, including new and original separation of concerns (SoC) metrics. As shown in the table, this is the most cited SBES paper considering all SBES editions. It illustrates that a paper published in SBES can be read and recognized by the international SE research community. Many of the empirical assessments developed by the AOSD-BR community (Figueiredo et al., 2008a; Greenwood et al., 2007) adopted this metrics suite and helped to disseminate the development of several

Table 5
Brazilian papers citation on top SE conferences, according to Google Scholar.

Conference	Paper title	Citations
ECOOP'07	(AOSD-BR) On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study	139
ECOOP'03	A Refinement Algebra for Object-Oriented Programming	49
ECOOP'08	(AOSD-BR) Assessing the Impact of Aspects on Exception Flows: An Exploratory Study	39
FSE'04	How a good software practice thwarts collaboration: the multiple roles of APIs in software development	74
FSE'06	(AOSD-BR) Exceptions and aspects: the devil is in the details	56
ICSE'08	(AOSD-BR) Evolving software product lines with aspects: an empirical study on design stability	174
ICSE'08	An empirical study of software developers' management of dependencies and changes	40
ICSE'10	(AOSD-BR) An exploratory study of fault-proneness in evolving aspect-oriented programs	9
ICSE'08	Improving the handsets network test process via DMAIC concepts	2
ICSM'06	(AOSD-BR) Quantifying the Effects of Aspect-Oriented Programming: A Maintenance Study	76
ICSM'04	RefaX: A Refactoring Framework Based on XML	19
ICSM'05	Comparative Analysis of Porting Strategies in J2ME Games	18
ICSM'07	(AOSD-BR) Extracting Error Handling to Aspects: A Cookbook	17
ICSM'05	A Risk Taxonomy Proposal for Software Maintenance	15
ICSM'07	(AOSD-BR) JAT: A Test Automation Framework for Multi-Agent Systems	7
ICSM'08	(AOSD-BR) Non-invasive and non-scattered annotations for more robust pointcuts	1
ICSM'11	(AOSD-BR) Identifying overly strong conditions in refactoring implementations	0
ICSM'11	(AOSD-BR) Structural conformance checking with design tests: An evaluation of usability and scalability	0
OOPSLA'02	(AOSD-BR) Implementing distribution and persistence aspects with AspectJ	299
OOPSLA'99	Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality	148

Table 6
Top-10 cited SBES papers, according to Google Scholar.

Year	Paper title	Citations
2003	(AOSD-BR) On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework	191
2001	CRE: A Systematic Method fot COTS Components Selection	96
2000	Ambientes de Desenvolvimento de Software Orientados a Dominio	52
1994	Constrained Mutation in C programs	37
2001	Using Objects and Patterns to Implement Domain Ontologies	32
1992	Estrategia data-driven no desenvolvimento de software	31
1997	Um modelo Fuzzy para avaliacao da qualidade de software	30
2002	Formalizing Object-Oriented Design Metrics upon The UML Meta-Model	25
2003	(AOSD-BR) Adapting the NFR Framework to Aspect-Oriented Requirements Engineering	24
2005	(AOSD-BR) Taming Heterogeneous Aspects with Crosscutting Interfaces	24

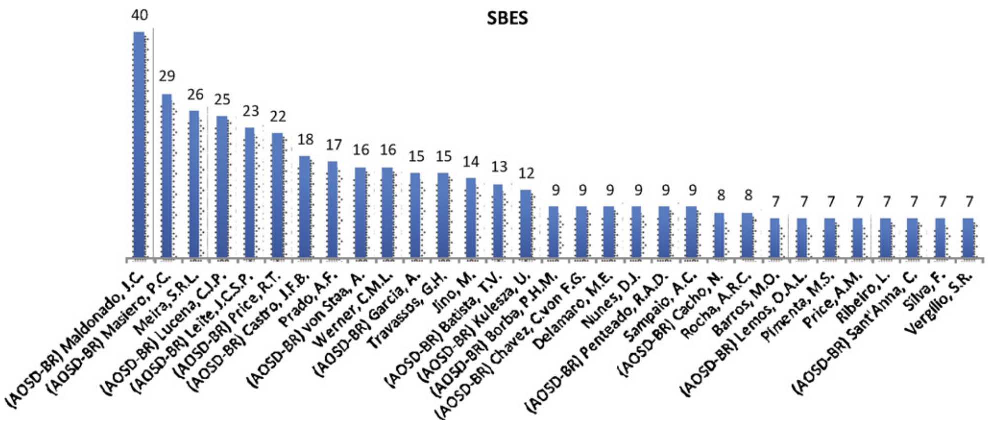

Fig. 11. Publications per author at SBES.

quantitative comparative studies between AO and OO implementations. Besides the most cited SBES paper, the AOSD-BR community has other two papers among the top-10 most cited papers, as shown in Table 6.

We also analyzed the data collected from SBES to identify the authors that have published most often at this conference since its first edition (1987). Fig. 11 depicts the list of the 30 researchers that have more papers published in the SBES conference (AOSD-BR-Community, 2011). A total of 16 researchers from this list are active authors from AOSD-BR community; the label (AOSD-BR) prefixes their names. It is important to mention that we account all papers from each author, not only AOSD-related papers. The number also shows that the AOSD-BR community comprises experienced and very active researchers. This might be one of the reasons for the community success.

Last but not least, there is an impressive number of AOSD-BR papers in SBES. A total of 52 papers have been published in SBES between 1999 and 2011. This number represents about 18% of the 290 papers published in SBES from 1999 to 2011, and 9.5% of the 547 papers published in the lifetime of SBES. In addition, if we consider only the papers published since the birth of AOSD-BR, from 2004 to 2011, this percentage increases to 22.4% of all SBES papers in this period. Again, it reflects the active presence of the AOSD-BR community over the last years at the main SE conference in Brazil.

4.2.3. *AOSD-BR most cited papers, awards & benchmarks*

Another important quality indicator for the research conducted by the AOSD-BR community is the number of citations of papers published by this community in international and national SE conferences. Two research contributions (Soares et al., 2002; Garcia et al., 2006b) from the community have received almost 300 citations[6] each. The first one (Soares et al., 2002) is a contribution from the Software Productivity Group (SPG) of UFPE that illustrates the modularization of the persistence and distribution crosscutting concerns using AspectJ. The second one (Garcia et al., 2006b) was the first relevant international publication of the SE Laboratory (LES) from PUC-Rio, which quantifies and compares OO and AO implementations of classical design patterns using software internal metrics. There are also several papers (Figueiredo et al., 2008a; Greenwood et al., 2007; Sant'anna et al., 2003) from our community that have received more than 100 citations, even though many of them have been published only recently.

Several other papers written by our community and published in international SE conferences have received between 40 and 100 citations. Finally, many papers published in SBES have been cited by other international and national papers. A complete list of citations to papers from the AOSD-BR research community can be found elsewhere (AOSD-BR-Community, 2011). Also, as a result of this high number of citations to their publications, researchers from the AOSD-BR community are listed in the Top-100 SE research ranking considering the citations over the last five years in the Microsoft Academic Search engine.

A number of AOSD-BR papers have been nominated to the best paper award in the SBES – the main Brazilian SE symposium. Our community has received 12 nominations for the SBES best paper award along the last decade (2001–2011), and received the best paper award on three occasions (Sant'Anna et al., 2004; Chavez et al., 2005; Carneiro et al., 2010). This reflects the high quality and

[6] Source: Google Scholar.

recognition that research work conducted by AOSD-BR groups has garnered. A list of all awards received by AOSD-BR research work is available elsewhere (AOSD-BR-Community, 2011).

The AOSD-BR community has been involved in the development and evolution of two application benchmarks used by several researchers around the world: HealthWatcher (Soares et al., 2002) and MobileMedia (Figueiredo et al., 2008a). HealthWatcher (Soares et al., 2002) is a real web information system that stores information about the public health system and registers complaints from citizens. HealthWatcher was first implemented in Java, and the Software Productivity Research Group from UFPE derived an AspectJ implementation from it. From this point, both Java and AspectJ versions were evolved, resulting in several versions that have been used in several studies at Brazilian and foreign institutions. For example, Health Watcher was the recommended case study for the papers submitted to the Early Aspects workshop at ICSE 2007 (Workshop in AO Requirements Engineering and Architecture Design). Currently, HealthWatcher encompasses nine subsequent releases that provide support for the execution of AO maintainability studies. Our main partner in the evolution of HealthWatcher is the Lancaster University, where the versions of HealthWatcher are publicly available[7].

MobileMedia (Figueiredo et al., 2008a) is a Software Product Line (SPL) for media management on mobile devices. MobileMedia was first implemented in Java and AspectJ by Brazilian students at Lancaster University. After a few years, the current version of MobileMedia gathered contributions from a set of 16 institutions, in many different ways. For example, some institutions' members directly provided initial MobileMedia artifacts that followed AO design guidance cooperatively developed by them. Other partners contributed by reviewing the artifacts according to their area of expertise. Experts have contributed with the use of specific techniques to produce the MobileMedia artifacts, such as goal models and architecture descriptions. Currently, MobileMedia encompasses eight subsequent releases that support the execution of studies of AO SPL maintainability. In addition, MobileMedia has already been successfully used in a number of assessments of AOSD (Figueiredo et al., 2008a, 2009c; Burrows et al., 2010b; Ferrari et al., 2010).

5. Conclusions

This paper reported on a study to characterize the AOSD-BR research community in terms of its impact on SE research in Brazil and AOSD research abroad. This characterization has been performed with respect to three research questions that were presented previously and from three different perspectives.

First of all, we have presented a timeline for the AOSD-BR community, a historical chronology highlighting important milestones conquered by the community. This chronology can be used as a roadmap to help the development and evolution of new research communities around the world. In addition, it also contributes to illustrate how our community has evolved over the last decade together with the AOSD international community;

An overview of the research developed by our community in several prominent SE topics was also reported, and confronted with AOSD challenges, related work from international partners and open research issues. The research overview summarizes the main contributions of the AOSD-BR community, so far, and can be useful to guide newcomers and to identify gaps and new opportunities for AOSD research.

Finally, the impact of the research developed by the AOSD-BR community was quantified in terms of the number of accepted papers in top SE conferences around the world, the number of citations, and the number of best paper awards and nominations received in international and national conferences, among others. This perspective provided an objective view about the importance of the AOSD-BR community in terms of the quality and quantity of its more relevant contributions.

Besides these three dimensions of assessment and related contributions, the study methodology can itself be used as guidance to support similar studies in other research areas.

5.1. Main results

In Section 1, three research questions were presented and discussed throughout the paper. The highlights of our study, in terms of the results for such questions are:

(i) **International IMPACT of the AOSD-BR community.** In the international context, the AOSD-BR community has given influential contributions to six top SE engineering conferences. Our analysis has revealed that many of the AOSD-BR technical papers published in these conferences are listed among the top 10 most cited AOSD-related papers from 1999 to 2011. Finally, we found that Professor Alessandro Garcia from PUC-Rio/Brazil is the author with the highest number of publications in the AOSD conference, the premier international conference on modularity.

(ii) **Impact of AOSD-BR and Brazilian SE communities in the international context.** Regarding the impact of the AOSD-BR community against other SE Brazilian researchers, our analysis has shown that at least 44% of the total number of Brazilian publications in six top SE conferences was a direct result of the scientific production of the AOSD-BR community. Considering all the papers written by Brazilian researchers in these top SE conferences, from 1999 to 2011, we also found out that the AOSD-BR papers have a more expressive number of citations than the non-AOSD Brazilian papers (AOSD-BR-Community, 2011), except for FSE.

(iii) **Impact of AOSD-BR and Brazilian SE communities in the national context.** We have also analyzed data collected from SBES – the main Brazilian SE conference – in order to characterize the impact of AOSD-BR in the national context. Our main findings were: (i) the most cited paper from all SBES editions was written by AOSD-BR researchers, and two other papers from our community are listed as top 10 most cited; and (ii) an impressive number papers (52) were published by the AOSD-BR community at SBES from 1999 to 2011, which represents 18% of SBES papers published in the same period and 9.5% of all SBES papers published since the first edition of the conference in 1987. Finally, we have also found that 16 researchers from the list of top 30 most prolific authors of SBES are AOSD-BR members and have significantly contributed to the growth, consolidation and qualified production of the community.

5.2. Lessons

This study has also taught us important lessons which, we believe, should be considered by young and senior researchers when proposing and developing new research communities.

Support from established forums. The major forum that has fostered the AOSD-BR community and related research is the Brazilian Symposium on Software Engineering (SBES). Most of the AOSD research work in Brazil has been published at SBES and most of

[7] http://bit.ly/HealthWatcher.

the AOSD-BR researchers have active participation in the SBES conference. Since 2004, SBES (now CBSoft[8]) hosted the WASP/LA-WASP workshops. These workshops provided the elements that allowed the community to emerge, become mature, active, highly collaborative, and internationalized.

Presence of experienced researchers. The AOSD-BR community aggregates several experienced and very active researchers from different software engineering subareas since its initial steps. This might be one of the reasons for the success of the community.

International "exposure". A paper published at SBES can be read and recognized by the international SE research community (Sant'anna et al., 2003), specially if it is written in English. This is an issue that some countries have to discuss: the language used to write papers and theses in order to make their research public and useful worldwide.

Interaction fosters collaboration. The AOSD-BR experience demonstrates that a rich interaction might promote great opportunities gathering researchers from different universities, countries and subareas. This paper is important proof for that.

5.3. Future work

AOSD research has been a hot topic for almost 15 years worldwide and more than 10 years in Brazil, with a growing number of high-quality publications, academic theses and projects, and increasing international impact. Several challenges were presented for different subareas in which the AOSD-BR community has concentrated its research efforts.

Aspect-Oriented Software Development has always been related to advanced modularity techniques that promote the separation of concerns principle in the development of complex software systems. From this perspective, it is fundamental to spread the message: AOSD is not only related to the modularization of crosscutting concerns at different abstraction levels, which was the main focus of most research work in the area over the last 15 years. In this sense, since the AOSD 2011 conference, the AOSD international community has motivated the presentation of modularity approaches that propose new and innovative ideas in a special track called "Modularity Visions".

Recent research work has proposed and evaluated these new modularity ideas. Two prominent approaches are Mylin and CIDE. Mylin[9] (Kersten and Murphy, 2005, 2006) is a task management tool that allows developers to work efficiently with their different tasks by providing and automatically organizing the context (for example, different slices of implementation artifacts) that are related to each task. On the other hand, CIDE[10] (Kästner et al., 2008) is a tool that provides support for virtual separation of concerns by allowing the visual annotation of features instead of physically extracting variable features in software product lines implementations. Both tools can be seen as concrete approaches that reflect the community vision on new modularity proposals and that contribute to improve the quality and productive development in software engineering founded on the separation of concerns principle. Senior and young AOSD researchers should take into consideration these new modularity perspectives when developing research work to address most of the open issues presented for different AOSD areas (Section 3).

A necessary and challenging path for the AOSD community to follow is to address practical problems in industry. The research community needs to look over real problems, and apply AOSD solutions to these problems. On the other hand, the AOSD-BR community might have influenced one of the most successful industry cases of AOSD in practice, SpringSource.[11] The OOPSLA 2002 paper from Soares et al. (2002), the most cited paper from the AOSD-BR community, presented a way to use aspects to implement database transactions. Currently, Spring integration framework still uses a similar approach. This is evidence that the Brazilian community needs to better exploit the existing synergy between goals and challenges related to AOSD and problems in industry, thus contributing to increase the number of applications, tools, and users.

[8] Since 2010, the Brazilian SE community has integrated all the software engineering scientific conferences in a unique event called CBSoft – Brazilian Conference on Software: Theory and Practice.

[9] http://www.eclipse.org/mylin.

[10] http://wwwiti.cs.uni-magdeburg.de/iti_db/research/cide.

[11] http://www.springsource.org.

Acknowledgements

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), grants 573964/2008-4 (CNPq) and APQ-1037-1.03/08 (FACEPE).

References

Adachi, E., Batista, T., Kulesza, U., Medeiros, A.L., Chavez, C., Garcia, A., 2009. Variability management in aspect-oriented architecture description languages: an integrated approach. In: Proceedings of the 2009 XXIII Brazilian Symposium on Software Engineering, SBES '09, IEEE Computer Society, Washington, DC, USA, pp. 1–11, ISBN 978-0-7695-3844-0.

Aksit, M., Wakita, K., Bosch, J., Bergmans, L., Yonezawa, A., 1994. Abstracting object interactions using composition filters. In: Guerraoui, R., Nierstrasz, O., Riveill, M. (Eds.), Workshop on Object-Based Distributed Programming at ECOOP'93, Springer-Verlag, pp. 152–184, citeseer.nj.nec.com/aksit94abstracting.html.

Alencar, F., Castro, J., Moreira, A., Araújo, J.a., Silva, C., Ramos, R., Mylopoulos, J., 2008. Integration of aspects with i* models. In: Proc. 8th Int'l Bi Conf. on Agent-oriented Inf. systems IV (AOIS 2006). pp. 183–201. http://portal.acm.org/citation.cfm?id=1787479.1787493. ISBN 3-540-77989-2, 978-3-540-77989-6.

Alencar, F., Castro, J., Lucena, M., Santos, E., Silva, C., Araújo, J.A., Moreira, A., 2010. Towards modular i* models. In: 25th Annual ACM Symp. on Applied Computing (ACM SAC 2010), ACM, New York, NY, USA, pp. 292–297, ISBN 978-1-60558-639-7.

Alexander, R.T., et al., 2004. Towards the Systematic Testing of Aspect-Oriented Programs. Tech. Report CS-04-105, Dept. of Computer Science. Colorado State Univ., Fort Collins, CO, USA.

Alexander, R., 2003. Aspect-oriented programming: the real costs? IEEE Software 20 (6), 90–93.

Alférez, M., Santos, J.P., Moreira, A., Garcia, A., Kulesza, U., Araújo, J., Amaral, V., 2009. Multi-view composition language for software product line requirements. In: Software Language Engineering, Second International Conference, SLE 2009, Denver, CO, USA, October 5–6, 2009, Revised Selected Papers, Springer, pp. 103–122.

Alves Jr., V.P.M., Cole, L., Borba, P., Ramalho, G., 2005. Extracting and evolving mobile games product lines. In: Proc. 9th Int'l Software Product Line Conf. (SPLC 2005), vol. 3714 of LNCS, Springer-Verlag, Rennes, France, pp. 70–81.

Alves, V., Gheyi, R., Massoni, T., Kulesza, U., Borba, P., de Lucena, C.J.P., 2006. Refactoring product lines. In: GPCE, pp. 201–210.

Alves, V., Matos, P., Cole, L., Vasconcelos, A., Borba, P., Ramalho, G., 2007. Extracting and evolving code in product lines with aspect-oriented programming. Transactions on AOSD 4, 117–142.

Alves, V., Calheiros, F., Nepomuceno, V., Menezes, A., Soares, S., Borba, P., 2008. FLiP: managing software product line extraction and reaction with aspects. In: Software Product Lines, 12th Int'l Conf., SPLC, Limerick, Ireland, p. 354.

Alves, M., Pires, P., Delicato, F., Campos, M., 2008a. CrossMDA: a model-driven approach for aspect management. Journal of Universal Computer Science 14 (8), 1314–1343.

AMPLE, 2006. Aspect-Oriented Product Line Engineering. http://www.ample-project.net/

Anastasopoulos, M., Muthig, D., 2004. An evaluation of aspect-oriented programming as a product line implementation technology. In: ICSR, pp. 141–156.

Andrade, R., Ribeiro, M., Gasiunas, V., Satabin, L., Rebêlo, H., Borba, P., 2011. Assessing idioms for implementing features with flexible binding times. In: 15th European Conference on Software Maintenance and Reengineering, CSMR 2011, 1–4 March 2011, Oldenburg, Germany, pp. 231–240.

Andrews, J.H., 2001. Process-algebraic foundations of aspect-oriented programming. In: Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, REFLECTION'01, pp. 187–209.

Antonelli, L., Rossi, G., Leite, J., 2010. Early identification of crosscutting concerns in the domain model guided by states. In: 25th Annual ACM Symp. on Applied Computing (ACM SAC 2010), ACM, pp. 275–280.

AOM, 2011. Aspect-Oriented Modeling Workshop. http://www.aspect-modeling.org/

AOSD Europe, 2002. AOSD-Europe Network of Excellence. http://www.aosd-europe.net/

AOSD, 2002. 1st International Conference on Aspect-Oriented Software Development. http://trese.cs.utwente.nl/aosd2002/

AOSD, 2011. 10th International Conference on Aspect-Oriented Software Development, http://aosd.net/2011

AOSD-BR, 2002. AOSD-BR Discussion List. http://tech.groups.yahoo.com/group/aosd-br/

AOSD-BR-Community, 2011. AOSD-related papers in top SE conferences from 1999 to 2011. http://bit.ly/AOSD-BR-ASSESSMENT-1999-2011

AOSD-BR-Community, 2011. Production of the AOSD-BR Community. http://bit.ly/AOSD-BR_Production

Apel, S., Leich, T., Rosenmüller, M., Saake, G., 2005. FeatureC++: on the symbiosis of feature-oriented and aspect-oriented programming. In: GPCE, pp. 125–140.

Apel, S., Leich, T., Saake, G., 2006. Aspectual mixin layers: aspects and features in concert. In: Proc. of International Conference on Software Engineering (ICSE), pp. 122–131.

Apel, S., Leich, T., Saake, G., 2008. Aspectual Feature Modules. IEEE Transactions on Software Engineering 34 (2), 162–180.

Apel, S., Kästner, C., Größlinger, A., Lengauer, C., 2010. Type safety for feature-oriented product lines. Automated Software Engineering 17 (3), 251–300.

Barbosa, E.A., Batista, T., Garcia, A., Silva, E., 2011. PL-AspectualACME: an aspect-oriented architectural description language for software product lines. In: Proceedings of the 5th European Conference on Software Architecture, ECSA'11, Springer-Verlag, Berlin, Heidelberg, pp. 139–146, ISBN 978-3-642-23797-3, http://dl.acm.org/citation.cfm?id=2041790.2041808

Batista, T., Chavez, C., Garcia, A., Sant'Anna, C., Kulesza, U., Rashid, A., Castor Filho, F., 2006a. Reflections on architectural connection: seven issues on aspects and ADLs. In: Early Aspects 2006 at ICSE 2006, Shangai, China, pp. 3–10.

Batista, T., Chavez, C., Garcia, A., Kulesza, U., Sant'Anna, C., Lucena, C., 2006b. Aspectual connectors: supporting the seamless integration of aspects and ADLs. In: XX Brazilian Symp. on Software Engineering (SBES 2006), Florianopolis, Brazil, pp. 17–32.

Batra, U., Dahiya, D., 2009. Modularization of concerns in a distributed framework: an aspect oriented approach. In: IEEE International Conference on Computer Science and Information Technology (ICCSIT, 2009), pp. 64–68.

Berg, K.V.D., et al., 2005. AOSD Ontology 1.0 Public Ontology of Aspect-Orientation. Tech. Rep. AOSD-Europe Deliverable D9, AOSD-Europe-UT-01. Universiteit Twente, 2005.

Bergmans, L., Aksit, M., 2001. Composing crosscutting concerns using composition filters. Communications of the ACM 44 (10), 51–57.

Bernardo, R., Sales Jr., R., Castor, F., Coelho, R., Cacho, N., Soares, S., 2011. Agile testing of exceptional behavior. In: XXV Brazilian Symp. on Software Engineering (SBES 2011), Sao Paulo, Brazil.

Bertran, I.M., Garcia, A., von Staa, A., 2011. An exploratory study of code smells in evolving aspect-oriented systems. In: Borba, A., Chiba, S. (Eds.), AOSD, ACM, pp. 203–214, ISBN 978-1-4503-0605-8.

Binder, R.V., 1999. Testing Object-Oriented Systems: Models, Patterns and Tools, 1st ed. Addison Wesley, Reading, MA, USA.

Binkley, D., Ceccato, M., Harman, M., Ricca, F., Tonella, P., 2006. Tool-supported refactoring of existing object-oriented code into aspects. IEEE Transactions on Software Engineering 32 (9), 698–717.

Bonifácio, R., Borba, P., 2009. Modeling scenario variability as crosscutting mechanisms. In: 8th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2009), ACM, New York, NY, USA, pp. 125–136, ISBN 978-1-60558-442-3.

Borba, P., et al., 2012. A theory of Software Product Line Refinement. Theoretical Computer Science.

Braga, R., Germano, F.S.R., Pacios, S.F., Masiero, P., 2007. AIPLE-IS: an approach to develop product lines for information systems using aspects. In: I Brazilian Symp. on Components, Architectures and Software Reuse (SBCARS 2007), pp. 17–30.

Braga, R., Pereira, C.A.F., Masiero, P., 2010. Using aspect-oriented programming to promote reuse across domains in software product lines. In: 4th Latin-American Work. on Aspect-Oriented Software Development (LA-WASP 2010), pp. 1–6.

Burrows, R., Ferrari, F.C., Garcia, A., Taïani, F., 2010a. An empirical evaluation of coupling metrics on aspect-oriented programs. In: ICSE Workshop on Emerging Trends in Software Metrics (WETSoM), ACM Press, Cape Town, South Africa, pp. 53–58, ISBN 978-1-60558-976-3.

Burrows, R., Ferrari, F.C., Lemos, O.A.L., Garcia, A., Taïani, F., 2010b. The impact of coupling on the fault-proneness of aspect-oriented programs: an empirical study. In: Proceedings of the 21th International Symposium on Software Reliability Engineering (ISSRE), IEEE Computer Society, San Jose, CA, USA, pp. 329–338, ISBN 978-1-4244-9056-1, ISSN 1071-9458.

Burrows, R., Taïani, F., Garcia, A., Ferrari, F.C., 2011. Reasoning about faults in aspect-oriented programs: a metrics-based evaluation. In: Proceedings of the 19th International Conference on Program Comprehension (ICPC), IEEE Computer Society, Kingston/ON, Canada, pp. 131–140, ISBN 978-1-61284-308-7, ISSN 1063-6897.

Bynens, M., Landuyt, D.V., Truyen, E., Joosen, W., 2010. Reusable aspect-oriented implementations of concurrency patterns and mechanisms. In: 9th Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS '10), AOSD '10, pp. 17–20.

Cacho, N., Sant'Anna, C., Figueiredo, E., Garcia, A., Batista, T., Lucena, C., 2006. Composing design patterns: a scalability study of aspect-oriented programming. In: 5th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2006), ACM, New York, NY, USA, pp. 109–121, ISBN 1-59593-300-X.

Cacho, N., et al., 2006. Improving modularity of reflective middleware with aspect-oriented programming. In: Proc. 6th Int'l Workshop on Software Engineering and Middleware (SEM 2006), ACM, New York, NY, USA, pp. 31–38, ISBN 1-59593-585-1.

Cacho, N., Castor Filho, F., Garcia, A., Figueiredo, E., 2008. EJFlow: taming exceptional control flows in aspect-oriented programming. In: 7th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2008), Brussels, Belgium, pp. 72–83.

Cacho, N., Dantas, F., Garcia, A., Castor, F., 2009. Exception flows made explicit: an exploratory study. In: XXIII Brazilian Symp. on Software Engineering (SBES 2009), Fortaleza, Brazil.

Cafeo, B.B.P., Masiero, P.C., 2011. Contextual integration testing of object-oriented and aspect-oriented programs: a structural approach for Java and AspectJ. In: Proceedings of the 25th Brazilian Symposium on Software Engineering (SBES), IEEE Computer Society, S ao Paulo, SP, Brazil, ISBN 978-1-4577-2187-8, 214-223.

Calheiros, F., Nepomuceno, V., Borba, P., Soares, S., Alves, V., 2007. Product line variability refactoring tool. In: 1st Workshop on Refactoring Tools, WRT 2007 with ECOOP 2007, Berlin, pp. 32–33.

Camargo, V., Masiero, P.C., 2004. Implementação de Variabilidades em Frameworks Orientados a Aspectos desenvolvidos em AspectJ. In: WASP 2004.

Camargo, V., Masiero, P., 2005. Frameworks Orientados a Aspectos. In: XIX Brazilian Symp. on Software Engineering (SBES 2005), pp. 200–216.

Camargo, V., Masiero, P., 2008. A pattern to design crosscutting frameworks. In: 23rd Annual ACM Symp. on Applied Computing (ACM SAC 2008), pp. 759–764.

Cappelli, C., Leite, J., Batista, T., Silva, L., 2009. An aspect-oriented approach to business process modeling. In: Proc. Early Aspects Workshop at AOSD 2009, ACM, pp. 7–12.

Cappelli, C., Santoro, F., Leite, J., Batista, T., Medeiros, A., Romeiro, C., 2010. Reflections on the modularity of business process models: the case for introducing the aspect-oriented paradigm. Business Process Management Journal 16 (4), 662–687.

Cardelli, L., Wegner, P., 1985. On understanding types, data abstraction, and polymorphism. ACM Computing Surveys 17 (4), 471–523, ISSN 0360-0300.

Carneiro, G., Silva, M., Mara, L., Figueiredo, E., Sant'Anna, C., Garcia, A., Mendonça, M., 2010. Identifying code smells with multiple concern views. In: XXIV Brazilian Symp. on Software Engineering (SBES 2010), IEEE Comp. Soc., Washington, DC, USA, pp. 128–137, ISBN 978-0-7695-4273-7.

Castor Filho, F., et al., 2006. exceptions and aspects: the devil is in the details. In: Proc. 14th ACM SIGSOFT FSE, Portland, USA, pp. 152–162.

Castor Filho, F., Garcia, A., Rubira, C., 2007. Extracting error handling to aspects: a cookbook. In: Proc. ICSM 2007, Paris, France, pp. 134–143.

Castor Filho, F., Garcia, A., Rubira, C., 2007. Error handling as an aspect. In: Proc. Workshop on Best Practices in Applying AOSD at AOSD 2007, Vancouver, Canada.

Cazzola, W., Pini, S., Ancona, M., 2006. Design-based pointcuts robustness against software evolution. In: RAM-SE, pp. 35–45.

Ceccato, M., Tonella, P., 2004. Measuring the effects of software aspectization. In: 1st Workshop on Aspect Reverse Engineering.

Chavez, C., Lucena, C., 2003. A theory of aspects for aspect-oriented software development. XVII Brazilian Symp. on Software Engineering (SBES 2003), 130–145.

Chavez, C., Garcia, A., Kulesza, U., Sant'Anna, C., Lucena, C., 2005. Taming heterogeneous aspects with crosscutting interfaces. In: XIX Brazilian Symp. on Software Engineering (SBES 2005), pp. 216–231.

Chavez, C., Garcia, A., Batista, T., 2007. Are architectural aspects style-dependent? In: Workshop on Aspects in Architectural Description at AOSD 2007, pp. 1–4.

Chavez, C., Garcia, A., Batista, T., Oliveira, M., Sant'anna, C., Rashid, A., 2009. Composing architectural aspects based on style semantics. In: 8th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2009), vol. 11, pp. 1–122.

Chavez, C., Kulesza, U., Soares, S., Borba, P., Lucena, C., Masiero, P., Sant'Anna, C., Piveta, E., Ferrari, F., Castor, F., Coelho, R., Silva, L., Alves, V., Mendonca, N., Figueiredo, E., Camargo, V., Silva, C., Pires, P., Batista, T., Cacho, N., von Staa, A., Leite, J., Silveira, F., Lemos, O., Penteado, R., Delicato, F., Braga, R., Valente, M., Ramos, R., Bonifacio, R., Alencar, F., Castro, J., 2011. The AOSD research community in brazil and its crosscutting impact. In: Software Engineering (SBES), 2011 25th Brazilian Symposium on, pp. 72–81.

Chavez, C., 2004. Um Enfoque Baseado em Modelos para o Design Orientado a Aspectos. Ph.D. Thesis. PUC-Rio.

Chidamber, S., Kemerer, C., 1994. A metrics suite for object oriented design. IEEE Transactions on Software Engineering 20 (8), 476–493.

Chitchyan, R., et al., 2005. Survey of Aspect-Oriented Analysis and Design Approaches, Tech. Rep. AOSD-Europe-ULANC-9, AOSD-Europe. http://www.aosd-europe.net/documents/index.htm/analys.pdf

Chitchyan, R., Rashid, A., Rayson, P., Waters, R., 2007. Semantics-based composition for aspect-oriented requirements engineering. In: Barry, B.M., de Moor, O. (Eds.), 6th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2007), vol. 208, ACM, pp. 36–48, ISBN 1-59593-615-7.

Chitchyan, R., et al., 2009. Semantic vs. syntactic compositions in aspect-oriented requirements engineering: an empirical study. In: 8th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2009), ACM, pp. 149–160.

Chung, L., et al., 2000. Non-functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston.

Clarke, S., Walker, R., 2005. Aspect-Oriented Software Development, chap. Generic Aspect-Oriented Design with Theme/UML. Addison-Wesley, pp. 425–458.

Clements, P., Northrop, L.M., 2001. Software Product Lines: Practices and Patterns. Addison-Wesley.

Coelho, R., et al., 2008. Assessing the impact of aspects on exception flows: an exploratory study. In: Proc. ECOOP 2008, Paphos, Cyprus, pp. 207–234.

Coelho, R., Rashid, A., Kulesza, U., von Staa, A., Lucena, C., Noble, J., 2008. Exception handling bug patterns in aspect oriented programs. In: Proc. PLOP 2008, Chicago, USA, pp. 1–19.

Coelho, R., Kulesza, U., Rashid, A., von Staa, A., Lucena, C., 2008. Unveiling and taming liabilities of aspect libraries reuse. In: XXII Brazilian Symp. on Software Engineering (SBES 2008), Campinas, Brazil.

Coelho, R., et al., 2009. On the robustness assessment of aspect-oriented programs. In: Proc. 3rd ACoM Workshop at OOPSLA 2009, Orlando, FL, USA.

Coelho, R., von Staa, A., Kulesza, U., Rashid, A., Lucena, C., 2011. Unveiling and taming liabilities of aspects in the presence of exceptions: a static analysis based approach. Information Sciences 181 (13), 2700–2720.

Cole, L., Borba, P., 2005. Deriving refactorings for AspectJ. In: 4th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2005), Chicago, USA, pp. 123–134.

Cole, L., Piveta, E.K., Sampaio, A., 2004. A RUP based analysis and design with aspects. In: XVIII Brazilian Symp. on Software Engineering (SBES 2004), Brasilia, Brazil.

Cole, L., Borba, P., Mota, A., 2005. Proving aspect-oriented programming laws. In: Proc. FOAL 2005, p. 1.

Costanza, P., Hirschfeld, R., 2005. Language constructs for context-oriented programming: an overview of ContextL. In: Proceedings of the 2005 Symposium on Dynamic languages, DLS '05, ACM, pp. 1–10.

Cugola, G., de Cote, J.E.M., 2005. On introducing location awareness in publish-subscribe middleware. In: Proceedings of the Fourth International Workshop on Distributed Event-Based Systems, IEEE Computer Society, pp. 377–382.

Cunha, C.A., Sobral, J.A.L., Monteiro, M.P., 2006. Towards reusable aspects: the call-back mismatch problem. In: Proceedings of the 5th International Conference on Aspect-oriented Software Development, AOSD '06, ACM, New York, NY, USA, pp. 134–145, ISBN 1-59593-300-X.

Delamare, R., et al., 2009. A test-driven approach to developing pointcut descriptors in AspectJ. In: Proc. 2nd Int'l Conf. on Software Testing, Verification and Validation (ICST), IEEE Comp. Soc., Denver, CO, USA, pp. 376–385, ISBN 978-0-7695-3601-9.

Delamare, R., Baudry, B., Le Traon, Y., 2009b. AjMutator: a tool for the mutation analysis of AspectJ pointcut descriptors. In: Proceedings of the 4th International Workshop on Mutation Analysis (Mutation), IEEE Computer Society, Denver, CO, USA, pp. 200–204.

DeMillo, R.A., et al., 1978. Hints on test data selection: help for the practicing programmer. IEEE Computer 11 (4), 34–43.

Dijkstra, E., 1976. A Discipline of Programming. Prentice-Hall, Englewood Cliffs, New Jersey, EUA, 217 p.

do Prado Leite, J.C.S., Yu, Y., Liu, L., Yu, E.S.K., Mylopoulos, J., 2005. Quality-based software reuse. CAiSE 53, 5–550.

Douence, R., Motelet, O., Südholt, M., 2001. A formal definition of crosscuts. In: Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, REFLECTION'01, pp. 170–186.

Eaddy, M., Zimmermann, T., Sherwood, K.D., Garg, V., Murphy, G.C., Nagappan, N., Aho, A.V., 2008. Do crosscutting concerns cause defects? IEEE Transactions on Software Engineering 34, 497–515, 0098-5589.

Evermann, J., 2007. A meta-level specification and profile for AspectJ in UML. Journal of Object Technology 6 (7), 27–49.

Fayad, M.E., Schmidt, D., 1997. Object-oriented Application Frameworks, Communications of the ACM 40.

Fenton, N.E., Pfleeger, S.L., 1998. Software Metrics: A Rigorous and Practical Approach, 2nd ed. PWS Publishing Co., Boston, MA, USA. ISBN 0534954251.

Fernandes, V., Delicato, F., Pires, P., Kulesza, U., Batista, T., 2009. CrossMDA2: Uma Abordagem Baseada em Modelos para Gerência de Evoluç ao de Pointcuts. In: III Brazilian Symp. on Components, Architectures and Software Reuse (SBCARS 2009), pp. 195–208.

Ferrari, F., Maldonado, J.C., Rashid, A., 2008. Mutation testing for aspect-oriented programs. In: Proc. 1st Int'l Conf. on Software Testing, Verification and Validation (ICST), IEEE Comp. Soc., Lillehammer, Norway, pp. 52–61, ISBN 978-0-7695-3127-4.

Ferrari, F., Nakagawa, E.Y., Rashid, A., Maldonado, J., 2010. Automating the mutation testing of aspect-oriented java programs. In: Proc. 5th ICSE Int'l Workshop on Automation of Software Test (AST), ACM Press, Cape Town, South Africa, pp. 51–58, ISBN 978-1-60558-970-1.

Ferrari, F., Burrows, R., Lemos, O., Garcia, A., Figueiredo, E., Cacho, N., Lopes, F., Temudo, N., Silva, L., Soares, S., Rashid, A., Masiero, P., Batista, T., Maldonado, J.C., 2010. An exploratory study of fault-proneness in evolving Aspect-Oriented Programs. In: 32nd Int'l Conf. on Software Engineering (ICSE 2010), ACM Press, Cape Town – South Africa, pp. 65–74. ISBN 978-1-60558-719-6.

Ferrari, F., Burrows, R., Lemos, O., Garcia, A., Maldonado, J., 2010. Characterising faults in aspect-oriented programs: towards filling the gap between theory and practice. In: XXIV Brazilian Symp. on Software Engineering (SBES 2010), IEEE Comp. Soc., Salvador, BA, Brazil, pp. 50–59, ISBN 978-0-7695-4273-7.

Ferrari, F., 2010. A contribution to the fault-based testing of aspect-oriented software. Ph.D. Thesis. ICMC/USP, S ao Carlos, SP, Brasil.

Figueiredo, E., Cacho, N., Sant'Anna, C., Monteiro, M., Kulesza, U., Garcia, A., Soares, S., Ferrari, F.C., Khan, S.S., Filho, F.C., Dantas, F., 2008a. Evolving software product lines with aspects: an empirical study on design stability. In: 30th Int'l Conf. on Software Engineering (ICSE 2008), pp. 261–270.

Figueiredo, E., Sant'Anna, C., Garcia, A., Bartolomei, T., Cazzola, W., Marchetto, A., 2008b. On the maintainability of aspect-oriented software: a concern-oriented measurement framework. In: Proc. 12th Eur. Conf. on Software Maintenance and Reengineering (CSMR), pp. 183–192.

Figueiredo, E., Whittle, J., Garcia, A., 2009a. Concernmorph: metrics-based detection of crosscutting patterns. In: Proc. ACM SIGSOFT FSE 2009.

Figueiredo, E., Sant'Anna, C., Garcia, A., Lucena, C., 2009b. Applying and evaluating concern-sensitive design heuristics. In: XXIII Brazilian Symp. on Software Engineering (SBES 2009), IEEE Comp. Soc., Washington, DC, USA, pp. 83–93, ISBN 978-0-7695-3844-0.

Figueiredo, E., et al., 2009c. Detecting architecture instabilities with concern traces: an exploratory study. In: Proc. WICSA/ECSA 2009, pp. 261–264.

Figueiredo, E., Silva, B., Sant'Anna, C., Garcia, A., Whittle, J., Nunes, D., 2009d. Crosscutting patterns and design stability: an exploratory analysis. In: 17th Int'l Conf. on Program Comprehension (ICPC), vol. 13, pp. 8–147.

Figueroa, I., Tanter, E., 2011. A semantics for execution levels with exceptions. In: Proceedings of the 10th Workshop on Foundations of Aspect Languages.

Filman, R., Elrad, T., Clarke, S., Akşit, M. (Eds.), 2005. Aspect-Oriented Software Development. Addison-Wesley, Boston. ISBN 0-321-21976-7.

Fowler, M., Beck, K., Brant, J., Opdyke, W.F., Roberts, D., 1999. Refactoring – Improving the Design of Existing Code, Object Technologies Series. Addison-Wesley.

Gamma, E., et al., 1995. Design patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Garcia, V.C., Piveta, E.K., Lucredio, D., Alvaro, A., Almeida, E., Prado, A., Zancanella, L., 2004. Manipulating crosscutting concerns. In: 4th Latin American Conference on Patterns Languages of Programming (SugarLoafPLoP), Porto das Dunas, Brazil.

Garcia, A., Sant'Anna, C., Figueiredo, E., Kulesza, U., Lucena, C., von Staa, A., 2005. Modularizing design patterns with aspects: a quantitative study. In: 4th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2005), ACM, New York, NY, USA, 3–14, 2005. ISBN 1-59593-042-6.

Garcia, A., Chavez, C., Batista, T., Sant'anna, C., Kulesza, U., Rashid, A., Lucena, C., 2006. On the modular representation of architectural aspects. In: Proc. EWSA 2006, Nantes, France.

Garcia, A., Sant'Anna, C., Figueiredo, E., Kulesza, U., Lucena, C., von Staa, A., 2006b. Modularizing design patterns with aspects: a quantitative study. In: Transactions on AOSD I, pp. 36–74.

Garcia, A., 2004. From Objects to Agents: An Aspect-Oriented Approach. Ph.D. Thesis. PUC-Rio, Brazil.

Ghezzi, C., Sharifloo, A.M., 2011. Verifying non-functional properties of software product lines: towards an efficient approach using parametric model checking. In: SPLC, pp. 170–174.

Goodenough, J.B., 1975. Exception handling: issues and a proposed notation. Communications of the ACM 18 (12), 683–696.

Greenwood, P., et al., 2007. On the contributions of an end-to-end AOSD testbed. In: Early Aspects: Workshop in AO Requirements Engineering and Architecture Design, IEEE Comp. Soc, p. 8, ISBN 0-7695-2957-7.

Greenwood, P., Bartolomei, T., Figueiredo, E., Garcia, A., Cacho, N., Sant'Anna, C., Borba, P., Kulesza, U., Rashid, A.,2007. On the impact of aspectual decompositions on design stability: an empirical study. In: Proc. ECOOP 2007. Springer-Verlag, pp. 176–200.

Grundy, J., 1999. Aspect-oriented requirements engineering for component-based software systems. In: Proc. IEEE Int'l Symp. on Req. Engineering, IEEE, pp. 84–91.

Gybels, K., Brichau, J., 2003. Arranging language features for more robust pattern-based crosscuts. In: 2nd Int'l Conf. on Aspect-Oriented Software Development (AOSD 2003), ACM, New York, NY, USA, pp. 60–69, ISBN 1-58113-660-9.

Hanenberg, S., Schmidmeir, S., 2003. AspectJ Idioms for aspect-oriented software construction. In: Proc. EuroPLoP 2003.

Hanenberg, S., Oberschulte, C., Unland, R., 2003. Refactoring of aspect-oriented software. In: 4th Net.ObjectDays Conference, Erfurt, Germany.

Harrison, W., Ossher, H., 1993. Subject-oriented programming (a critique of pure objects). In: 7th Conf. on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA93), pp. 411–428.

Heidenreich, F., Sánchez, P., Santos, J.P., Zschaler, S., Alférez, M., Araújo, J., Fuentes, L., Kulesza, U., Moreira, A., Rashid, A., 2010. Relating Feature Models to Other Models of a Software Product Line – A Comparative Study of FeatureMapper and VML*, Transactions on Aspect-Oriented Software Development VII – A Common Case Study for Aspect-Oriented Modeling, vol. 7. Springer Verlag, LNCS, pp. 69–114.

Hirschfeld, R., Igarashi, A., Masuhara, H., 2011. ContextFJ: a minimal core calculus for context-oriented programming. In: Proceedings of the 10th FOAL Workshop, ACM, pp. 19–23.

Hoffman, K.J., Eugster, P., 2008. Towards reusable components with aspects: an empirical study on modularity and obliviousness. In: Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, pp. 91–100.

Iborra, J., Pastor, O., Pelechano, V., 2006. Dealing with crosscutting concerns in a model based software production method. In: Proceedings of the 2006 International Workshop on Early Aspects at ICSE, EA'06, ACM, New York, NY, USA, pp. 27–34, ISBN 1-59593-405-7.

Iwamoto, M., Zhao, J., 2003. Refactoring aspect-oriented programs. In: 5th AOSD Modeling with UML Workshop (AOM), San Francisco, USA.

Kästner, C., Apel, S., Kuhlemann, M., 2008. Granularity in software product lines. In: ICSE, pp. 311–320.

Kamina, T., Aotani, T., Masuhara, H., 2011. EventCJ: a context-oriented programming language with declarative event-based context transition. In: Proceedings of the 10th AOSD, ACM, pp. 253–264.

Kande, M.M., Strohmeier, M.M.K.A., 2000. On the role of multi-dimensional separation of concerns in software architecture. In: Proceedings OOPSLA Workshop on Advanced Separation of Concerns.

Katara, M., Katz, S., 2003. Architectural views of aspects. In: Proceedings of the 2nd International Conference on Aspect-oriented Software Development, AOSD'03, ACM, New York, NY, USA, pp. 1–10, ISBN 1-58113-660-9.

Kellens, A., et al., 2006. Managing the evolution of aspect-oriented software with model-based pointcuts. In: Proc. ECOOP 2006, Nantes, France, pp. 501–525.

Kersten, M., Murphy, G.C., 2005. Mylar: a degree-of-interest model for IDEs. In: Proceedings of the 4th International Conference on Aspect-Oriented Software Development, AOSD 2005, Chicago, IL, USA, March 14–18, 2005, pp. 159–168.

Kersten, M., Murphy, G.C., 2006. Using task context to improve programmer productivity. In: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006, Portland, OR, USA, November 5–11, 2006, pp. 1–11.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., Irwin, J., 1997. Aspect-oriented programming. In: Proc. ECOOP 1997, LNCS 1241, pp. 220–242.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W., 2001. Getting started with AspectJ. Communications of the ACM 44 (10), 59–65.

Koppen, C., Stoerzer, M., 2004. PCDiff: attacking the fragile pointcut problem. In: Gybels, K., Hanenberg, S., Herrmann, S., Wloka, J. (Eds.), European Interactive Workshop on Aspects in Software (EIWAS).

Krechetov, I., Tekinerdogan, B., Garcia, A., Chavez, C., Kulesza, U., 2006. Towards an integrated aspect-oriented modeling approach for software architecture design. In: Proc. AOM 2006 at AOSD 2006, Bonn, Germany.

Krueger, C.W., 2001. Easing the transition to software mass customization. In: Software Product-Family Engineering, 4th International Workshop, PFE 2001, Bilbao, Spain, October 3–5, 2001, Revised Papers, Lecture Notes in Computer Science, pp. 282–293.

Kulesza, U., Garcia, A., Bleasby, F., Lucena, C., 2005. Instantiating and customizing product line architectures using aspects and crosscutting feature models. In: Workshop on Early Aspects at OOPSLA 2005, San Diego, USA.

Kulesza, U., Sant'Anna, C., Garcia, A., Coelho, R., von Staa, A., Lucena, C., 2006a. Quantifying the effects of aspect-oriented programming: a maintenance study. In: Proc. 22nd IEEE Int'l Conf. on Software Maintenance, IEEE Comp. Soc., Washington, DC, USA, pp. 223–233, ISBN 0-7695-2354-4.

Kulesza, U., Coelho, R., Alves, V., Neto, A.C., Garcia, A., Lucena, C., Staa, A.V., Borba, P., 2006b. Implementing framework crosscutting extensions with EJPs and AspectJ. In: XX Brazilian Symp. on Software Engineering (SBES 2006).

Kulesza, U., de Lucena, C.J.P., Alencar, P.S.C., Garcia, A., 2006c. Customizing aspect-oriented variabilities using generative techniques. In: Proceedings of the Eighteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2006), San Francisco, CA, USA, July 5–7, 2006, pp. 17–22.

Kulesza, U., Alves, V., Garcia, A., Lucena, C., Borba, P., 2006d. Improving extensibility of object-oriented frameworks with aspect-oriented programming. In: Proc. 9th Int'l Conf. on Software Reuse (ICSR 2006), pp. 231–245.

Kulesza, U., Alves, V., Garcia, A., Neto, A., Cirilo, E., de Lucena, C., Borba, P., 2007. Mapping features to aspects: a model-based generative approach. In: Moreira, A., Grundy, J. (Eds.), Early Aspects: Current Challenges and Future Directions, vol. 4765 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, pp.155–174. ISBN 978-3-540-76810-4.

Lämmel, R., 2002. A semantical approach to method-call interception. In: 1st Int'l Conf. on Aspect-Oriented Software Development (AOSD 2002), pp. 41–55.

LatinAOSD, 2007. The Latin American Research Network on AOSD. http://wiki.dcc.ufba.br/LatinAOSD/

Lee, K., Kang, K.C., Kim, M., Park, S., 2006. Combining feature-oriented analysis and aspect-oriented programming for product line asset development. In: SPLC, pp. 103–112.

Lemos, O., Masiero, P., 2008. Integration testing of aspect-oriented programs: a structural pointcut-based approach. In: XXII Brazilian Symp. on Software Engineering (SBES 2008), pp. 49–64.

Lemos, O., Masiero, P., 2011. A pointcut-based coverage analysis approach for aspect-oriented programs. Information Sciences 181 (13), 2721–2746, ISSN 0020-0255.

Lemos, O., Vincenzi, A., Maldonado, J.C., Masiero, P.C., 2004. Teste de Unidade de Programas Orientados a Aspectos. In: XVIII Brazilian Symp. on Software Engineering (SBES 2004), Brasília/DF, Brasil, pp. 55–70.

Lemos, O., Ferrari, F., Masiero, P., Lopes, C., 2006. Testing aspect-oriented programming pointcut descriptors. In: Proc. 2nd Workshop on Testing AO Programs (WTAOP) at ISSTA, ACM Press, Portland, USA, pp. 33–38, ISBN 1-59593-415-4/06/0007.

Lemos, O., Vincenzi, A., Maldonado, J., Masiero, P., 2007. Control and data flow structural testing criteria for aspect-oriented programs. Journal of Systems and Software 80 (6), 862–882, ISSN 0164-1212.

Lemos, O., Franchin, I., Masiero, P., 2009. Integration testing of object-oriented and aspect-oriented programs: a structural pairwise approach for java. Science of Computer Programming 74, 861–878, ISSN 0167-6423.

Lieberherr, K., Orleans, D., Ovlinger, J., 2001. Aspect-oriented programming with adaptive methods. Communications of the ACM 44 (10), 39–41.

Lieberherr, K., 1996. Adaptive Object-oriented Software: The Demeter Method with Propagation Patterns. PWS Publishing Company, Boston. ISBN 0-534-94602-X.

Lippert, M., Lopes, C.V., 2000. A Study on exception detection and handling using aspect-oriented programming. In: 22nd Int'l Conf. on Software Engineering (ICSE 2000), Limerick, Ireland, pp. 418–427.

Lobato, C., Garcia, A., Kulesza, U., Staa, A.V., Lucena, C., 2008. Evolving and composing frameworks with aspects: the mobigrid case. In: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), IEEE Computer Society, Washington, DC, USA, pp. 53–62, http://dl.acm.org/citation.cfm?id=1343591.1343618, ISBN 978-0-7695-3091-8.

Lopes, C.V., 2005. AOP: a historical perspective (what's in a name?). In: Filman et al. (2005), pp. 97–122.

Loughran, N., Rashid, A., Zhang, W., Jarzabek, S., 2004. Supporting product line evolution with framed aspects. In: Proc. of Workshop on Aspects, Components and Patterns for Infrastructure Software (held with AOSD 2004).

Machado, I., Bonifácio, R., Alves, V., Turnes, L., Machado, G., 2011. Managing variability in business processes: an aspect-oriented approach. In: Proc. Early Aspects Workshop at AOSD 2011, ACM, New York, NY, USA, pp. 25–30, ISBN 978-1-4503-0645-4.

Masuhara, H., Kiczales, G., 2003. Modeling Crosscutting in Aspect-Oriented Mechanisms. In: Cardelli, L. (Ed.), ECOOP, vol. 2743 of Lecture Notes in Computer Science, Springer, pp. 2–28. ISBN 3-540-40531-3.

Medeiros, A., Silva, L., Batista, T., Minora, L., 2007. Requisitos e Arquitetura de Software Orientada a Aspectos: Uma Integraç ao Sinérgica. In: XXI Brazilian Symp. on Software Engineering (SBES 2007).

Medeiros, A., et al., 2007. MARISA-Uma Ferramenta para Mapeamento bidirecional de Modelos Orientados a Aspectos: Requisitos e Arquitetura de Software. In: 1st Latin-American Work. on Aspect-Oriented Software Development (LA-WASP 2007), pp. 55–66.

Mellor, S., et al., 2004. MDA Distilled. Addison-Wesley, Redwood City, CA, USA. ISBN 0201788918.

Mens, T., Tourwe, T., 2004. A survey of software refactoring. IEEE Transactions on Software Engineering 30 (2), 126–139.

Mens, K., 2002. Architectural aspects. In: Proceedings of AOSD 2002 Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design.

Mezini, M., Ostermann, K., 2004. Variability management with feature-oriented programming and aspects. In: Proc. 12th ACM SIGSOFT FSE, pp. 127–136.

Molesini, A., Garcia, A., Batista, T., Chavez, C., 2010. Stability assessment of aspect-oriented software architectures: a quantitative study. Journal of Systems and Software.

Monteiro, M.P., Fernandes, J.M., 2004. Object-to-aspect refactorings for feature extraction. In: International Conference on Aspect-Oriented Software Development (AOSD), ACM Press, Lancaster, UK.

Monteiro, M.P., Fernandes, J.M., 2005. Towards a catalog of aspect-oriented refactorings. In: 4th International Conference on Aspect-Oriented Software Development (AOSD), ACM Press, Chicago, USA, pp. 111–122.

Monteiro, M.P., Fernandes, J.M., 2006. Towards a catalogue of refactorings and code smells for aspectJ. Transactions on Aspect-Oriented Software Development I (2880), 214–258.

Monteiro, C., et al., 2007. The iAspectPlugin to automatize the identification of crosscutting concerns on i* models. In: 1st Latin-American Work. on Aspect-Oriented Software Development (LA-WASP 2007).

Moreira, A., et al., 2002. Crosscutting quality attributes for requirements engineering. In: Proc. SEKE 2002, pp. 167–174.

Moreira, A., Rashid, A., Araujo, J., 2005. Multi-dimensional separation of concerns in requirements engineering. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, IEEE Computer Society, Washington, DC, USA, pp. 285–296, ISBN 0-7695-2425-7.

Morell, L.J., 1990. A theory of fault-based testing. IEEE Transactions on Software Engineering 16 (8), 844–857, ISSN 0098-5589.

Mortensen, M., Ghosh, S., 2006. Using aspects in object-oriented frameworks. In: 5th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2006).

Neves, L., Teixeira, L., Sena, D., Alves, V., Kulesza, U., Borba, P., 2011. Investigating the safe evolution of software product lines. In: GPCE.

e Oliveira, M.S., 2007. Modelagem de Arquitetura de Software Orientada a Aspectos em UML 2.0. Master's thesis. IME/RJ.

Opdyke, W., 1992. Refactoring Object-Oriented Frameworks. Ph.D. Thesis. University of Illinois at Urbana Champaign.

Ossher, H., Tarr, P., 2001. Using multi-dimensional separation of concerns to (re)shape evolving software. Communications of the ACM 44 (10), 43–50.

Pacios, S.F., Masiero, P., Braga, R., 2006. Guidelines for using aspects to evolve product lines. In: 3rd Brazilian Work. on Aspect-Oriented Software Development (WASP 2006), pp. 111–120.

Parnas, D., 1972. On the criteria to be used in decomposing systems into modules. Communications of the ACM 15 (12), 1053–1058.

Pereira, C., Braga, R., Masiero, P., 2008. Captor-AO: an application generator supported by aspect-oriented programming (in Portuguese only). In: Proc. Tools Session at the SBCARS 2008, ISBN 978-85-7430-796-1, pp. 1–8.

Perry, D.E., Wolf, A.L., 1992. Foundations for the study of software architecture, SIGSOFT Software Engineering Notes 17, 40–52. ISSN:0163-5948.

Pinto, M., Fuentes, L., 2007. AO-ADL: an ADL for describing aspect-oriented architectures. In: Proc. 10th Int'l Conf. on Early Aspects: Current Challenges and Future Directions, Springer-Verlag, Berlin, Heidelberg, pp. 94–114. http://portal.acm.org/citation.cfm?id=1783274.1783283, ISBN 3-540-76810-6, 978-3-540-76810-4.

Pinto, M., et al., 2002. Separation of coordination in a dynamic aspect-oriented framework. In: 1st Int'l Conf. on Aspect-Oriented Software Development (AOSD 2002).

Pinto, M., Fuentes, L., Valenzuela, J.A., Pires, P.F., Delicato, F.C., Marinho, E., 2009. On the need of architectural patterns in AOSD for software evolution. In: Proc. WICSA/ECSA 2009, pp. 245–248.

Piveta, E., Hecht, M., Pimenta, M., Price, R.T., 2005. Bad smells em sistemas orientados a aspectos. In: XIX Brazilian Symp. on Software Engineering (SBES 2005), Uberlândia, Brazil.

Piveta, E., Hecht, M., Pimenta, M., Price, R., 2006. Detecting bad smells in AspectJ. Journal of Universal Computer Science 12 (7), 811–827.

Piveta, E., et al., 2007. Avoiding bad smells in aspect-oriented software. In: 19th Int'l Conf. on Software Engineering and Knowledge Engineering (SEKE), Boston, USA.

Piveta, E., et al., 2008. Searching for opportunities of refactoring sequences: reducing the search space. In: 32nd IEEE COMPSAC, IEEE Press, Turku, Finland.

Piveta, E., Moreira, A., Pimenta, M., Araujo, J., Guerreiro, P., Price, R.T., 2008. Ranking refactoring patterns with the analytic hierarchy process. In: 10th Int'l Conf. on Enterprise Information Systems (ICEIS), Barcelona, Spain.

Piveta, E.K., Moreira, A., Pimenta, M.S., Araujo, J., Guerreiro, P., Price, R.T., 2012. An empirical study of aspect-oriented metrics, Science of Computer Programming. ISSN 0167-6423.

Piveta, E., 2009. Improving the Search for Refactoring Opportunities on Aspect-Oriented and Object-Oriented Software. Ph.D. Thesis. UFRGS.

Prehofer, C., 1997. Feature-oriented programming: a fresh look at objects. In: ECOOP, pp. 419–443.

Ramos, R., et al., 2006. Avaliaç ao da Qualidade de Documentos de Requisitos Orientado a Aspectos. In: Proc. X Iberoamericano de Ingenieria de Req. y Ambientes de Software (IDEAS 06).

Ramos, R., Castro, J., ao Araújo, J., Moreira, A., Alencar, F., 2008. Refatoraç ao para Documento de Requisitos: Uma Abordagem Aspectual. IEEE Latin America Transactions 6(3).

Rashid, A., Chitchyan, R., 2003. Persistence as an aspect. In: 1st Int'l Conf. on Aspect-Oriented Software Development (AOSD 2002), pp. 759–764.

Rashid, A., Moreira, A., 2006. Domain models are NOT aspect free. In: Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems, MoDELS'06, Springer-Verlag, Berlin, Heidelberg, pp. 155–169, ISBN 3-540-45772-0, 978-3-540-45772-5.

Rashid, A., et al., 2002. Early aspects: a model for aspect-oriented requirements engineering. In: Proc. 10th Anniv. IEEE Joint Int'l Conf. on Req. Engineering, IEEE Comp. Soc.

Reina, A.M., Torres, J., 2005. Weaving AspectJ by means of transformations. In: Proc. 1st Workshop on Models and Aspects – Handling Crosscutting Concerns in MDSD at ECOOP 2005.

Rojas, K.A.L., et al., 2009. Derivación de casos de uso con aspectos a partir de modelos organizacionales i*. In: Memorias de la XII Conferencia Iberoamericana de Software Engineering (CIbSE 2009), pp. 253–258.

Rumbaugh, J., Jacobson, I., Booch, G., 2004. Unified Modeling Language Reference Manual. 2nd ed. Pearson Higher Education. ISBN 0321245628.

Saaty, T.L., 1990. How to make a decision: the analytic hierarchy process. European Journal of Operational Research 48 (1), 9–26.

Saaty, T.L., 2003. Decision-making with the AHP: why is the principal eigenvector necessary? European Journal of Operational Research 145 (1), 85–91.

Sampaio, A., et al., 2005. EA-Miner: a tool for automating aspect-oriented requirements identification. In: Proc. 20th IEEE/ACM Int'l Conf. on ASE, ACM, pp. 352–355.

Sande, M., Choren, R., Chavez, C., 2006. Mapping AspectualACME into UML2.0. In: AOM 2006 at MODELS 2006, Genova, Italy.

Sant'anna, C., Garcia, A., Chavez, C., Lucena, C., von Staa, A.V., 2003. On the reuse and maintenance of aspect-oriented software: an assessment framework. In: XVII Brazilian Symp. on Software Engineering (SBES 2003), pp. 19–34.

Sant'Anna, C., Garcia, A., Kulesza, U., Lucena, C., von Staa, A., 2004. Design patterns as aspects: a quantitative assessment. In: XVIII Brazilian Symp. on Software Engineering (SBES 2004), Brasilia, Brazil.

Sant'Anna, C., Figueiredo, E., Garcia, A., Lucena, C., 2007. On the modularity of software architectures: a concern-driven measurement framework. In: Proc. ECSA 2007, pp. 207–224.

Sant'Anna, C., et al., 2008. On the modularity assessment of aspect-oriented multiagent architectures: a quantitative study. IJAOSE 2, 34–61, 1746-1375.

Sant'Anna, C., 2008. On the Modularity of Aspect-Oriented Design: A Concern-Driven Measurement Approach. Ph.D. Thesis. PUC-Rio.

Santos, A.L., Lopes, A., Koskimies, K., 2007. Framework specialization aspects. In: Proceedings of the 6th International Conference on Aspect-Oriented Software Development, AOSD '07, ACM, New York, NY, USA, pp. 14–24, ISBN 1-59593-615-7.

Santos, L., Silva, L., Batista, T., 2011. On the integration of the feature model and PL-AOVGraph. In: Proc. Early Aspects Workshop at AOSD 2011, ACM, pp. 31–36.

Saraiva, D., Pereira, L., Batista, T., Delicato, F.C., Pires, P.F., 2010. Architecting a model-driven aspect-oriented product line for a digital TV middleware: a refactoring experience. In: Proceedings of the 4th European conference on Software architecture, ECSA'10, Springer-Verlag, Berlin, Heidelberg, pp. 166–181, ISBN 3-642-15113-2, 978-3-642-15113-2, http://dl.acm.org/citation.cfm?id=1887899.1887915

Schmidt, D., 2006. Guest Editor's introduction: model-driven engineering. IEEE Computer 39 (2), 25–31.

Siegmund, N., Rosenmüller, M., Kästner, C., Giarrusso, P.G., Apel, S., Kolesnikov, S.S., 2011. Scalable prediction of non-functional properties in software product lines. In: SPLC, pp. 160–169.

Silva, L., Leite, J., 2005a. Uma Linguagem de Modelagem de Requisitos Orientada a Aspectos. In: Proc. WER 2005 – Workshop on Req. Engineering, pp. 13–25.

Silva, L., Leite, J., 2005b. Integraç ao de características transversais durante a modelagem de requisitos. In: XIX Brazilian Symp. on Software Engineering (SBES 2005), pp. 26–39.

Silva, L., et al., 2007. On the symbiosis of aspect-oriented requirements and architectural descriptions. In: Proc. 10th Int'l Conf. on Early Aspects: Current Challenges and Future Directions, Springer-Verlag, pp. 75–93.

Silva, C., et al., 2008. Tailoring an aspectual goal-oriented approach to model features. In: SEKE 2008, pp. 472–477.

Silva, B., Figueiredo, E., Garcia, A., Nunes, D., 2009. On the support and application of macro-refactorings for crosscutting concerns. In: III Brazilian Symp. on Components, Architectures and Software Reuse (SBCARS 2009).

Silva, C., Castro, J., Araújo, J., Moreira, A., Tedesco, P., Mylopoulos, J., 2009b. Advanced separation of concerns in agent-oriented design patterns. IJAOSE 3 (2/3), 306–327.

Silva, B., Sant'Anna, C., Chavez, C., 2011. Concern-based cohesion as change proneness indicator: an initial empirical study. In: Proc. 2nd Int'l workshop on Emerging Trends in Software Metrics (WETSoM 2011), ACM, New York, NY, USA, pp. 52–58, ISBN 978-1-4503-0593-8.

Silveira, F.F., et al., 2005. The testing activity on the aspect-oriented paradigm. In: Proc. 1st Workshop on Testing Aspect Oriented Programs at AOSD 2005, Chicago, IL, USA.

Silveira, F., 2007. METEORA: Um Método de Teste Baseado em Estados para Software de Aplicaç ao Orientado a Aspectos. Ph.D. Thesis. ITA.

Simmonds, D., Solberg, A., Reddy, R., France, R., Ghosh, S., 2005. An aspect oriented model driven framework, computing, pp. 119–130. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1540673

Soares, S., Laureano, E., Borba, P.,2002. Implementing distribution and persistence aspects with AspectJ. In: Proc. OOPSLA 2002. ACM Press, Seattle, USA, pp. 174–190.

Soares, S., Calheiros, F., Nepomuceno, V., Menezes, A., Borba, P., Alves, V., 2008. Supporting software product lines development: FLiP – product line derivation tool. In: OOPSLA Companion, pp. 737–738.

Soares, S., 2004. An Aspect-Oriented Implementation Method. Ph.D. Thesis. Informatics Center, Federal University of Pernambuco – CIn-UFPE – Brazil, 2004.

Solberg, A., 2005. Using aspect oriented techniques to support separation of concerns in model driven development. In: Proc. SAC 2005, pp. 121–126.

Sousa, G., Silva, I., Castro, J., 2003. Adapting the NFR framework to aspect-oriented requirements engineering. In: XVII Brazilian Symp. on Software Engineering (SBES 2003), pp. 83–98.

Srivisut, K., Muenchaisri, P., 2007. Defining and detecting bad smells of aspect-oriented software. In: IEEE International Computer Software and Applications Conference (COMPSAC), pp. 65–70.

Stein, D., Hanenberg, S., Unland, R., 2006. Expressing different conceptual models of join point selections in aspect-oriented design. In: 5th Int'l Conf. on Aspect-Oriented Software Development (AOSD 2006), AOSD '06, ACM, New York, NY, USA, pp. 15–26.

Sullivan, K., et al., 2005. Information hiding interfaces for aspect-oriented design. In: Proc. 10th ESEC/FSE-13, ACM, New York, NY, USA, pp. 166–175, ISBN 1-59593-014-0.

Sztajnberg, A., et al., 1999a. Configurando protocolos de intera??o na abordagem R-RIO. In: XIII Brazilian Symp. on Software Engineering (SBES 1999).

Sztajnberg, A., et al., 1999b. Towards integrating meta-level programming and configuration programming. In: XIII Brazilian Symp. on Software Engineering (SBES 1999), pp. 309–323.

Tanter, E., 2010. Execution levels for aspect-oriented programming. In: Proceedings of the 9th AOSD.

TAOSD, 2006. Transactions on Aspect-Oriented Software Development. http://www.springer.com/computer/lncs?SGWID=0-164-2-109318-0

Tarr, P., Ossher, H., Harrison Jr., W.S.S., 1999. N degrees of separation: multi-dimensional separation of concerns. In: 21st Int'l Conf on Software Engineering (ICSE'99), pp. 107–119.

Taveira, J.C., et al., 2009. Assessing intra-application exception handling reuse with aspects. In: XXIII Brazilian Symp. on Software Engineering (SBES 2009), Fortaleza, Brazil.

Teixeira, L., Borba, P., Gheyi, R., 2011. Safe composition of configuration knowledge-based software product lines. In: SBES, pp. 263–272.

Uetanabara, J., Chavez, C., Camargo, V., 2009. UML-AOF: a profile for modeling aspect-oriented frameworks. In: Proc. AOM at AOSD 2009.

Uetanabara, J., Penteado, R., Camargo, V., 2010. An overview and an empirical evaluation of UML-AOF: a UML profile for aspect-oriented frameworks. In: 25th Annual ACM Symp. on Applied Computing (ACM SAC 2010), pp. 1–6.

Valente, M.T., Malta, M.N., Domingues, S., 2009. Guidelines for enabling the extraction of aspects from existing object-oriented code. Journal of Object Technology 8 (3), 101–119.

Valente, M.T., Couto, C., Faria, J., Soares, S., 2010. On the benefits of quantification in AspectJ systems. Journal of the Brazilian Computer Society 16 (2), 133–146.

Valente, M.T., Borges, V., Passos, L., 2012. A semi-automatic approach for extracting software product lines. IEEE Transactions on Software Engineering 38 (4), 737–754.

van Dooren, M., Steegmans, E., 2005. Combining the robustness of checked exceptions with the flexibility of unchecked exceptions using anchored exception declarations. In: Proceedings of OOPSLA'2005, pp. 455–471.

Venners, B., Eckel, B., 2003. The Trouble with Checked Exceptions: A Conversation with Anders Hejlsberg, Part II. Available at http://www.artima.com/intv/handcuffs.html. Last visited in January 9th 2012.

Wampler, D., 2003. The Role of Aspect-Oriented Programming in OMG's Model-Driven Architecture.

Wand, M., Kiczales, G., Dutchyn, C., 2004. A semantics for advice and dynamic join points in aspect-oriented programming. ACM Transactions on Programming Languages and Systems 26 (5), 890–910.

WASP, 2004. 1st Brazilian Workshop on AOSD. http://tech.groups.yahoo.com/group/aosd-br/

WASP, 2005. 2nd Brazilian Workshop on AOSD. www.labes.icmc.usp.br/wasp2005

WASP, 2006. 3rd Brazilian Workshop on AOSD. http://www.les.inf.puc-rio.br/wasp2006

Wimmer, M., Schauerhuber, A., Kappel, G., Retschitzegger, W., Schwinger, W., Kapsammer, E., 2011. A survey on UML-based aspect-oriented design modeling. ACM Computing Surveys 43 (4), 28.

Xu, D., Xu, W., Nygard, K., 2005. A state-based approach to testing aspect-oriented programs. In: Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE), Taiwan.

Yu, Y., Leite, J., Mylopoulos, J., 2004. From goals to aspects: discovering aspects from requirements goal models. In: Proc. 12th IEEE Int'l Requirements Engineering Conf, pp. 38–47.

Yu, E.S.K., 1997. Towards modeling and reasoning support for early-phase requirements engineering. In: Proc. of the 3rd IEEE International Symposium on Requirements Engineering, RE '97, IEEE Computer Society, Washington, DC, USA. http://dl.acm.org/citation.cfm?id=827255.827807. ISBN 0-8186-7740-6.

Zhao, J., Xu, B., 2004. Measuring aspect cohesion. In: Wermelinger, M., Margaria, T. (Eds.), 7th International Conference Fundamental Approaches to Software Engineering (FASE 2004), vol. 2984 of Lecture Notes in Computer Science, Springer, pp. 54-68. ISBN 3-540-21305-8.

Zhao, J., 2002. Tool support for unit testing of aspect-oriented software. In: OOPSLA 2002 Workshop on Tools for AOSD, Seattle, WA, USA.

Zhao, J., 2003. Data-flow-based unit testing of aspect-oriented programs. In: Proc. 27th Annual IEEE Int'l Computer Software and Applications Conf, IEEE Computer Society, Dallas/Texas, USA, pp. 188–197.

Zhao, J., 2004. Measuring coupling in aspect-oriented systems. In: Proceedings of the 10th International Software Metrics Symposium, (Late Breaking Paper).

Zschaler, S., Sánchez, P., Santos, J.P., Alférez, M., Rashid, A., Fuentes, L., Moreira, A., Araújo, J., Kulesza, U., 2009. VML* – a family of languages for variability management in software product lines. In: Software Language Engineering, Second International Conference, SLE 2009, Denver, CO, USA, October 5–6, 2009, Revised Selected Papers, Springer, pp. 82–102.

Uirá Kulesza is an Associate Professor at the Federal University of Rio Grande do Norte (UFRN), Brazil. He obtained his PhD in Computer Science at PUC-Rio – Brazil (2007). His main research interests include: software product lines, software evolution, and generative development. He has co-authored over 100 referred papers in journals, conferences, and book chapters. He worked as a post-doc researcher member of the AMPLE project (2007–2009) – Aspect- Oriented Model-Driven Product Line Engineering (www.ample-project.net) at the New University of Lisbon, Portugal.

Sérgio Soares received his PhD in Computer Science at UFPE – Brazil (2004). He is an Associate Professor at the Federal University of Pernambuco (UFPE) and Executive Coordinator of INES, the National Institute of Science and Technology for Software Engineering. His main research interests include Empirical Software Engineering, Software Product Lines, Software Evolution, Software Modularity, and Object-Oriented Programming.

Christina von Flach Garcia Chavez received her PhD in Computer Science at PUC-Rio – Brazil (2004). She is an Associate Professor at the Federal University of Bahia (UFBA) - Brazil. Her main research interests include Software Evolution and Software Design, with emphasis on architecture, principles, patterns and practices, digital archeology of software and Software Engineering Education.

Fernando Castor is an Assistant Professor at the Federal University of Pernambuco (UFPE) – Brazil. He holds a PhD in Computer Science from UNICAMP – Brazil. His is interested in developing techniques, tools, and methods to ease the construction and maintenance of large-scale dependable systems. His main research areas are exception handling, fault tolerance, concurrent programming, and advanced modularization techniques.

Paulo Borba is Professor of Software Development at the Informatics Center of the Federal University of Pernambuco, Brazil, where he leads the Software Productivity Group. His main research interests are in the following topics and their integration: software modularity, software product lines, and refactoring.

Carlos Lucena is a Full Professor of Computer Science at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) since 1982 and an Adjunct Professor of Computer Science and a Senior Research Associate of the Computer Systems Group at the University of Waterloo, which he has visited on a regular basis since 1975. He completed his undergraduate studies in Economics and Mathematics between 1962 and 1965 at PUC-Rio and received his Masters degree from the University of Waterloo (1969), Canada, and his PhD from the University of California in Los Angeles (1974). His current research focuses on agent-oriented software engineering, multi-agent applications, autonomic computing and software reuse.

Paulo Masiero is a Professor of the Department of Computing Systems of the Institute of Mathematics and Computer Science of the Universidade de São Paulo, Brazil, where he is one of the leaders of the Software Engineering Research Group. His main research interests are testing of aspect-oriented programs, web services and embedded systems, software product lines, software design.

Claudio Sant'Anna received her PhD in Computer Science at PUC-Rio – Brazil (2008). He is an Assistant Professor at the Federal University of Bahia (UFBA) – Brazil. His main research interests include software design, with emphasis on modularity, separation of concerns, comprehension, metrics, visualization, architecture, aspect-oriented programming and empirical software engineering.

Fabiano Ferrari received his PhD in Computer Science from the University of São Paulo (USP) in 2010. He is currently an Assistant Professor at the Computing Department of the Federal University of São Carlos (UFSCar). His research interests include: testing of Object-Oriented and Aspect-Oriented software, software metrics, Experimental Software Engineering and Agile Methods.

Vander Alves earned his PhD in Computer Science from the Federal University of Pernambuco (UFPE), in 2007. Since then, he held postdocs positions at Lancaster University (2007–2008) and at Fraunhofer IESE (2008–2009). He is interested in research and development of Software Product Lines and currently holds the tenured position of Professor Adjunto 2 at University of Brasilia (UnB).

Roberta Coelho is an Associate Professor at Federal University of Rio Grande do Norte, Brazil. She holds a PhD in Computer Science fom Pontifical Catholic University of Rio (PUC-Rio) (2004–2008). She have conducted empirical studies in the context of reliability of AO applications, and her research interests include static analysis, exception handling, dependability and empirical software engineering.

Eduardo Figueiredo is an Assistant Professor at the Federal University of Minas Gerais (UFMG) since 2010. He received his PhD degree in Software Engineering from Lancaster University (UK) in 2009. His research interests include aspect-oriented programming, software product lines, software reuse, empirical software engineering, and software metrics.

Paulo F. Pires is currently an Associate Professor of the Federal University of Rio de Janeiro and also a member of the Centre for Distributed and High Performance Computing at the University of Sydney. He received his PhD from Federal University of Rio de Janeiro, Brazil, in 2002.

Flávia C. Delicato received her PhD from Federal University of Rio de Janeiro, Brazil, in 2005. She is currently an Associate Professor of the Federal University of Rio de Janeiro. In 2010 she was a visiting academic at the University of Sydney, Australia. She integrates the Centre for Distributed and High Performance Computing at the University of Sydney. She participates in several research projects with funding from International and Brazilian government agencies.

Eduardo Piveta holds a PhD in Computer Science from Federal University of Rio Grande do Sul (UFRGS). His research interests are mainly related to programming languages, and to the design, implementation, and evolution of software systems. Currently he is a lecturer at Universidade Federal de Santa Maria (UFSM) at Santa Maria, Brazil.

Carla Silva received her PhD in Computer Science at UFPE – Brazil (2007). She is an Associate Professor at the Federal University of Pernambuco (UFPE), where she is member of the Requirements Engineering Laboratory (LER). Her main research interests include Requirements Engineering, Agent Oriented Development, Aspect Oriented Development, Model-driven Development and Software Product Lines.

Valter Camargo holds a PhD in Software Engineering from University of São Paulo. Currently, he is an assistant professor at the Compurting Department in Federal University of São Carlos (UFSCar), Brazil. His research interests involve Reuse (Frameworks and Software Product Lines), ADM (architecture-driven modernization), aspect-oriented frameworks and model-driven engineering.

Rosana Braga is an Assistant Professor of the Department of Computing Systems of the Institute of Mathematics and Computer Science at Universidade de São Paulo. Her research at the Software Engineering Research Group comprise software product lines, aspect-oriented software development, component and service oriented architectures, model-driven development, application generators, frameworks, patterns, and pattern languages.

Julio Leite is an Associate Professor at PUC-Rio; member of the Working Group 2.9 (Software Requirements Engineering) of IFIP (International Federation for Information Processing); member of the editorial board of the Journal of Requirements Engineering Springer; has been a member, since 1993, of the program committee of the IEEE international conference on requirements (RE); member of the Association for Computing Machinery and the IEEE Computer Society; founding member of the Brazilian Computer Society (SBC), co-editor of the book Perspectives on Software Requirements and co-founder of the WER (Workshop on Requirements Engineering) series and of the FEES (Education Forum in Software Engineering).

Otávio Lemos received his M.Sc. (2005) and D.Sc. (2009) degrees in computer science from the University of São Paulo (USP). He was a visiting researcher at UCIrvine from January to July 2007, and from June to August 2012. He's currently an Assistant Professor at the Federal University of São Paulo (UNIFESP), doing research on software testing, reuse, and experimentation.

Nabor Mendonça has a PhD in Computing from Imperial College London – U.K. (1999). He is a Titular Professor at the University of Fortaleza (UNIFOR), Brazil. His main research interests are distributed systems, software engineering, middleware and cloud computing.

Thais Batista received her PhD in Computer Science at PUC-Rio – Brazil (2000). She is an Associate Professor at the Federal University of Rio Grande do Norte (UFRN) – Brazil. Her main research interests include Software Architecture, Software Product Lines, Modularization, Middleware and Cloud Computing.

Rodrigo Bonifácio is an assistant professor at the Computer Science Department / University of Brasília. He received his PhD in Computer Science from the Informatics Center / Federal University of Pernambuco. His research interests include: automated analysis and derivation of software product liens and empirical software engineering.

Nélio Cacho is an assistant professor in computer science at the Federal University of Rio Grande do Norte, Brazil. His research interests include exception handling, aspect-oriented programming, software product lines, and empirical software engineering. He received his PhD in computer science from the University of Lancaster (2008).

Lyrene Silva received her PhD in Computer Science at PUC-Rio, Brazil (2006). She is an Assistant Professor at the Federal University of Rio Grande do Norte (UFRN) – Brazil. Her research interests include Requirements Engineering, Software Product Line and Domain Modeling.

Arndt von Staa received his PhD degree in Computer Science from the University of Waterloo, Canada (1974). He is a Full Professor at PUC-Rio. His main research interests include modularity, software quality, testing, and CASE tools.

Fábio Silveira received his PhD in Computer Science from the Technological Institute of Aeronautics (ITA) in 2007. He is currently an Assistant Professor at the Science and Technology Institute of the Federal University of São Paulo (UNIFESP). His research interests include: Object-Oriented and Aspect-Oriented software testing, Experimental Software Engineering, Agile Methods, and Metadata.

Marco Túlio Valente is an assistant professor in the Computer Science Department at the Federal University of Minas Gerais, Brazil. His research interests include software architecture, software remodularization and software maintenance and evolution. Valente has a PhD in computer science from the Federal University of Minas Gerais.

Fernanda Alencar received her Ph.D. in Computer Science (1999) at Federal University of Pernambuco, Postdoctoral at the University of Lisbon, Portugal (2006) and Postdoctoral Universidad Politécnica de Valencia (2008–2009). She is currently Associate Professor at Federal University of Pernambuco. Her main research interest include requirement engineering, agent oriented development, aspect-oriented software engineering, development driven by models and sociotechnical systems.

Jaelson Castro is an associate professor at the Universidade Federal de Pernambuco, Brazil, where he leads the Requirements Engineering Laboratory (LER), since 1992. He received his Ph.D. in 1990 from Imperial College, London. His research interests include software engineering, requirements engineering, agent-oriented development, aspect-oriented development, model-driven development, and software product lines.

Ricardo Ramos received his PhD in Computer Science at UFPE – Brazil (2009). He is an Associate Professor at the Federal University of Vale do São Francisco (UNIVASF). His main research interests include requirement engineering, distance learning, educational games and Object-Oriented Programming.

Rosangela Penteado is an Associate Professor of Computing Department at the Federal University of Sao Carlos. She received her Ph.D from the University of Sao Paulo. Her research interests include: Object-Oriented and Aspect-Oriented Software Reengineering, Software Maintenance and Evolution, Frameworks, Model-Driven Development, Software Product Lines.

Cecília Rubira is a Full Professor of the University of Campinas (UNICAMP), Brazil. She received her Ph.D. in Computing Science (1994) from the University of Newcastle, UK. Her current research interests are fault tolerance, exception handling, software architectures, component-based software engineering, software product lines and aspect-based software development. She has co-authored more than 100 scientific papers, book chapters, and books in these areas, and also supervises a number of MSc and PhD students at UNICAMP.