Information and Software Technology 55 (2013) 395–411

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

Aspect-oriented model-driven code generation: A systematic mapping study

Abid Mehmood *, Dayang N.A. Jawawi

Department of Software Engineering, Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

ARTICLE INFO

Article history:
Received 19 September 2011
Received in revised form 21 July 2012
Accepted 9 September 2012
Available online 17 September 2012

Keywords:
Aspect-oriented software development
Model-driven engineering
Code generation
Systematic map

ABSTRACT

Context: Model-driven code generation is being increasingly applied to enhance software development from perspectives of maintainability, extensibility and reusability. However, aspect-oriented code generation from models is an area that is currently underdeveloped.
Objective: In this study we provide a survey of existing research on aspect-oriented modeling and code generation to discover current work and identify needs for future research.
Method: A systematic mapping study was performed to find relevant studies. Classification schemes have been defined and the 65 selected primary studies have been classified on the basis of research focus, contribution type and research type.
Results: The papers of solution proposal research type are in a majority. All together aspect-oriented modeling appears being the most focused area divided into modeling notations and process (36%) and model composition and interaction management (26%). The majority of contributions are methods.
Conclusion: Aspect-oriented modeling and composition mechanisms have been significantly discussed in existing literature while more research is needed in the area of model-driven code generation. Furthermore, we have observed that previous research has frequently focused on proposing solutions and thus there is need for research that validates and evaluates the existing proposals in order to provide firm foundations for aspect-oriented model-driven code generation.

1. Introduction

Model-Driven Engineering (MDE) is an approach to software development that stresses upon making models the primary development artifact and subjecting them to a refinement process, through automatic transformations, until a running system is obtained. By doing so, MDE aims at providing higher level of abstraction in development of systems which further results in an improved understanding of complex systems. Moreover, it addresses problems in software systems development that originate from existence of heterogeneous platforms. It achieves this through keeping different levels of model abstractions; and by transforming models from Platform Independent Models (PIMs) to Platform Specific Models (PSMs). In this context, automatic generation of application code (i.e. automatic model-driven code generation) offers many advantages such as the rapid development of high quality code, reduced number of accidental programming errors, enhanced consistency between design and code, to name a few. In addition to these, several other benefits have also been reported in [1,2].

Aspect-oriented software development is an approach to software engineering which allows explicit identification, separation, and encapsulation of concerns that cut across the primary modularization of a system. These *crosscutting concerns* cannot be clearly decomposed from primary functionality (*core concerns*) of the system, and thus cannot be effectively modularized, when using other well-known development techniques such as object-oriented development. Hence these concerns end up *scattered* throughout the system and *tangled* with the core concerns of system. Even though the crosscutting concerns usually originate from non-functional requirements such as logging, security, persistence, and optimization but the phenomenon encompasses the functional ones also, which often have their behavioral logic spread out over several modules. Using aspect-orientation, these concerns are identified, modeled and implemented independent of each other as well as separate from the main functional concerns of the system. Once separated in this way into modules, these concerns need some composition mechanism to control where and when concern behavior is applied. This effectiveness of modularization is achieved through applying aspect-orientation at analysis phase using Early Aspects [3], during design using Aspect-oriented Modeling [4], and using Aspect-oriented Programming [5] for implementation. The separation of crosscutting concerns from core functionality of the system achieved through aspect-orientation eventually results in improving several software quality factors including maintainability, extensibility and reusability.

* Corresponding author. Tel.: +966 55 228 13 73.
E-mail addresses: mabid4@live.utm.my (A. Mehmood), dayang@utm.my (D.N.A. Jawawi).

In the context of MDE, Aspect-Oriented Modeling (AOM) uses the aspect technology by modularizing and composing cross-cutting concerns during the design phase of a software system. A number of approaches to AOM (e.g. [6,7]) have been proposed in the past. All these modeling languages are compliant with Model-Driven Architecture (MDA) vision [8], and thus provide tools to integrate themselves with the MDE process. The main purpose of these modeling languages is to support specification of PIMs and subsequent automatic transformation to PSMs and code.

Aspect-oriented models developed using AOM approaches can be integrated with model-driven development in at least two different ways; and this eventually results in forming two distinct lines of research in this context. Along the first line are approaches that propose using a model weaver to compose the base model (one that models core concerns) and the aspect model (model which represents crosscutting concerns) in such a way that a non-aspect-oriented (object-oriented) model is obtained. Then standard code generation approaches can be used to generate code into one of the object-oriented programming languages. In contrast, the second line of research comprises of approaches that explore direct transformation of the source aspect-oriented model into code of a target aspect-oriented language, and then rely on weaver provided by the target language to deal with crosscutting aspects.

This study has been conducted in the specific context of approaches that contribute to aspect-oriented modeling or use these models for code generation. In order to get an overview of existing research in this context, we actually performed a systematic mapping study of aspect-oriented modeling and aspect-oriented model-driven code generation approaches. Apart from getting an overview, this study also aims at identifying and presenting results from literature that are valuable from perspective of possible future enhancements and use. Further, it aims at identifying needs for future research in this particular area. Some earlier papers and brief reviews have presented overview of aspect-oriented modeling techniques and associated challenges. Similarly, there are few studies that have elaborated challenges and prospects in model-driven code generation. However, no existing reports have conducted a systematic mapping study of the area.

A systematic mapping study is a way of identifying and classifying research related to a topic, and has been adapted from other disciplines to software engineering by Kitchenham and Charters [9]. When used for a specific research area, it categorizes different types of research reports in various dimensions, and often provides a map of its results. Systematic mapping studies have been recommended mostly if little relevant evidence is found during initial study of the domain or if the topic to be investigated is very broad [9]. In contrast to systematic literature reviews, systematic mapping studies are conducted at a coarse-grained level and they aim only at finding and identifying evidence relating to a research question, and at identifying research gaps in order to direct future research. In this context, we believed it would be appropriate to conduct a systematic mapping study since aspect-oriented modeling appeared to be a broader concept with multiple research focus areas, while little work was found that addressed the specific area of aspect-oriented code generation.

Following this introduction, this paper is structured as follows: In Section 2, we present a short overview of the context in which current study has been conducted and justify its need. Section 3 describes how the systematic mapping methodology has been applied. The classification schemes and their various dimensions are discussed in Section 4. Section 5 is dedicated to present the results of mapping the selected primary studies and discussion of research questions. We discuss the overall results and identify the potential limitations of our study in Section 6. Section 7 concludes our presentation.

2. Background and motivation

In this section we provide a brief background to aspect-oriented model-driven code generation presenting the core ideas and motivation behind it, and some works related to it. We also justify the need for a systematic mapping study in this section.

2.1. Aspect-oriented model-driven code generation

Software development projects are aimed at producing high-quality software within allocated time. However, as the projects grow in size and complexity, achieving the goals of quality and on-time delivery become more challenging. For this reason, software projects often end up running over schedule, as found in software project management studies such as [10]. Moreover, these off-schedule projects often relinquish quality in order to meet the project deadlines, further leading to software products which are less reliable, less maintainable, and less adaptable. Therefore, there is need for techniques that can promise reduced delivery time for high quality software products. At the design level, visual modeling languages [11–14] help in given context by providing modeling and model-checking capabilities. During the implementation and maintenance phases, the same can be achieved by applying automatic code generation. Automatically generated code, if correctly obtained, can enhance the benefits of high-level modeling and analysis. Hence, in past, it has been deemed ideal to develop approaches that generate or help to generate executable code from high level design models. So far as the benefits of automatic model-driven code generation are concerned, the most significant advantages include reduction in development time, and improvement in quality from different perspectives such as maintainability, extensibility, and reliability [15].

The majority of automatic code generation approaches have addressed automatic code generation for object-oriented analysis and design models. Moreover, code generation has been presented using formal notations. Examples of code generation using formal notations include Petri Nets [16], Software Cost Reduction (SCR) [17], and Cinderella SLIPPER [17]. These approaches have achieved full code generation and they have proposed techniques for optimized code generation. In some other works such as [18–21], models represented in UML have been used to generate fully executable object-oriented code. Many of currently available commercial (e.g. IBM Rational Software Architect [22], AjileJ StructureViews [23], MagicDraw UML [24]) as well as open source (ArgoUML, Eclipse-UML2 Tools) object-oriented CASE tools support the generation of code stubs.

It has to be emphasized here that while we focus on code generation (since code seems to be the eventual outcome), the importance of studying the associated modeling approaches cannot be overlooked. This is because a design model developed using one of modeling approaches serves as input to code generation process. Therefore, the comprehensiveness and quality of generated code is directly linked with those of the modeling approach. For this reason, this study includes both aspect-oriented modeling as well as code generation topics. Aspect-oriented model-driven code generation can in fact be seen as a realization of the idea of direct transformation from an aspect-oriented model into aspect-oriented programming language code (described in Section 1). In particular, it refers to an approach of generating code from base and aspect models independently in the first step, and then weaving base and aspect code in the second step. For this reason, the approach may be called as one of the Generate-Then-Weave approaches [25]. From the perspective of generated code, it is usually in contrast to Weave-Then-Generate approaches that generate code from a woven view of base and aspect design models since they are often used to generate object-oriented code. While Weave-Then-Generate approaches may be considered ideal for model analysis and execution, they may result in problems if the generated code is to be maintained manually [26,27]. It has to be emphasized here that one would expect the need for manual code maintenance and evolution until MDE becomes an extremely mature discipline. Another difficulty with these approaches is associated with separation of concerns that have once been composed during the weaving process. Once a model is woven, clear separation of concerns, the founding principle behind aspect-orientation, becomes blurred. This dilemma of losing clear boundaries of concerns while translating models into implementation may further lead to diminishing evolvability, traceability, reusability and maintainability of the software system [28].

The essential goal of research that proposes transformation of an aspect-oriented model directly into aspect-oriented code is to benefit from code level features of aspect-oriented programming languages. The approaches in this category do not suffer from problems mentioned above, as they tend to maintain the separation of concerns from model to code. Therefore, one may conclude that they are mainly inspired by benefits resulting from existence of a direct mapping between constructs of aspect-oriented design model and the aspect-oriented programming language. In this regard, Hovsepyan et al. [26] have reported that approaches that target aspect-oriented programming languages result in compact, smaller, less complex and more modular implementations. They have conducted a number of quantitative studies and experiments. They found that maintaining the aspect-oriented paradigm throughout the system development stages offers several benefits compared to shift from aspect-oriented models to object-oriented code. Similarly, some empirical studies such as [29–32] have also reported potential benefits of using aspect-oriented techniques in software development.

In past, few efforts have been made to achieve automatic aspect-oriented code generation and initial results have been reported in the literature. All such approaches are naturally based on model-driven architecture, meaning that they use a source model developed in some notation extended from UML as input and generate the target aspect-oriented code according to some transformation definition. However, as we have shown in a previous study [33], each of these approaches formulates certain specific features of aspect-oriented model-driven code generation while eliminating others. There is need for more research in this particular area to raise maturity of such approaches and to serve the long term goal of fully executable aspect-oriented code generation from models.

2.2. Related work

In an effort to evaluate and compare different approaches to aspect-orientation, some studies have been presented in recent years. Such studies can broadly be distinguished into two categories with respect to their focus: studies focusing on aspect-oriented modeling and ones particularly emphasizing on aspect-oriented code generation approaches.

So far as first category is concerned, some previous work has presented comparison of aspect-oriented modeling approaches pursuing some distinguished goals. Wimmer et al. [7] have defined a detailed evaluation framework to evaluate existing AOM approaches with focus on comparability in general. The major distinction of their work from all other surveys on the topic is the breadth and depth of evaluation. Chitchyan et al. [34] have presented an extensive work with the goal of "developing integrated aspect-oriented requirements engineering, architectures, and design approaches". Therefore, they have provided review of all significant work on both aspect-oriented as well as non-aspect-oriented approaches to software development. Similarly, Reina et al. [35] have investigated some AOM approaches with specific goal of evaluating dependency of each approach on particular platform and on specific concerns. Op de beeck et al. [6] have presented a comparison of AOM approaches within the context of product line engineering and with the goal to position AOM approaches within software development life cycle for large-scale system.

With respect to studies on code generation approaches, in an effort to evaluate existing approaches for aspect-oriented code generation, we have compared six approaches on the basis of a set of well-defined questions [33]. To the best of our knowledge, it was the first work intended to evaluate specifically the aspect-oriented code generation approaches. However its scope is limited to highlighting the presence or absence of features in the surveyed approaches.

2.3. Need for a systematic mapping study

Systematic mapping studies belong to Evidence-Based Software Engineering (EBSE) paradigm [36]. They provide new, empirical and systematic methods of research. Although several studies have been reported in the broader context of aspect-orientation (e.g. [6,7,34,35]), we are not aware of any systematic mapping study that has been conducted in this field. Given the fact that various types of research have appeared addressing varying focus areas at different levels of granularity related to broader topic of aspect-oriented model-driven code generation, there is need for a more systematic investigation of the topic. Therefore, the current study is intended to contribute to aspect-oriented model-driven code generation through a systematic and evidence-based approach. This study may help researchers in the field of aspect-oriented modeling and its integration with MDE through providing an overview of the current research in the area. Furthermore, it may serve as a first step towards more thorough examination of the topics addressed in it with the help of systematic literature reviews.

3. Research method

Petersen et al. [37] describe the process of carrying on a systematic study in software engineering. We conducted the current study by considering their guidelines. However, in addition to using the classification schemes proposed in their work for some dimensions, we found it appropriate to define some schemes specific to our topic.

As shown in Fig. 1, our mapping study was performed in essential process steps of: (1) defining research questions, (2) defining search strategy, (3) screening of primary studies, (4) defining classification schemes, and (5) mapping of studies.

3.1. Research questions

This study aims at obtaining an overview of current research in the area of aspect-oriented modeling and subsequent code generation using these models. We defined three research questions to elaborate this overall goal:

RQ1 What are the most examined aspect-oriented design and aspect-oriented model-driven code generation topics, and to what extent have these topics been investigated? Moreover, what types of contributions have been presented so far? Aspect-oriented constructs can be supported in different ways and using different modeling diagrams at the design level. Similarly, support for code generation from aspect-oriented models can be implemented based on different models and constructs. We defined this question to see which modeling constructs and code generation methods have been addressed already. Answer to this question is expected

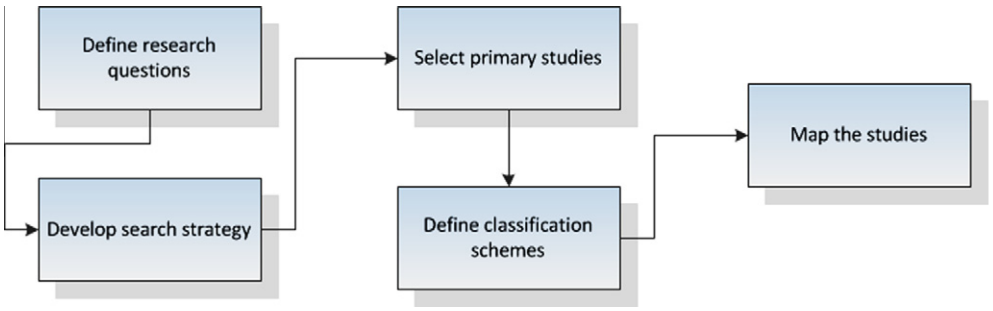

Fig. 1. Systematic mapping process (adapted from Petersen et al. [37]).

to identify needs for complementary research. Furthermore, this question is intended to see to what extent these approaches currently support the overall goals, specifically by what type of contributions.

RQ2 Which forums are often used to publish research on aspect-oriented modeling and aspect-oriented code generation? In our initial investigation of the domain we found that aspect-oriented software development was topic of some dedicated conferences and workshops. By defining this question, we want to see what other forums are used to publish research in this area.

RQ3 What different types of research are presented in literature and what extent has been addressed? In order to increase the credibility of research, the use of empirical studies and better established approaches is advised [38]. In this context, we want to classify different research types in the specific area of aspect-oriented modeling and code generation.

3.2. Search strategy

We developed a well-defined search strategy in order to identify the maximum number of relevant primary studies. We describe it from three perspectives: search scope, search method, and search strings used.

As far as the search scope is concerned, we did not limit the scope of our search to any specific research venues to find the maximum amount of related research work. However, the search results are limited to publication period between July 1997 and May 2012. We chose this start date because the first paper introducing aspect-oriented programming (i.e. [5]) published in ECOOP'97. The search results up to May 2012 have been considered in this study because we performed the search at this time. However, the search scope for manual search (described below) is limited to the periods specified for each venue in the following.

In terms of search method, both manual and automatic searches were conducted. By manual search we mean the search conducted by manually browsing journals or conference proceedings. While the automatic search means searching major electronic data sources using a combination of predefined search strings. We conducted automatic search for a majority of venues since the manual search for some journals and conference proceedings published on those venues was expected to be extremely time-consuming.

For the manual search, we selected following venues because a large number of studies related to aspect-oriented modeling and code generation were found there during initial exploratory searches.

- *Journals*:
  - Transactions on Aspect-Oriented Software Development (TAOSD). Volumes I–VIII (2006–2011).
- *Conferences and workshops*:
  - International Conference on Aspect-Oriented Software Development – AOSD (2002–2009).
  - International Conference on Model Driven Engineering Languages and Systems – MODELS (2007–2011).
  - International Workshop on Aspect-Oriented Modeling – AOM (2007–2009).

For the automatic searches, we used the search string given in Table 1, which is representative of four basic concepts related to aspect-oriented modeling and code generation. The final string was developed by performing a number of pilot searches on selected electronic data sources. The main digital sources that were used to conduct automatic search were IEEEXplore, Science Direct, ACM Digital Library, and Springer Link.

Note that since the features provided by various sources as well as the exact syntax of search strings to be applied vary from one source to other, the string given in Table 1 was actually used to construct a semantically equivalent string specific to each source. The first part aims at finding studies related to aspect-oriented modeling techniques, while the second part is related to model-driven engineering domain in the context of aspect-oriented, and the third part is related to code generation techniques based on aspect-oriented models. Identical set of metadata values (i.e. title, abstract and keywords) from all sources was selected to apply the search string in order to ensure consistency.

3.3. Selection of primary studies

As previously stated, we used a combination of manual and automatic searches. Fig. 2 shows the process of selecting primary studies. We started with performing a number of exploratory searches on previously given digital libraries to determine an initial set of publications. In this step we also used some previously known papers [39–44] as the starting point and followed the references and citing publications. This step resulted in 18 publications. We used this initial set of publications to help us identify some journals and conference proceedings relevant to our study. Therefore, we decided to manually search Transactions on Aspect-Oriented Software Development (TAOSD), proceedings of the annual conference AOSD, proceedings of the MODELS conference and AOM workshop since they were identified to be well-known among AOP researchers and publications related to our study were likely to be found there. By screening titles in these four venues, we obtained some more relevant studies and the total number of studies increased to 69. These publications were screened to get an overview of the area and to define preliminary classification schemes. At this stage, we realized that modeling of aspect-oriented concepts has been topic of several studies, while little work was available on the specific topic of aspect-oriented code generation. Therefore, we decided to keep track of studies on both topics independently (see Table 2). However, studies which present a significant contribution to both aspect-oriented modeling and aspect-oriented code generation were listed under both categories.

Table 1
Search string used for automatic searches.

Concept	Alternatives used	
Aspect-oriented	[(aspect AND oriented) OR (aspect-oriented) OR AO OR aspectual OR crosscutting OR cross-cutting]	AND
Modeling	[design OR modeling OR model OR AOM OR UML OR weaving OR (design AND notation)]	OR
Model-driven	[(model AND driven) OR model-driven OR automatic OR automated OR MD]	AND
Code generation	[Code OR (Code AND generation) OR (generating AND code) OR (code AND transformation) OR (code AND evolution) OR (mapping AND code)]	

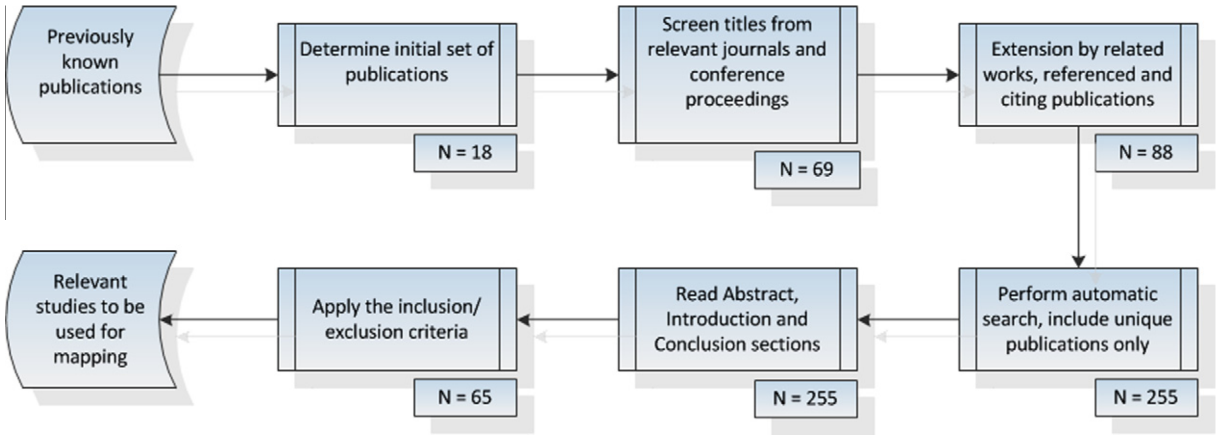
Previously known publications
Determine initial set of publications
N = 18
Screen titles from relevant journals and conference proceedings
N = 69
Extension by related works, referenced and citing publications
N = 88
Perform automatic search, include unique publications only
N = 255
Read Abstract, Introduction and Conclusion sections
N = 255
Apply the inclusion/ exclusion criteria
N = 65
Relevant studies to be used for mapping

Fig. 2. Study selection process.

In the next step we read Introduction and Related Work sections of the previously obtained set of publications and obtained a few more papers which were related to our study. We included publications with a strong focus on the aspect-oriented modeling techniques and using these aspect-oriented models as input to the automatic code generation process. This resulted in an additional 19 studies. Total number of studies up to this step was 88.

In the next phase we performed automatic searches using the search engines of electronic data sources i.e. IEEEXplore, Science Direct, ACM Digital Library, and Springer Link. We applied the search string given in Table 1. An overview of results obtained from manual and automatic searches is presented in Table 2. We also applied the search string to Google Scholar and ISI Web of Science. However, no unique contributions were found from ISI Web of Science, thus it is not shown in Table 2.

After performing automatic search, we excluded the duplicate publications by comparing results obtained in this step to those obtained through manual search. In case, a study was found reported on more than once, we selected the most recent and detailed version of the paper. At this stage, we also narrowed down the categories of publications to some extent by excluding non-peer reviewed publications, in order to ensure a level of quality as well as to avoid redundancy in contributions. Consequently, the automatic search resulted in 167 new unique contributions (see Table 2) after exclusion of non-peer reviewed publications. The sum of studies obtained by means of automatic search appears greater than the number of unique contributions stated above since some of the studies appear under both modeling and code generation in Table 2.

Table 2
Overview of search results.

Source	Studies retrieved		Studies selected	
	Modeling	Code gen.	Modeling	Code gen.
Manual searcho				
TAOSD	12	1	11	1
AOSD	15	1	7	1
MODELS	12	1	8	1
AOM	32	2	8	2
Automatic search				
IEEEXplore	33	18	6	3
ACM Digital Lib	35	13	3	2
SpringerLink	19	22	4	1
ScienceDirect	19	2	0	1
Google Scholar	12	6	5	2
Sum of studies	189	66	52	14

The authors considered the Abstract, Keywords, Introduction and Conclusion of each of these 255 studies identified to this point, for a second time, to decide about its inclusion or exclusion. A total of 190 studies were excluded either due to their limited relevance or meeting one of the other exclusion criterions. We found that according to our selection criteria, 51 publications were relevant to aspect-oriented modeling, 13 studies were relevant to code generation, while only one study (i.e. [45]) was found relevant to both sub-topics for this mapping study.

A listing of all criterions on basis of which studies were included or excluded is given below.

Inclusion:

- Studies that explicitly present an aspect-oriented modeling approach or technique, either by defining new constructs into UML or by using its extension mechanisms.
- Papers that present a unique solution to some modeling or code generation problem, although they do not contribute an approach on their own.
- Papers that implement an existing aspect-oriented modeling approach in practice and evaluate it.
- Studies that contribute to weaving mechanisms for aspect-oriented models, either by proposing a weaver or by proposing a unique extension to an existing weaver.

- Studies that report on approaches to aspect model transformations (model-to-model) or aspect-oriented code generation (model-to-code).
- Studies that propose approaches to mapping aspect-oriented models to aspect-oriented code.
- Papers that propose basic frameworks such as common case studies for demonstration or validation of aspect-oriented modeling or code generation approaches.

Exclusion:

- Papers which mentioned aspect-oriented modeling in the abstract only. This was required because we found many studies that mentioned aspect-oriented modeling in their opening sentences as a principal concept, however, the studies did not really address it. The same criterion was used for other concepts such as code generation and model-driven engineering as well.
- Papers that address weaving from aspect-oriented programming languages perspective rather than from modeling perspective.
- Papers that present only recommendations, guidelines or principles, instead of presenting a practical approach to modeling or code generation.
- Introductory papers for books and workshops.
- Editorials, keynotes, tutorial summaries, tool demonstrations and panel discussions, books, technical reports and other non-peer-reviewed publications.
- Duplicate reports of the same study found in different sources.
- Papers from industrial conferences, posters, and non-English publications.

Table 2 shows an overview of studies obtained through manual and automatic searches. Note that the automatic search shows only the number of unique contributions after removal of duplicate results that were retrieved from manual search previously. It also shows the number of studies that were selected based on the inclusion criteria mentioned above. The one study [45] that made a significant contribution to both modeling and code generation topics was added to both columns.

3.4. Defining a classification scheme

We used the classification schemes proposed by Petersen et al. [37] and classified the publications into categories from three perspectives: (1) focus area, (2) type of contribution and (3) research type. However, these categories were adapted to specifics of our mapping study. While categorizing and mapping the studies into classification schemes, we used an iterative approach. The resultant classification schemes are presented in Section 4.

3.5. Mapping of studies

The actual mapping was carried out by mapping each included study to a particular intersection set in the classification schemes defined in Section 4. The resulting mapping is presented in Section 5.

4. Classification schemes

As discussed previously in Section 3, publications are categorized from three different perspectives: focus area, type of contribution and research type.

4.1. Focus area

From a broader perspective, selected studies were divided into five research focus areas based on unique research topics they addressed. The *keywording* method described in [37] was used to identify these research focus areas. The five categories of research focus areas are briefly described below.

4.1.1. Modeling notation and process

This category includes studies that present a notation on its own, or in some way, contribute to the modeling process which uses some existing notation. Moreover, studies that elaborate or provide meta-models for definition of aspect-oriented modeling notations e.g. [46], have also been mapped into this category. It has to be emphasized here that since core modeling notations are generally complemented with a weaving mechanism, they are not categorized a second time into the model composition and interaction management category (that follows).

4.1.2. Model composition and interaction management

Studies that present a novel method of weaving aspect-oriented models, or present some solution related to management of model weavers are categorized here. Studies that contribute to improving methods of interaction between the base and aspect model constructs are also included in this category.

4.1.3. AO Code generation

This category reflects papers that present an explicit approach to generating aspect-oriented code from models (model-to-code), or contribute significantly to aspect-oriented code generation e.g. by elaborating mapping mechanisms from aspect-oriented models to code.

4.1.4. Code generation from specification of NFRs

Non-functional requirements (NFRs) just like aspects often crosscut more than one components of a software system. Therefore, we can say that specification of non-functional requirements (NFRs) is an area of software engineering that is very relevant to aspect-oriented design and programming. We identified some code generation approaches that address transformation of systems with specification of non-functional requirements. Keeping in view their similarity to aspect-oriented code generation, we have mapped them into this distinct category.

4.1.5. Applicability

This category includes papers that mainly focus on reporting evidence related to applying a specific modeling technique in practice. For example, papers that present a system model using some existing modeling approach, or proposals that develop a common infrastructure to apply a specific modeling notation.

4.2. Contribution type

The contribution type is divided into five categories described below:

4.2.1. Tool

It refers to contributions that focus on providing tool support for aspect-oriented modeling or code generation, either in the form of a prototype or a tool that can be integrated with existing frameworks.

4.2.2. Method

It refers to contributions that specifically provide a modeling, model composition or a code generation approach, e.g. a presentation of how a specific aspect-oriented construct will be modeled.

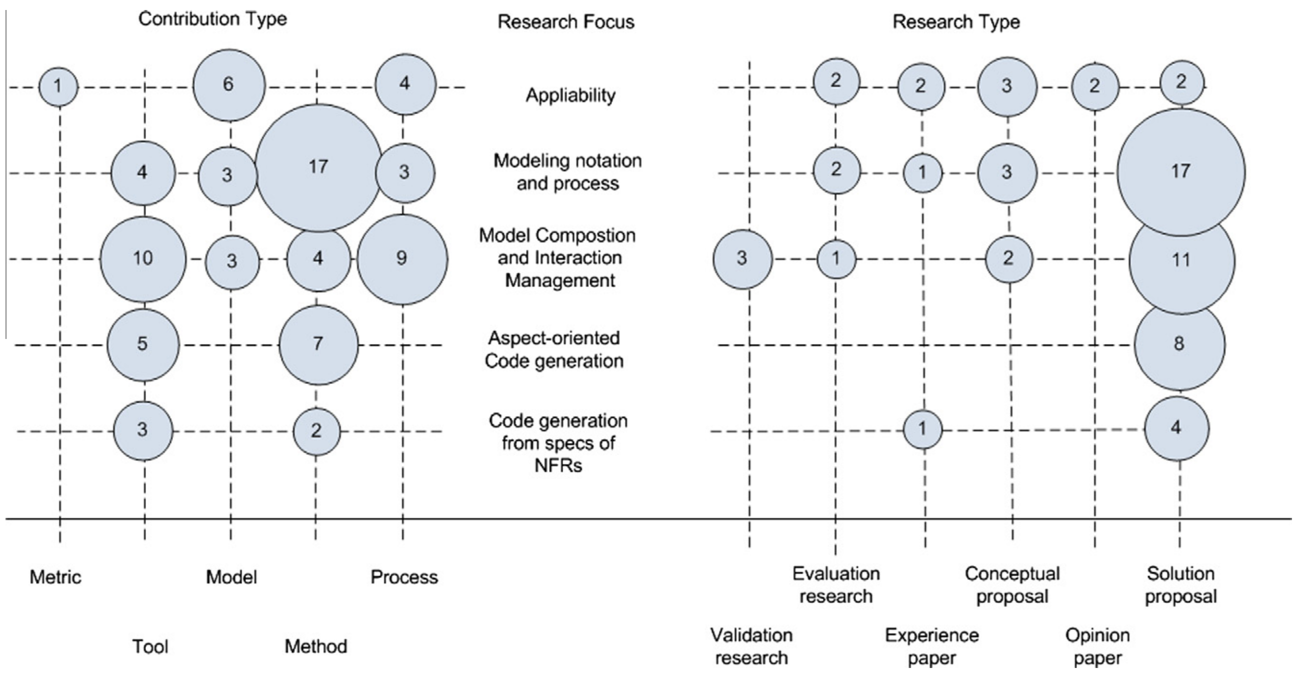

Fig. 3. Map of research focus on aspect-oriented modeling and code generation.

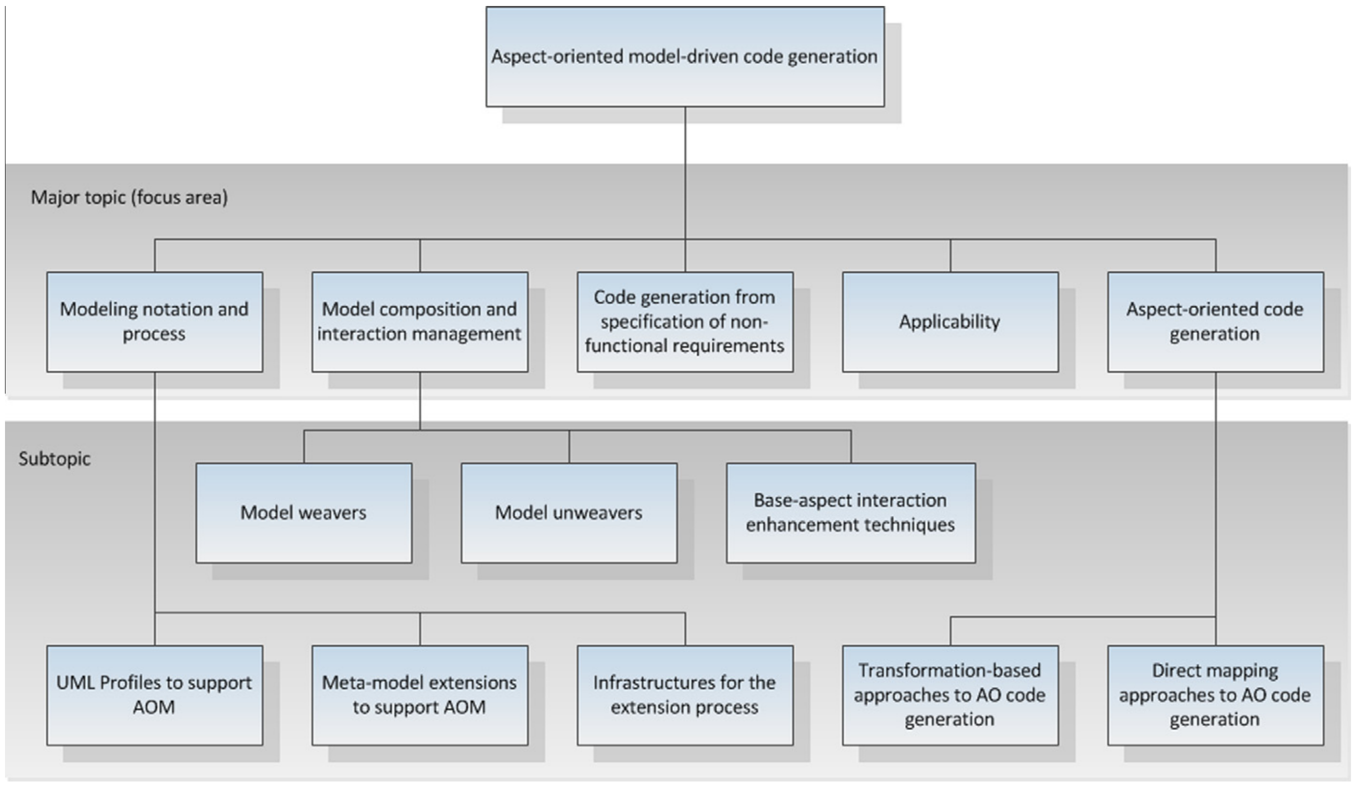
Aspect-oriented model-driven code generation
Major topic (focus area)
Modeling notation and process
Model composition and interaction management
Code generation from specification of non-functional requirements
Applicability
Aspect-oriented code generation
Subtopic
Model weavers
Model unweavers
Base-aspect interaction enhancement techniques
UML Profiles to support AOM
Meta-model extensions to support AOM
Infrastructures for the extension process
Transformation-based approaches to AO code generation
Direct mapping approaches to AO code generation

Fig. 4. Classification of the research related to aspect-oriented modeling and code generation.

4.2.3. Process

Papers that elaborate the aspect-oriented modeling or code generation approaches and provide description on their integration in overall software development process are categorized as papers contributing a process. For example, a paper presenting a way in which an abstraction can be used to represent behavioral aspects. Additionally, contributions that solve some particular problem related to aspect-oriented modeling or code generation, e.g. a technique proposed to detect conflicts between aspects at model level, are also mapped into this category.

4.2.4. Model

It refers to papers that conceptually discuss or make comparisons, explore relationships, identify challenges, or make classifications, etc.

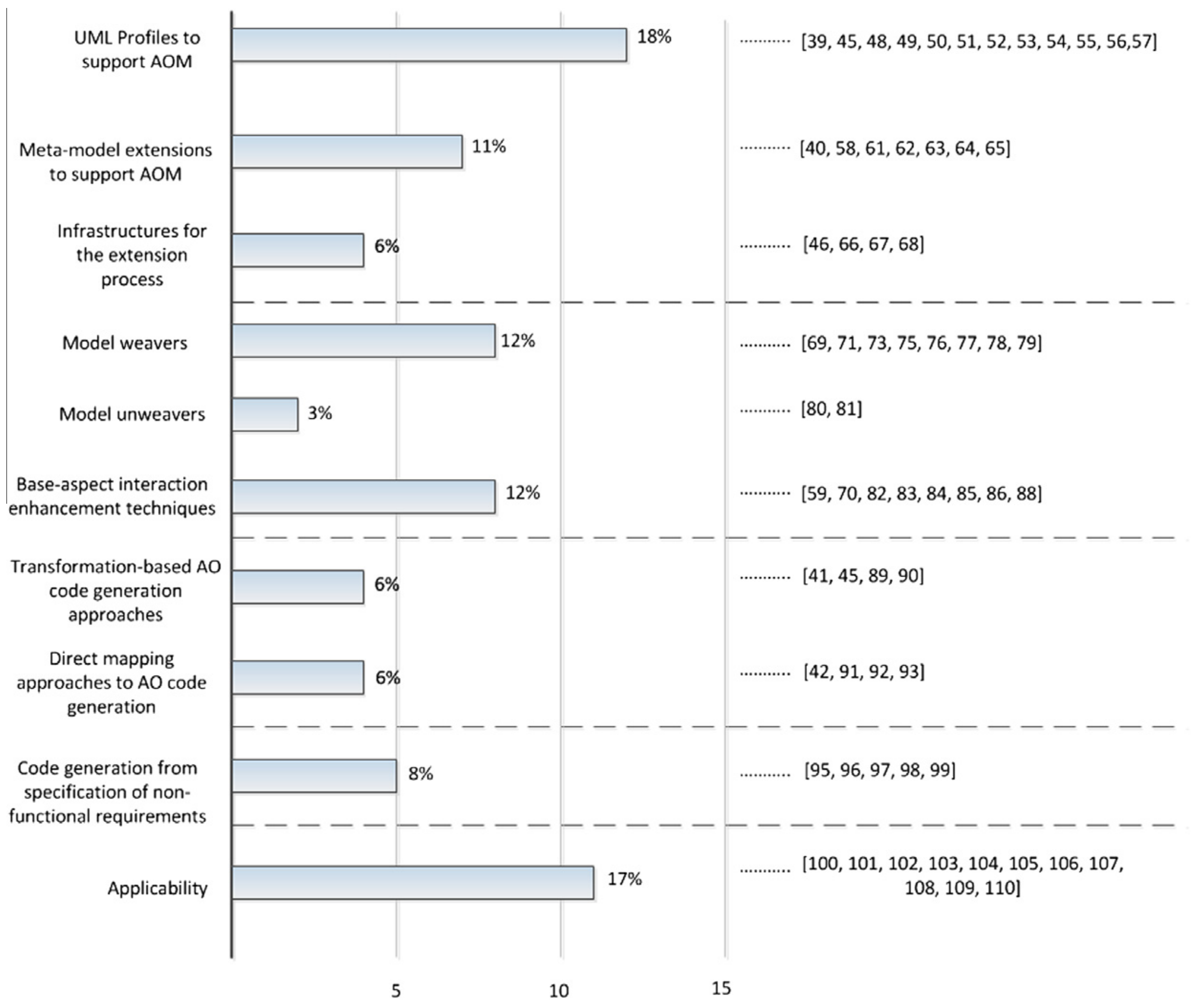

Fig. 5. Number of papers per research topic and references.

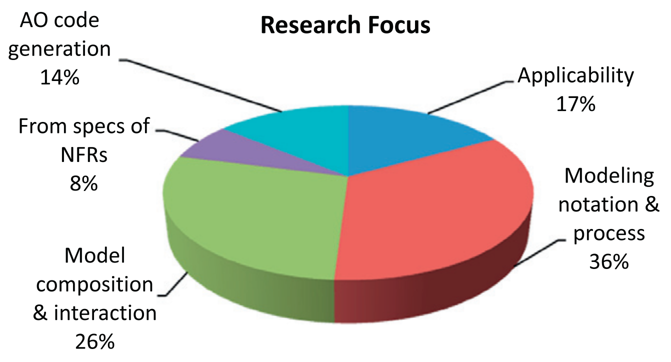

Fig. 6. Distribution of research focus.

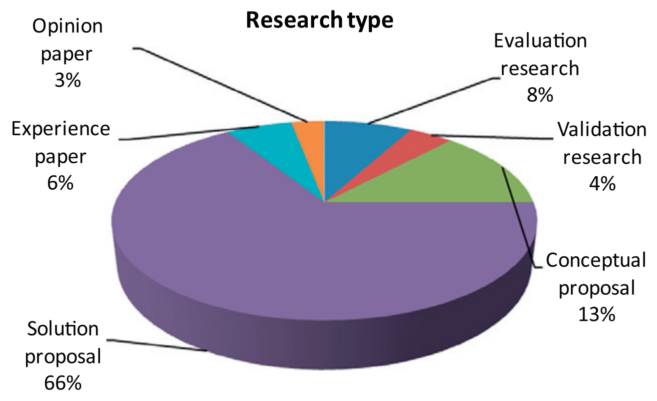

Fig. 7. Distribution of research type.

4.2.5. Metric

This is type of contributions that focus on proposing or applying metrics to effectiveness of aspect-oriented modeling or code generation approaches.

4.3. Research type

The research type reflects the research approach used in the primary study. We have used a scheme proposed by Wieringa et al. [47] for the classification of research types (*RQ3*). A brief description of research types follows:

4.3.1. Solution proposal

A solution proposal solves a problem by providing either a novel solution or a significant extension of an existing technique. It also highlights its benefits by either an example or thorough reasoning.

4.3.2. Validation research

The main purpose of validation research is to examine a solution proposal that has not yet been practically applied. Validation

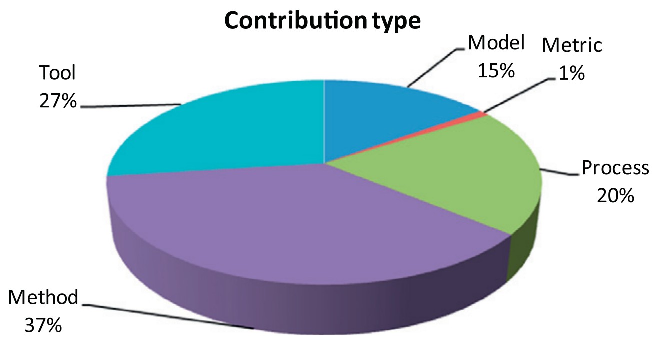

Fig. 8. Distribution of contribution type.

research is conducted in a systematic way and may present any of these: experiments, prototypes, simulations, mathematical analysis, etc.

4.3.3. Evaluation research

In contrast to validation research, evaluation research aims at examining a solution that has already been practically applied. It investigates the practical implementation of solution and usually presents results using field studies or case studies, etc.

4.3.4. Conceptual proposal

A conceptual proposal presents an arrangement to see things that already exist, in a novel way. However it does not precisely solve a particular problem. Conceptual proposals may include taxonomies, theoretical frameworks, etc.

4.3.5. Experience paper

An experience paper reports on personal experience of the author from one or more real life projects. It usually elaborates on what was accomplished in the project as well as how it was actually done.

4.3.6. Opinion paper

Opinion papers report on personal opinion of the author on suitability or unsuitability of a specific technique or tool. Similarly, these are sometimes used to share personal opinion describing as to how some technique or tool should have been developed, etc.

5. Mapping and discussion of research questions

In order to provide an overview of the field, the map over five existing research focus areas in the context of aspect-oriented modeling and code generation was developed with respect to research and contribution types (see Fig. 3). The map provides both

Table 3
Overview of publication forums for selected studies.

No.	Forum	Modeling	Code gen.
Journals			
1	Transactions on Aspect-Oriented Software Development	11	1
2	Journal of Object Technology (JOT)	3	
3	Software and Systems Modeling	1	
4	Science of Computer Programming		1
Conference proceedings			
5	International Conf. on Aspect-Oriented Software Development (AOSD)	7	1
6	International Conf. on Model-Driven Engineering Languages and Systems (MoDELS)	8	1
7	International Conference on the Unified Modeling Language	2	
8	International Conf. on Software Engineering Research, Management and Application (SERA)	1	
9	International Conference on Composition-Based Software Systems (ICCBSS)	1	
10	International Conf. on Model Transformation (ICMT)	1	
11	International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)		1
12	International Symposium on Computers and Communications (ISCC)	1	
13	Asia–Pacific Software Engineering Conference (APSEC)	1	
14	International Symposium on Computer Science and Computational Technology (ISCSCT)	1	
15	International Conf. on Information and Communication Technologies (ICTTA)	1	1
16	Brazilian Conference on Software Engineering		1
17	International symposium on Foundations of software engineering		1
18	World Congress on Intelligent Control and Automation (WCICA)		1
	World Congress on Computer Science and Information Engineering	1	
19	International Conference on Availability, Reliability and Security (ARES)		1
20	International Embedded Systems Symposium 2009		1
Workshops			
21	International Workshop on Aspect-Oriented Modeling (AOM)	9	2
22	International Workshop on Computer Science and Engineering	1	
23	International Workshop on Education Technology and Computer Science	1	
24	International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES)	1	1

Table 4
Research and contribution types presented by 22 papers on modeling notation and process.

Contrib. type	Research type				
	Tool	Method	Process	Model	Metric
Evaluation research	–	[52]	[68]	–	–
Solution proposal	[54,39,48,51]	[54,61,39,40,53,45,57,63,48,50,55,62,58,51,49]	[64,51]	[65]	–
Conceptual proposal	–	[46]	–	[66,67]	–
Validation research	–	–	–	–	–
Experience paper	–	[56]	–	–	–
Opinion paper	–	–	–	–	–

Table 5
Research and contribution types presented by 18 papers on model composition and interaction management.

Contrib. type	Research type				
	Tool	Method	Process	Model	Metric
Evaluation research	[84]	–	[84]	–	–
Solution proposal	[71,73,59,81,79,78,80,83]	[69,81,77,80,75]	[59,83,79,77,85]	–	–
Conceptual proposal	–	–	[70]	[86]	–
Validation research	[87]	[87]	[82,76]	–	–
Experience paper	–	–	–	–	–
Opinion paper	–	–	–	–	–

an outline of the emphasis of existing research as well as an indication of research gaps in the area.

The results of mapping indicate that majority of research papers are specifically dedicated to providing aspect-oriented modeling languages and elaborating the associated process. Composition of models and interactions between base and aspect models is another area which has been explored to a greater extent. However, integration of aspect-oriented models with model-driven engineering process, i.e. automated transformation of aspect-oriented models to code appears to be an underdeveloped area.

As far as the coverage of aspect-oriented design and code generation topics in existing research is concerned, we present our results in two different dimensions: (1) main topics in the area along with extent of their coverage and contribution types (*RQ1*) and research type (*RQ3*), and (2) forums used for publishing the related research (*RQ2*).

The first dimension of our results i.e. major topics along with specification of research types has been covered in the following Sections 5.1–5.5. We have organized each subsection in a way that it briefly describes the studies selected for each topic while highlighting the extent and nature of research. Furthermore, it identifies the type of contribution made by each selected study. The publications in this area can be divided into five major focus areas (see Fig. 4): modeling notations and process, model composition and interaction management, aspect-oriented code generation, code generation from specification of non-functional requirements and applicability of aspect-oriented modeling and code generation approaches. Fig. 4 also shows the major topics addressed by the existing research divided into related subtopics, where possible. Fig. 5 shows a summary of groups of papers identified per research subtopic.

An overview of the volume of research selected by major research focus areas is shown in Fig. 6. It shows that aspect-oriented modeling notations and associated composition mechanisms have been covered collectively by over 60% of the current research. On the other hand, code generation from aspect-oriented models has been addressed by a comparatively very small number of publications (14%). As far as the research type is concerned, solution proposals are in vast majority, covering over 65% of the aggregate, see Fig. 7. A small percentage of publications have reported on real-life experiences (6%), while validation and evaluation research has been presented collectively by rather small percentage of 12%. In terms of contribution types, Fig. 8 shows that previous research has turned much attention to presenting modeling notations and weavers (i.e. Method 37%). Tool support has also been significantly covered for both modeling and code generation sides.

Table 3 covers the second dimension of our results by summarizing selected papers by publication forums. Even though research has appeared on different forums, but we can see that the journal TAOSD and the annual conference AOSD appear to be the most relevant forums. The table also reveals that there are only two publications on aspect-oriented code generation that have appeared in a journal so far.

5.1. Modeling notation and process

In this section, we briefly discuss different studies related to modeling notations and the associated process. Table 4 lists the papers that focus on this topic. This is an area where most research effort is spent. Most works address the aspect-oriented modeling problem by providing a complete aspect-oriented extension to UML, while some provide limited support for selected aspect-oriented constructs. There are two different approaches to extend UML for aspect orientation: (1) *UML Profiles* (also referred to as *lightweight extension mechanism*) and (2) *meta-model extension* (also referred to as *heavyweight extension mechanism*). Some papers propose a common infrastructure for defining UML profiles to support aspect-oriented modeling. A summary of papers related to each of the subtopics discussed in the following is shown in Fig. 5.

5.1.1. UML Profiles to support AOM

UML Profiles extend the UML by defining profiles which group user-defined extensions to meta-model elements in terms of stereotypes allowing only the extensions that do not change the meta-model.

Zakaria et al. [48] present a UML profile and provide its integration with the Rational Rose CASE tool package. Aldawud et al. [49] elaborate the requirements for a UML profile, present a profile to support aspect orientation and describe its use for general-purpose aspect modeling. In [50] a UML profile has been presented that supports executable modeling at the specification level. This profile uses Object Constraint Language (OCL) to specify behavior on a higher level of abstraction than messaging between objects. Specification models are constructed incrementally.

Table 6
Research and contribution types presented by nine papers on aspect-oriented code generation.

Contrib. type	Research type				
	Tool	Method	Process	Model	Metric
Evaluation research	–	–	–	–	–
Solution proposal	[41,89,42,45,91]	[41,89,90,42,92,93,91]	–	–	–
Conceptual proposal	–	–	–	–	–
Validation research	–	–	–	–	–
Experience paper	–	–	–	–	–
Opinion paper	–	–	–	–	–

The Motorola WEAVR approach to aspect modeling [51] has been developed for the telecom infrastructure software industry. It is based on UML 2.0 and provides a lightweight profile completing the UML specification towards both the Specification and Description Language (SDL) and AOM concepts. Various diagrams such as composite structure diagrams and state machines have been adopted in UML 2.0 from the SDL. The modeling notation has been complemented with a powerful static model weaver. The weaver is integrated with the TAU G2 tool.

UML and its Action Semantics can be used to constructing executable models for object-oriented software systems. Some papers present support for aspect-oriented executable models by providing UML profiles. A UML profile that develops aspect-oriented executable models by extending the UML and its action semantics has been presented by Fuentes et al. [39]. They also provide a tool to support weaving and execution of their models. Ref. [52] presents a combination of aspect-oriented modeling approaches and a UML profile. This profile also performs model weaving for behavior models using UML 2 action semantics.

Wang et al. [53] present an approach to incorporate aspect-oriented constructs in the architecture design phase. They extend UML by standard extension mechanisms and propose notations for each AO construct to be used at the architecture design level. Mouheb et al. [54] present a profile for aspect-oriented modeling that specifies and integrates security concerns into UML. These concerns are defined in generic terms by security experts, instantiated by application developers and later automatically woven into UML model. They have also implemented their approach and provided the tool as a plug-into the IBM Rational Software Architect.

Some papers have presented profiles which are specific to meta-model of AO programming languages. Refs. [45,55] propose profiles to support meta-model of AspectJ language in UML. All programming constructs of AspectJ have been defined at the meta-level in each proposed profile. Pawlak et al. [56] have reported on their experience with evolving an elementary high-level graphical notation for aspect-oriented models. This notation is also based on the concepts defined in AspectJ. Jingjun et al. [57] use standard extension mechanism of UML to define new stereotypes to support various constructs of AspectJ. They add three new concepts to UML: *groups*, *point-cut relations* and *aspect classes*.

5.1.2. Meta-model extensions to support AOM

Meta-model extension mechanism to support aspect orientation refers to extending the UML meta-model through inheritance and redefinition of meta-model elements.

An aspect-oriented modeling solution based on the notion of Reusable Aspect Models (RAMs) has been proposed by Klein and Kienzle [58]. RAM models can describe both the structure and behavior of a concern since it integrates modeling techniques for class diagram, sequence diagram and state diagram. Furthermore, RAM also addresses the reuse of aspect models and supports the creation of aspect dependency chains. For weaving, it combines two distinct approaches to weaving of class diagrams [59] and sequence diagrams [60]. RAM has also been extended to support multi-view modeling by Kienzle et al. [61]. Moreover, the RAM models have also been extended to allow a modeler to expand an aspect model of a concern that can crosscut the nodes of a distributed system with distribution role definitions [62].

Zhang et al. have proposed techniques to design aspect-oriented class diagrams [63] and state machines [64] by introducing a new language construct, an aspect, to the UML. Similarly, Ballal and Hoffman [40] propose notations for major constructs and describe the use of a new diagram, the weaving diagram, to represent behavior.

A meta-model based on UML extension profile [49] for aspect-oriented modeling is proposed in [65]. Moreover, this work builds a framework of aspect-oriented modeling AOMF, and presents a way to model the dynamic behaviors of aspect-oriented systems.

5.1.3. Infrastructures for the extension process

Some papers present a general core and infrastructure for defining extensions or profiles to support aspect-orientation at modeling level. In order to identify commonalities and promote reuse in construction of meta-models, Fuentes et al. present a generic MOF meta-model of aspect-oriented design languages [46]. Their proposed generic meta-model contains a kernel package of metamodeling elements. They propose to extend this kernel package for construction of new AO design languages. They have also outlined an all-purpose guidance on construction of UML profiles to support aspect-orientation [66]. Similarly, a core for aspect-oriented support in UML has been presented in [67]. This core has been developed gathering different modeling elements from literature that focus supporting aspect-oriented constructs. Each of these modeling elements and related notations has been associated to a discipline.

The common core and aspect-oriented modeling approach proposed by Amalio et al. [68] is based on a visual language with formal semantics VCL. The modeling language is used for abstract and precise modeling and comprises a novel approach to visually modeling behavior of the system. The definition of this common core also contributes to the establishment of standards for modeling notations.

5.2. Model composition and interaction management

Several papers have contributed to specific topic of base and aspect models composition their interactions. Table 5 summarizes papers on this topic, whereas a summary of papers related to each of the subtopics discussed in the following is shown in Fig. 5.

5.2.1. Model weavers

Most papers in this category focus on providing weaving mechanisms for aspect-oriented models. Sanchez et al. [69] present an aspect-oriented model weaver that is capable of handling complex pointcuts in a model specified using JPDDs [70]. The model is processed through a number of transformations to add JPDD-specific details to base model. The abstractions provided by JPDDs hide the weaving complexity, thus making the input models more concise and problem-oriented.

Barais et al. [71] have proposed model composition operators to be used in conjunction with the KerTheme aspect separation and composition techniques [72]. White et al. [73] have proposed constraint-based weaving, which maps the model weaving to a constraint satisfaction problem [74], and afterwards determines the best weaving strategy with the help of a constraint solver. This strategy eventually provides some built-in benefits of using a constraint solver.

Hanenberg et al. [75] address the so called problem of *complete weaving*. They argue that run-time checks that determine positions where an advice associated to a specific joinpoint be executed are time-consuming and result in overhead. Therefore, they propose a weaver that supports partly woven aspects, the *morphing aspects*, and provides continuous weaving.

A proposal for an open framework to modeling and weaving of applications with crosscutting features, UMLAUT, has been evaluated by Ho et al. [76]. They show that the framework supports construction of new weavers simply by changing the weaving rules and automatically provides implementation of the new weaver. They elaborate the weaving process in UMLAUT through model transformation techniques. XWeave [77] is a model weaver presented by Groher and Volter. It weaves models and meta-models based on the Eclipse Modeling Framework.

Some papers address weaving of aspects specifically into sequence diagrams. Klein et al. [78] and Gronmo et al. [79] propose automated, semantics-based techniques to weaving behavioral aspects into sequence diagrams.

5.2.2. Model unweavers

Model unweaving is specifically relevant in the context of adaptive systems since without the unweaving support the adapted model has to be re-generated as a whole, including the aspects that were not modified. An approach to unweaving of aspects has been proposed by Klein et al. [80].

Another approach to dynamically weaving and unweaving of executable UML models is proposed by Fuentes et al. [81]. This approach also presents the essential infrastructure and tool support required to make the aspect-oriented models fully executable.

5.2.3. Base-aspect interaction enhancement techniques

Since aspects capture potentially crosscutting concerns, the interactions among different entities containing these crosscutting aspects tend to become extremely complex. Therefore, some papers are dedicated to discussing the issue of detection of inconsistencies in interactions. Mehner et al. [82] investigate interactions among different entities with crosscutting concerns and detecting potential inconsistencies at modeling level. For this purpose, they actually examine the use of an existing model analysis technique, which is based on graph transformations and inherently supports analysis and detection of possible inconsistencies. Similarly in [83], a graph-based technique along with tool support has been presented that uses a model checker to automatically detect aspect inconsistencies for models defined in a UML extension.

Join Point Designation Diagrams (JPDDs) [70] are proposed to support complex join point selections. These JPDDs are capable of expressing join point selections in compliance with three different conceptual models that are frequently used in aspect-oriented development.

MATA is a significant approach to handling communication between aspect models proposed by Whittel et al. [84]. It is different from other approaches in that aspect models defined in MATA have no explicit join points. MATA uses graph transformations to specify and compose models; therefore, any model element can become a joinpoint. This makes the model composition a special case of model transformation. The MATA approach has also been examined by applying it to a number of industrial case studies.

An approach based on mutually exclusive use of a merging algorithm and some composition directives has been presented in [59]. The composition directives provide an enhanced way of composition besides supporting model modification and allowing the default composition rules to be overridden, resulting in increased flexibility.

Some papers have generally highlighted the problems in handling interactions between aspects. In [86], some issues related to *composability* have been identified to be taken into account while specifying and matching joinpoints. Oldevik et al. [87] have systematically examined model composition contracts with the help of a non-trivial example from literature [88]. These contracts actually dictate how aspect compositions may or may not access and change the base models. To validate the use of contracts, they have also implemented a prototype. In a similar context, other work has focused on managing complexity of variability in aspect oriented models [85].

5.3. Aspect-oriented code generation

We identified a small number of papers that address the aspect-oriented code generation from models, see Table 6. The code generation approaches presented by these papers can be categorized into two major types: transformation-based and direct mapping approaches. A summary of papers related to each of the subtopics discussed in the following is shown in Fig. 5.

5.3.1. Transformation-based approaches to AO code generation

All approaches in this category use one of the existing transformation techniques and explicitly define the details of transformation from the visual model to code.

Victora et al. [89] propose a pattern-based approach to generate AspectJ code from Theme/UML [44] models. They develop XML representation of Theme/UML models and use the Theme approach to mapping from model to code. The code generator uses XMI to exchange model between UML and XML and has itself been implemented using XSLT. Similarly, in [90], an approach to generating AspectJ code from visual models of multi-modal scenario-based system specifications, defined in Live Sequence Charts (LSCs), has been presented. A pattern-based technique transforms the LSCs into AspectJ.

Evermann has proposed a template-based approach that generates code from UML-based specification of AspectJ meta-model [45]. UML XMI model interchange facilities are used to implement the actual code generation.

An aspect-oriented code generation based on graph transformations has been proposed by Bennett et al. [41]. Currently, this approach can be considered the most significant approach. This is because it has benefitted from some obvious advantages of using graph-based transformation; it has explicitly addressed the performance and correctness of transformation algorithms; and it is complemented with an open and extensive validation mechanism. In order to generate code, this approach first transforms the visual design model (developed in FDAF [94]) into a text-based notation (XML), then it transforms this XML model into AspectJ code. Both transformations are fully automated.

5.3.2. Direct mapping approaches to AO code generation

The approaches in this category do not provide much detail on the transformation process, but rather directly manipulate the source model to map it onto constructs in the target programming language.

Groher et al. [42] use extended UML model as input to code generation and generate AspectJ code stubs. Code is generated following concrete rules of mapping between extend UML model and AspectJ concepts. The CASE tool Together from Borland has been selected to automatically generate object-oriented base elements. The aspect-oriented code generation is implemented using extension mechanism of the tool.

Haitao et al. [92] have presented an approach that interprets aspect-oriented domain-specific models to aspect-oriented code in AspectC++. First the crosscutting concerns are modeled as separate aspects. Then the aspect-oriented model interpreter generates code by traversing these aspects. Weaver of AspectC++ is then used to weave these aspects into a real time system. Similarly, Hanenberg et al. [93] describe a tool-supported concrete mapping of Join Point Designation Diagrams (JPDDs) [70] to AspectJ.

An approach to mapping of aspect-oriented models developed using Reusable Aspect Models (RAM) into Java and AspectJ code has been proposed in [91]. Apart from addressing the aspect-oriented code generation, this paper contributes greatly to identifying and solving the common problems involved in the mapping process. This is contrary to other works e.g. [41,42] in that they mention the generation of aspect-oriented code without explicitly defining the correlation between elements at model and code levels.

5.4. Code generation from specification of non-functional requirements

Table 7 lists papers on code generation from specifications of non-functional requirements (NFRs).

Alhalabi et al. [95] provide modeling and a template-based C++ code generation framework for systems having non-functional (QoS) requirements. Further to systems with QoS properties, the approach presented in [96] generates automated, template-based code for safety–critical real-time systems, which also have numerous non-functional requirements. The PervML Generative Tool (PervGT) [97] presents support for modeling of such systems using a UML-like domain specific language (PervML) and model transformation into Java source code.

Wehrmeister et al. [98] propose a framework named GenERTiCA for modeling of distributed embedded real-time aspects that also supports code generation. They also report on results of their experience regarding the use of GenERTiCA to generate code and implement aspects in a research project.

Model-driven Theme/UML [99] proposes a transformation tool in compliance with model-driven architecture (MDA) standards, which specifies platform-independent models using Theme/UML and transforms them into platform-specific models as well as J2ME and .NET CF code.

5.5. Applicability

Table 8 lists papers on applicability of aspect-oriented modeling and code generation approaches.

In [100], the applicability of an aspect-oriented modeling approach during the software design phase has been demonstrated by presenting an aspect-oriented design of a common case study for aspect-oriented research community, the Crisis Management System (CMS) [101]. In [102], the HiLA approach has been presented to improve the modularity of UML state machines by extending them with aspect-oriented features. The paper focuses on highlighting the usefulness of HiLA by applying it to an extension of the CMS, a Car Crisis Management System.

Wehrmeister et al. [103] elaborate the development of an infrastructure for UML-based code generation tools by discussing one such infrastructure created to build GenERTiCA [98]. Gray et al. report on development of a programming-language-specific weaver which was built based on constructs of a specific transformation engine [104]. They make several observations to establish the feasibility of using a transformation system to develop a generic weaver.

Clarke et al. [105] take a step towards developing a standard design language for aspect-oriented software development approaches. They report on their experience regarding the investigation of the traceability between Theme/UML and AspectJ. They have assessed both languages in general and their incompatibilities in particular. Krechetov et al. [106] compare three existing aspect-oriented architecture design approaches and share their opinion regarding the suitability of a unified general-purpose aspect-oriented architecture modeling approach. Albunni et al. [107] define a criterion to evaluate ability of various UML diagrams to support aspect-oriented design. They select UML activity diagrams based on their evaluation and propose an approach based on these diagrams.

Some metrics for aspect-oriented design have been proposed [108] for designs developed using an extension of UML. Object Constraint Language (OCL) which is a part of UML has been used for definition of metrics.

In [109], the merging of model driven and aspect oriented techniques has been shown to work for better managing complexity of adaptive systems construction and execution.

Fabry et al. [110] have shared the results of their experience regarding the application of Theme/UML [44] and Motorola WEAVR [51] in an industrial setting. They found that both approaches failed to support half of the required interaction types whereas other half needed to be expressed using a workaround.

Table 7
Research and contribution types presented by 5 papers on code generation from specification of NFRs.

Contrib. type	Research type				
	Tool	Method	Process	Model	Metric
Evaluation research	–	–	–	–	–
Solution proposal	[99,97]	[95,96]	–	–	–
Conceptual proposal	–	–	–	–	–
Validation research	–	–	–	–	–
Experience paper	[98]	–	–	–	–
Opinion paper	–	–	–	–	–

Table 8
Research and contribution types presented by 11 papers on applicability (both modeling and code generation).

Contrib. type	Research type				
	Tool	Method	Process	Model	Metric
Evaluation research	–	–	[100]	–	–
Solution proposal			[102,109]	–	–
Conceptual proposal	–	–	–	[101,103]	[108]
Validation research	–	–	–	–	–
Experience paper		–	[104]	[105,110]	–
Opinion paper	–	–	–	[106,107]	–

6. Discussion

Overall goal of this systematic mapping study was to identify approaches that can contribute to applying aspect-oriented concepts in an MDE perspective; one way to do this is to generate aspect-oriented code from aspect-oriented models through an automated process. In order to achieve the goal of aspect-oriented model-driven code generation as a whole, we have found that further research is necessary in various dimensions. In this section we provide a summary of key findings of this systematic mapping study and discuss some effects of these findings on future research. We also highlight the limitations of this mapping study that may represent threats to its validity.

6.1. Principle findings

Recent proposals in domain of aspect-oriented modeling have focused mainly on providing design notations resulting in appearance of several proposals [34]. However, if we look at overall problem of the integration of aspect-oriented models into an MDE context, there is still a significant work to be done. In an MDE environment, models are the main focus for visualizing an executable view of the system and actually obtaining a working software system in an automated way. Therefore in order to integrate aspects into this larger context, the area of modeling accurate and complete behavior of aspects needs more attention along with solutions to verification of models. Very few works have been reported on in the literature (e.g. [39]) so far that have addressed model verification, but even these verification approaches provide limited infrastructure to verifying models, by means of execution only. Apart from the fact that verification done in this way cannot substitute a systematic verification process, it can lead to other problems, for instance it requires designers to know the precise details of advice transformations, thus resulting in usability problems. Therefore, there is need for systematic processes such as those defined by Xu and Xu [111] that can be applied to design test cases for verification of these models.

As far as model composition and aspect interaction are concerned, there are few approaches that view weaving as a special case of model-to-model transformation, thus combining the MDD and aspect-oriented modeling. Examples of such approaches are MATA [84] and XWeave [77]. These approaches can exploit some obvious advantages of the transformation approach they use, graph-based transformation in this case. However, Sanchez et al. [69] have reported that such approaches hardly ever consider the advanced mechanisms for specification of pointcuts (i.e. conditions under which aspects are composed to base). As a result of this limitation, these approaches are not appropriate to modeling situations that require weaving beyond simple model composition and transformation e.g. situations where complex join-points are required. Furthermore, the weavers are mostly static in a sense that they are unable to weave or unweave aspects during model execution. Fuentes and Sanchez [81] have identified several situations where static weaving is insufficient. They have presented an approach that dynamically weaves and unweaves aspects during the model execution time. This is a significant contribution in a sense that it sets the foundations for more sophisticated enhancements towards model simulation and model testing of adaptive applications. However, like many others discussed previously, this approach also lacks support for complex join-point specifications. Hence further research is required to enhance the composition techniques.

The next challenge that comes after developing an appropriate model of the system in an aspect-oriented way is the problem of code generation from this model. The modeling and composition approaches discussed above may ideally be customized and used as input to an automated code generation process. Some proposals have appeared in literature (e.g. [39,81]) that suggest generating object-oriented code rather that aspect-oriented, once the model has been transformed into a UML executable model. These approaches mainly rely on existing object-oriented code generation tools such as Rhapsody, TAU G2, Rational RT, and ArgoUML. However, some empirical studies have been conducted such as [29–32] that explore and report the benefits of using aspect-oriented techniques in software development. Hovsepyan et al. [26] have discovered on basis of a number of quantitative studies and experiments that preserving the aspect-oriented paradigm throughout the system development stages offers several benefits compared to shift from aspect-oriented models to object-oriented code. Thus we can say that the ultimate integration of an aspect-oriented modeling language with model-driven engineering would be to take a model developed using that language and automatically generate code from it into an aspect-oriented programming language. As revealed by results of this mapping study, however, very few proposals in existing literature address the specific problem of aspect-oriented code generation. All of these solution proposals are limited to generating only partial code. Moreover, none of the approaches has reported code generation from diagrams that present behavioral perspective of the system such as UML sequence and statechart diagrams. Therefore, this is another area that needs to be investigated further in order to get fruitful results from integration of aspect-oriented models and the code generation techniques.

6.2. Limitations and threats to validity of the study

In general, the most significant threats to validity of a mapping study are related to the selection bias, inaccuracy in data extraction process [112], and misclassification of studies [37]. Selection bias refers to the possibility of a publication being incorrectly included or excluded from the mapping process. To address this threat, we explicitly defined the inclusion and exclusion criteria. On the one hand, it helped us obtain the largest possible number of publications, and on the other hand it reduced the impact of grey literature and unpublished results on the mapping. Selection bias is sometimes result of the lack of standard language and terminology in the software engineering domain [112]. For this reason, an iterative process was used to identify the keywords for search process. Synonyms were carefully applied to determine the equivalent alternatives to be included in the list of keywords.

Inaccuracy in data extraction refers to the problem that information from a paper may be extracted in different ways by different reviewers. This could result in risk of authors introducing bias during the data extraction process. In order to alleviate this risk, data extraction was based only on the information found in each publication, that is, we did not consider other possibilities assumed in publications. Thus, for example, if a paper actually presented a prototypical examination of a specific method and claimed that the same could simply be adapted to examine it in an industrial setting, we have not classified it as "Evaluation research". Misclassification is a problem that is actually linked with inaccuracy in data extraction. In order to minimize this threat, classification of each paper was conducted by both authors individually and differences noticed in few cases were resolved through consensus.

Apart from significant threats discussed above, there is another one that is specific to the code generation perspective addressed in this mapping study. In particular, this work presents aspect-oriented code generation approaches that were obtained through research sources and peer reviewed publications only. This search process may have excluded other (possibly commercial) code generation tools, as not all such tools are represented in research literature. However, it is hoped that this possible exclusion will not have significant effect on validity of results. This is because such tools usually take the process of code generation as a software development activity and contribute little to research.

7. Conclusion

Research in the area of aspect-oriented modeling and model-driven code generation can result in significant advancement in development of software systems that are more maintainable, extensible and reusable. To get an overall view of the current research in this area, we defined a few research questions and launched a systematic mapping study. We found 65 publications that possess maximum relevance to fulfill objectives of our study. The selected papers have appeared between 2002 and 2011.

The results of this study indicate that aspect-oriented modeling and code generation is a rather underdeveloped area. The initial significant contributions to this area were presented in 2002 (i.e. [55,56]), most papers have appeared in workshops and conferences, while a few were reported in journals.

Aspect-oriented modeling has acquired reasonably established understanding about the modeling notations, composition mechanisms and management of complex interactions among models. However, if we examine the systematic map, we mostly find solution proposals. There are only a few research efforts that have actually used and evaluated these proposals.

Aspect-oriented model-driven code generation, on the other hand, is indicated being relatively more immature area. There are very few publications that address specifically this area; and only two achieved the maturity of a journal publication. The mapping shows that no work has been reported in literature that uses or evaluates any of the solution proposals.

As far as the answer to our first research question is concerned, main research topics identified are: (1) modeling notations and process, (2) model composition and interaction management, (3) aspect-oriented code generation, (4) code generation from specification of non-functional requirements, and (5) applicability of aspect-oriented modeling and code generation approaches. To respond to our second research question, we have determined that most research has appeared in conferences (52%) and workshops (21%). Relatively fewer publications (27%) have appeared in journals so far. As far as answer to our third question is concerned, research is mostly of solution proposal type (66%), some conceptual proposals have been presented (13%), while evaluation and validation appear being minor groups as a whole (12%).

References

[1] B. Karakostas, Y. Zorgios, Engineering Service Oriented Systems: A Model Driven Approach, IGI Global, 2008.
[2] M. Afonso, R. Vogel, J. Teixeira, From code centric to model centric software engineering: practical case study of MDD infusion in a systems integration company, in: Fourth and Third International Workshop on Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006, MBD/MOMPES 2006, 2006, p. 10, p. 134.
[3] A. Rashid, A. Moreira, J. Araujo, P. Clements, E. Baniassad, B. Tekinerdogan, Early aspects: aspect-oriented requirements engineering and architecture design, in: Electronic Document, 2006 <http://www.early-aspects.net/>.
[4] T. Elrad, O. Aldawud, A. Bader, in: D. Batory, C. Consel, W. Taha (Eds.), Aspect-Oriented Modeling: Bridging the Gap between Implementation and Design Generative Programming and Component Engineering, Springer, Berlin/Heidelberg, 2002, pp. 189–201.
[5] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin, Aspect-oriented programming, in: M. Aksit, S. Matsuoka (Eds.), ECOOP'97 – Object-Oriented Programming, Springer, Berlin/Heidelberg, 1997, pp. 220–242.
[6] S. Op de beeck, E. Truyen, N. Bouck'e, F. Sanen, M. Bynens, W. Joosen, A Study of Aspect-Oriented Design Approaches, Department of Computer Science K.U. Leuven, 2006.
[7] M. Wimmer, A. Schauerhuber, G. Kappel, W. Retschitzegger, W. Schwinger, E. Kapsammer, A survey on UML-based aspect-oriented design modeling, ACM Computing and Survey 43 (2011) 1–33.
[8] J. Mukerji, J. Miller, MDA Guide Version 1.0.1, Object Management Group, 2003 <http://www.omg.org>.
[9] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Keele University, University of Durham: EBSE Technical, Report EBSE-2007-01, 2007.
[10] F. Brooks, The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Pub. Co, 1995.
[11] O. Group, OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2, 2007.
[12] G. Booch, Object-Oriented Analysis and Design with Applications, second ed., Addison-Wesley Professional, 1993.
[13] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorenson, Object-Oriented Modeling and Design, Prentice Hall Inc., 1991.
[14] I. Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley Professional, 1992.
[15] A. Hovsepyan, S. Van Baelen, B. Vanhooff, W. Joosen, Y. Berbers, in: S. Vassiliadis, S. Wong, T. Hämäläinen (Eds.), Key Research Challenges for Successfully Applying MDD Within Real-Time Embedded Software Development Embedded Computer Systems: Architectures, Modeling, and Simulation, Springer, Berlin/Heidelberg, 2006, pp. 49–58.
[16] S. Philippi, Automatic code generation from high-level Petri-Nets for model driven systems engineering, Journal of Systems and Software 79 (2006) 1444–1455.
[17] Y. Rauchwerger, F. Kristoffersen, Y. Lahav, Cinderella SLIPPER: An SDL to C-Code Generator, in: A. Prinz, R. Reed, J. Reed (Eds.), SDL 2005: Model Driven, Springer, Berlin/Heidelberg, 2005, pp. 1159–1165.
[18] A. Stavrou, G.A. Papadopoulos, Automatic generation of executable code from software architecture models, in: Information Systems Development, Springer, US, 2009, pp. 447–458.
[19] R. Pilitowski, A. Dereziñska, Code generation and execution framework for UML 2.0 classes and state machines, in: T. Sobh (Ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, Springer, Netherlands, 2007, pp. 421–427.
[20] F. Chauvel, J.-M. Jézéquel, Code generation from UML models with semantic variation points, in: L. Briand, C. Williams (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2005, pp. 54–68.
[21] I.A. Niaz, J. Tanaka, An object-oriented approach to generate java code from UML statecharts, International Journal of Computer & Information Science 6 (2005).
[22] D. Leroux, M. Nally, K. Hussey, Rational software architect: a tool for domain-specific modeling, IBM Systems Journal 45 (2006) 555–568.
[23] AjileJ, AjileJ StructureViews, 2011 <www.ajilej.com>.
[24] NoMagic, MagicDraw UML, 2011 <www.magicdraw.com/>.
[25] D.M. Simmonds, Aspect-oriented approaches to model driven engineering, in: International Conference on Software Engineering Research and Practice, Las Vegas, Nevada, USA, 2008.
[26] A. Hovsepyan, R. Scandariato, S.V. Baelen, Y. Berbers, W. Joosen, From aspect-oriented models to aspect-oriented code? the maintenance perspective, in: Proceedings of the 9th International Conference on Aspect-Oriented Software Development, ACM, Rennes and Saint-Malo, France, 2010, pp. 85–96.
[27] D.M. Simmonds, Y.R. Reddy, A comparison of aspect-oriented approaches to model driven engineering, in: Conference on Software Engineering Research, and Practice, 2009, pp. 327–333.
[28] W. Harrison, H. Ossher, P. Tarr, Asymmetrically vs. symmetrically organized paradigms for software composition, 2002.
[29] J. Hannemann, G. Kiczales, Design pattern implementation in Java and aspectJ, SIGPLAN Notices 37 (2002) 161–173.
[30] A. Garcia, C. Sant'Anna, E. Figueiredo, U. Kulesza, C. Lucena, A.v. Staa, Modularizing design patterns with aspects: a quantitative study, in: Proceedings of the 4th International Conference on Aspect-Oriented Software Development, ACM, Chicago, Illinois, 2005, pp. 3–14.
[31] L. Fuentes, P. Sánchez, Execution of aspect oriented UML models, in: D. Akehurst, R. Vogel, R. Paige (Eds.), Model Driven Architecture-Foundations and Applications, Springer, Berlin/Heidelberg, 2007, pp. 83–98.
[32] N. Cacho, C. Sant'Anna, E. Figueiredo, A. Garcia, T. Batista, C. Lucena, Composing design patterns: a scalability study of aspect-oriented programming, in: Proceedings of the 5th International Conference on Aspect-Oriented Software Development, ACM, Bonn, Germany, 2006, pp. 109–121.
[33] A. Mehmood, D.N.A. Jawawi, A comparative survey of aspect-oriented code generation approaches, in: 5th Malaysian Conference in Software Engineering (MySEC), 2011, 2011, pp. 147–152.
[34] R. Chitchyan, A. Rashid, P. Sawyer, A. Garcia, M.P. Alarcon, J. Bakker, B. Tekinerdogan, S. Clarke, A. Jackson, Survey of Aspect-Oriented Analysis and Design Approaches, Technical Report AOSD, Europe Deliverable D11, AOSD-Europe-ULANC-9, Lancaster University, May 2005.
[35] A.M. Reina, J. Torres, M. Toro, Separating concerns by means of UML-profiles and metamodels in PIMs, in: O. Aldawud, G. Booch, J. Gray, J.o. Kienzle, D. Stein, Mohamed, F. Akkawi, T. Elrad (Eds.), The 5th Aspect-Oriented Modeling Workshop in Conjunction with UML 2004, 2004.
[36] T. Dybå, B.A. Kitchenham, M. Jørgensen, Evidence-based software engineering for practitioners, IEEE Software 22 (2005) 58–65.
[37] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evalulation and Assessment in Software Engineering (EASE), University of Bari, Italy, 2008. June.
[38] M. Shaw, What makes good research in software engineering?, International Journal of Software Tools for Technology Transfer (STTT) 4 (2002) 1–10
[39] L. Fuentes, P. Sanchez, Designing and weaving aspect-oriented executable UML models, Journal of Object Technology 6 (2007) 109–136.
[40] R. Ballal, M.A. Hoffman, Extending UML for aspect oriented software modeling, in: WRI World Congress on Computer Science and Information Engineering, 2009, 2009, pp. 488–492.
[41] J. Bennett, K. Cooper, L. Dai, Aspect-oriented model-driven skeleton code generation: a graph-based transformation approach, Science of Computer Programming 75 (2010) 689–725.
[42] I. Groher, S. Schulze, Generating aspect code from UML models, in: The Third International Workshop on Aspect-Oriented Modeling, 2003.
[43] J. Whittle, P. Jayaraman, MATA: a tool for aspect-oriented modeling based on graph transformation, in: H. Giese (Ed.), Models in Software Engineering, Springer, Berlin/Heidelberg, 2008, pp. 16–27.
[44] S. Clarke, E. Baniassad, Aspect-Oriented Analysis and Design: The Theme Approach, Addison Wesley Object Technology, 2005.
[45] J. Evermann, A meta-level specification and profile for AspectJ in UML, in: Proceedings of the 10th International Workshop on Aspect-Oriented Modeling, ACM, Vancouver, Canada, 2007, pp. 21–27.
[46] L. Fuentes, P. Sanchez, A generic MOF metamodel for aspect-oriented modelling, in: Fourth and Third International Workshop on Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006, MBD/MOMPES 2006, 2006, p. 10, 124.
[47] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requirements Engineering 11 (2005) 102–107.
[48] A.A. Zakaria, H. Hosny, A. Zeid, A UML extension for modeling aspect-oriented systems, in: Fifth International Conference on the Unified Modeling Language – The Language and its Applications, 2002.
[49] T. Aldawud, A. Bader, Tzilla Elrad, UML profile for aspect-oriented software development, in: The Third International Workshop on Aspect Oriented Modeling, 2003.
[50] R. Pitkänen, P. Selonen, A UML profile for executable and incremental specification-level modeling, in: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), UML 2004 –- The Unified Modelling Language, Springer, Berlin/Heidelberg, 2004, pp. 158–172.
[51] T. Cottenier, A.v.d. Berg, T. Elrad, Motorola WEAVR: aspect orientation and model-driven engineering, Journal of Object Technology 6 (2007) 51–88.
[52] M. Mosconi, A. Charfi, J. Svacina, J. Wloka, Applying and evaluating AOM for platform independent behavioral UML models, in: Proceedings of the 12th Workshop on Aspect-Oriented Modeling, ACM, Brussels, Belgium, 2008, pp. 19–24.

[53] R. Wang, X.-G. Mao, Z.-Y. Dai, Y.-N. Wang, Extending UML for aspect-oriented architecture modeling, in: Second International Workshop on Computer Science and Engineering, 2009, WCSE '09, 2009, pp. 362–366.
[54] D. Mouheb, C. Talhi, M. Nouh, V. Lima, M. Debbabi, L. Wang, M. Pourzandi, Aspect-oriented modeling for representing and integrating security concerns in UML, in: R. Lee, O. Ormandjieva, A. Abran, C. Constantinides (Eds.), Software Engineering Research, Management and Applications 2010, Springer, Berlin/Heidelberg, 2010, pp. 197–213.
[55] D. Stein, S. Hanenberg, R. Unland, An UML-based aspect-oriented design notation for AspectJ, in: Proceedings of the 1st International Conference on Aspect-Oriented Software Development, ACM, Enschede, The Netherlands, 2002, pp. 106–112.
[56] R. Pawlak, L. Duchien, G. Florin, F. Legond-Aubry, L. Seinturier, L. Martelli, A UML notation for aspect-oriented software design, in: AO Modeling with UML Workshop at the AOSD'02, 2002.
[57] Z. Jingjun, C. Yuejuan, L. Guangyuan, Modeling aspect-oriented programming with UML profile, in: First International Workshop on Education Technology and Computer Science, 2009, ETCS '09, 2009, pp. 242–245.
[58] J. Klein, J. Kienzle, Reusable aspect models, in: 11th Workshop on Aspect-Oriented Modeling, Nashville, TN, USA, 2007.
[59] Y. Reddy, S. Ghosh, R. France, G. Straw, J. Bieman, N. McEachen, E. Song, G. Georg, Directives for composing aspect-oriented design class models, in: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development I, Springer, Berlin/Heidelberg, 2006, pp. 75–105.
[60] J. Klein, L. Helouet, J.-M. Jezequel, Semantic-based weaving of scenarios, in: Proceedings of the 5th International Conference on Aspect-Oriented Software Development, ACM, Bonn, Germany, 2006, pp. 27–38.
[61] J. Kienzle, W.A. Abed, J. Klein, Aspect-oriented multi-view modeling, in: Proceedings of the 8th ACM International Conference on Aspect-Oriented Software Development, ACM, Charlottesville, Virginia, USA, 2009, pp. 87–98.
[62] W. Al Abed, J. Kienzle, Aspect-oriented modelling for distributed systems, in: J. Whittle, T. Clark, T. Kühne (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2011, pp. 123–137.
[63] G. Zhang, Towards aspect-oriented class diagrams, in: 12th Asia–Pacific, 2005 Software Engineering Conference, 2005, APSEC '05, p. 6.
[64] G. Zhang, M. Hölzl, A. Knapp, Enhancing UML state machines with aspects, in: G. Engels, B. Opdyke, D. Schmidt, F. Weil (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2007, pp. 529–543.
[65] Z. Xiao-Cong, L. Chang, N. Yan-Tao, L. Tai-Zong, Towards a Framework of Aspect-Oriented Modeling with UML, in: International Symposium on Computer Science and Computational Technology, 2008, ISCSCT '08, 2008, pp. 738–741.
[66] P.S.a. Lidia Fuentes, Elaborating UML 2.0 profiles for AO design, in: 8th Workshop on AOM, 5th International Conference on AOSD, Germany, 2006.
[67] A.M. Francisca Losavio, Patricia Morantes, UML extensions for aspect oriented software development, Journal of Object Technology 8 (2009) 105–132.
[68] N. Amálio, P. Kelsen, Q. Ma, C. Glodt, Using VCL as an aspect-oriented approach to requirements modelling, in: S. Katz, M. Mezini, J. Kienzle (Eds.), Transactions on Aspect-Oriented Software Development VII, Springer, Berlin/Heidelberg, 2010, pp. 151–199.
[69] P. Sánchez, L. Fuentes, D. Stein, S. Hanenberg, R. Unland, Aspect-oriented model weaving beyond model composition and model transformation, in: K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, M. Völter (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2008, pp. 766–781.
[70] D. Stein, S. Hanenberg, R. Unland, Expressing different conceptual models of join point selections in aspect-oriented design, in: Proceedings of the 5th International Conference on Aspect-Oriented Software Development, ACM, Bonn, Germany, 2006, pp. 15–26.
[71] O. Barais, J. Klein, B. Baudry, A. Jackson, S. Clarke, Composing Multi-view Aspect Models, in: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), IEEE Computer Society, 2008, pp. 43-52.
[72] J.K. Andrew Jackson, Benoit Baudry, Siobhán Clarke, KerTheme: Testing aspect oriented models, in: Workshop on Integration of Model Driven Development and Model Driven Testing (ECMDA'06), Bilbao, Spain, 2006.
[73] J. White, J. Gray, D. Schmidt, Constraint-based model weaving, in: S. Katz, H. Ossher, R. France, J.-M. Jézéquel (Eds.), Transactions on Aspect-Oriented Software Development VI, Springer, Berlin/Heidelberg, 2009, pp. 153–190.
[74] J. Cohen, Constraint logic programming languages, Communications of the ACM 33 (1990) 52–68.
[75] S. Hanenberg, R. Hirschfeld, R. Unland, Morphing aspects: incompletely woven aspects and continuous weaving, in: Proceedings of the 3rd International Conference on Aspect-Oriented Software Development, ACM, Lancaster, UK, 2004, pp. 46–55.
[76] W.-M. Ho, J.-M. Jezequel, F. Pennaneac'h, N. Plouzeau, A toolkit for weaving aspect oriented UML designs, in: Proceedings of the 1st International Conference on Aspect-Oriented Software Development, ACM, Enschede, The Netherlands, 2002, pp. 99–105.
[77] I. Groher, M. Voelter, XWeave: models and aspects in concert, in: Proceedings of the 10th International Workshop on Aspect-Oriented Modeling, ACM, Vancouver, Canada, 2007, pp. 35–40.
[78] J. Klein, F. Fleurey, J.-M. Jézéquel, Weaving multiple aspects in sequence diagrams, in: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development III, Springer, Berlin/Heidelberg, 2007, pp. 167–199.
[79] R. Grønmo, F. Sørensen, B. Møller-Pedersen, S. Krogdahl, Semantics-based weaving of UML sequence diagrams, in: A. Vallecillo, J. Gray, A. Pierantonio (Eds.), Theory and Practice of Model Transformations, Springer, Berlin/Heidelberg, 2008, pp. 122–136.
[80] J. Klein, J. Kienzle, B. Morin, J.-M. Jézéquel, Aspect model unweaving, in: A. Schürr, B. Selic (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2009, pp. 514–530.
[81] L. Fuentes, P. Sánchez, Dynamic weaving of aspect-oriented executable UML models, in: S. Katz, H. Ossher, R. France, J.-M. Jézéquel (Eds.), Transactions on Aspect-Oriented Software Development VI, Springer, Berlin/Heidelberg, 2009, pp. 1–38.
[82] K. Mehner, M. Monga, G. Taentzer, Analysis of aspect-oriented model weaving, in: A. Rashid, H. Ossher (Eds.), Transactions on Aspect-Oriented Software Development V, Springer, Berlin/Heidelberg, 2009, pp. 235–263.
[83] S. Ciraci, W. Havinga, M. Aksit, C. Bockisch, P. van den Broek, A graph-based aspect interference detection approach for UML-based aspect-oriented models, in: S. Katz, M. Mezini, J. Kienzle (Eds.), Transactions on Aspect-Oriented Software Development VII, Springer, Berlin/Heidelberg, 2010, pp. 321–374.
[84] J. Whittle, P. Jayaraman, A. Elkhodary, A. Moreira, J. Araújo, MATA: a unified approach for composing UML aspect models based on graph transformation, in: S. Katz, H. Ossher, R. France, J.-M. Jézéquel (Eds.), Transactions on Aspect-Oriented Software Development VI, Springer, Berlin/Heidelberg, 2009, pp. 191–237.
[85] B. Morin, G. Vanwormhoudt, P. Lahire, A. Gaignard, O. Barais, J.-M. Jézéquel, Managing variability complexity in aspect-oriented modeling, in: K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, M. Völter (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2008, pp. 797–812.
[86] J.-M. Jézéquel, Model driven design and aspect weaving, Software and Systems Modeling 7 (2008) 209–218.
[87] J. Oldevik, M. Menarini, I. Krüger, Model composition contracts, in: A. Schürr, B. Selic (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2009, pp. 531–545.
[88] B. Demchak, V. Ermagan, E. Farcas, T.-J. Huang, I.H. Kruger, M. Menarini, A rich services approach to CoCoME, in: R. Andreas, R. Ralf, M. Raffaela, Franti, P. ek, il (Eds.), The Common Component Modeling Example, Springer-Verlag, 2008, pp. 85–115.
[89] M.V. Hecht, E.K. Piveta, M.S. Pimenta, R.T. Price, Aspect-oriented Code Generation, in: XX Brazilian Conference on, Software Engineering, 2005.
[90] S. Maoz, D. Harel, From multi-modal scenarios to code: compiling LSCs into aspectJ, in: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, Portland, Oregon, USA, 2006, pp. 219–230.
[91] M. Kramer, J. Kienzle, Mapping aspect-oriented models to aspect-oriented code, in: J. Dingel, A. Solberg (Eds.), Models in Software Engineering, Springer, Berlin/Heidelberg, 2011, pp. 125–139.
[92] S. Haitao, S. Zhumei, Z. Shixiong, Mapping aspect-oriented domain-specific model to code for real time system, in: The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006, 2006, pp. 6426–6431.
[93] S. Hanenberg, D. Stein, R. Unland, From aspect-oriented design to aspect-oriented programs: tool-supported translation of JPDDs into code, in: Proceedings of the 6th International Conference on Aspect-Oriented Software Development, ACM, Vancouver, British Columbia, Canada, 2007, pp. 49–62.
[94] L. Dai, (Defense) Formal Design Analysis Framework: An Aspect-Oriented Architectural Framework, University of Texas at Dallas, Ph.D. Dissertation, 2005.
[95] F. Alhalabi, P. Vienne, M. Maranzana, J.L. Sourrouille, Code Generation from the Description of QoS-Aware Applications, in: 2nd Information and Communication Technologies, 2006. ICTTA '06, 2006, pp. 3216–3221.
[96] C. Buckl, M. Regensburger, A. Knoll, i.G. Schrott, Models for automatic generation of safety-critical real-time systems, in: Second International Conference on Availability, Reliability and Security '07, Vienna, Austria, 2007, pp. 580–587.
[97] C. Cetina, E. Serral, J. Muñoz, V. Pelechano, Tool support for model driven development of pervasive systems, in: Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES '07), Portugal, 2007, pp. 33–44.
[98] M.A. Wehrmeister, E.P. Freitas, C.E. Pereira, F. Rammig, GenERTiCA: A tool for code generation and aspects weaving, in: 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008, 2008, pp. 234–238.
[99] A. Carton, C. Driver, A. Jackson, S. Clarke, Model-driven theme/UML, in: S. Katz, H. Ossher, R. France, J.-M. Jézéquel (Eds.), Transactions on Aspect-Oriented Software Development VI, Springer, Berlin/Heidelberg, 2009, pp. 238–266.
[100] J. Kienzle, W. Al Abed, F. Fleurey, J.-M. Jézéquel, J. Klein, Aspect-oriented design with reusable aspect models, in: S. Katz, M. Mezini, J. Kienzle (Eds.), Transactions on Aspect-Oriented Software Development VII, Springer, Berlin/Heidelberg, 2010, pp. 272–320.
[101] J. Kienzle, N. Guelfi, S. Mustafiz, Crisis management systems: a case study for aspect-oriented modeling, in: S. Katz, M. Mezini, J. Kienzle (Eds.), Transactions on Aspect-Oriented Software Development VII, Springer, Berlin/Heidelberg, 2010, pp. 1–22.

[102] M. Hölzl, A. Knapp, G. Zhang, Modeling the car crash crisis management system using HiLA, in: S. Katz, M. Mezini, J. Kienzle (Eds.), Transactions on Aspect-Oriented Software Development VII, Springer, Berlin/Heidelberg, 2010, pp. 234–271.

[103] M. Wehrmeister, E. Freitas, C. Pereira, An infrastructure for UML-based code generation tools, in: A. Rettberg, M. Zanella, M. Amann, M. Keckeisen, F. Rammig (Eds.), Analysis, Architectures and Modelling of Embedded Systems, Springer, Boston, 2009, pp. 32–43.

[104] J. Gray, S. Roychoudhury, A technique for constructing aspect weavers using a program transformation engine, in: Proceedings of the 3rd International Conference on Aspect-Oriented Software Development, ACM, Lancaster, UK, 2004, pp. 36–45.

[105] S. Clarke, R.J. Walker, Towards a standard design language for AOSD, in: Proceedings of the 1st International Conference on Aspect-Oriented Software Development, ACM, Enschede, The Netherlands, 2002, pp. 113–119.

[106] I. Krechetov, B. Tekinerdogan, A. Garcia, C. Chavez, U. Kulesza, Towards an integrated aspect-oriented modeling approach for software architecture design, in: 8th Workshop on Aspect-Oriented Modelling (AOM06), AOSD06, Germany, 2006.

[107] N. Albunni, M. Petridis, Using UML for Modelling Cross-cutting concerns in aspect oriented software engineering, in: 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008, ICTTA 2008, 2008, pp. 1–6.

[108] N. Debnath, L. Baigorria, D. Riesco, G. Montejano, Metrics applied to Aspect Oriented Design using UML profiles, in: IEEE Symposium on Computers and Communications, 2008. ISCC 2008, 2008, pp. 654–657.

[109] B. Morin, F. Fleurey, N. Bencomo, J.-M. Jézéquel, A. Solberg, V. Dehlen, G. Blair, An aspect-oriented and model-driven approach for managing dynamic variability, in: K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl, M. Völter (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2008, pp. 782–796.

[110] J. Fabry, A. Zambrano, S. Gordillo, Expressing aspectual interactions in design: experiences in the slot machine domain, in: J. Whittle, T. Clark, T. Kühne (Eds.), Model Driven Engineering Languages and Systems, Springer, Berlin/Heidelberg, 2011, pp. 93–107.

[111] D. Xu, W. Xu, State-based incremental testing of aspect-oriented programs, in: Proceedings of the 5th International Conference on Aspect-Oriented Software Development, ACM, Bonn, Germany, 2006, pp. 180–189.

[112] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Information and Software Technology 50 (2008) 833–859.