

Towards a Meta Aspect for Traceability

Marta S. Tabares
Escuela de Sistemas,
Universidad Nacional de Colombia
Calle 59A No 63 – 20 Medellin, Colombia
mstabare@unalmed.edu.co

Ana Moreira
CITI/Departamento de Informática,
FCT/Universidade Nova de Lisboa
2829-516 Caparica, Portugal
amm@di.fct.unl.pt

Abstract. *Crosscutting concerns are represented in different ways using different artefacts throughout the life cycle. The evolution of crosscutting concerns must be controlled to guarantee the consistency of requirements and to avoid costly rework. This control capability is, in essence, traceability.*

This paper describes an initial general vision of how traceability of crosscutting concerns can be supported and controlled by means of a Meta Aspect for Traceability. This permits us to explore semantic consistency and completeness as basic characteristics to trace crosscutting concerns. Currently, each existing approach provides a set of elements with which we can conform a general semantics with formal validation properties that support and control the traceability of crosscutting concerns at the early stages of the software development process.

1. Introduction

Aspect-oriented software development (AOSD) is focusing efforts to provide different techniques for early aspects to help in identifying, modularizing, composing and tracing crosscutting concerns both for requirements engineering and for architecture design [1].

Traceability is specifically mentioned in each of the referred approaches utilizing some elements of the model. For example, [2, 3, 4] provide traceable elements through the identification of the influence of multiple concerns in the system as well as tables to support cross-reference between concerns, trade-off analysis, conflict solutions among stakeholders and mapping elements to architecture design. Clark et al. present a way to model and design concerns from requirements analysis to implementation [5, 6]. Their approach provides a lexical model (Theme/DOC) for the analysis of requirements. It permits separating functionality into action views that represent the requirements documentation, and are then refined into major action views as elements to represent traceability in the design phase [5]. Next it is possible to obtain

design artefacts encapsulated in units called “themes” (Theme/UML), and the crosscutting concern themes than can be traceable by means of a composition pattern [6].

Ferreira et al. present a template (based on that proposed by [18]) to describe concerns aiming at supporting traceability by offering a simple mechanism to control concerns’ versions and their evolution throughout the development lifecycle [13].

In general, all the above mentioned approaches offer basic elements to support traceability, but they do not provide a general semantics for traceability for the various activities of the lifecycle nor do they use the semantic consistency and completeness as traceability properties.

The aim of our work is to develop a general metamodel to support backward and forward traceability of crosscutting concerns within a lifecycle phase or between different phases. This metamodel should provide the necessary semantics to maintain and control the tracing through formal specifications to guarantee completeness and semantic consistency. We propose a Meta Aspect for Traceability (MAT) to improve the quality of crosscutting concerns by enhancing their correctness. This paper presents our initial ideas on MAT.

The structure of this paper is as follows. Section 2 presents some background work. Section 3 presents the identification of the problem. Section 4 describes the MAT approach, which reflects our vision to manage traceability for crosscutting concerns from a meta-level perspective. Section 5 discusses some related work. Finally, Section 6 concludes the paper and lists open issues and directions for future work.

2. Basic Concepts

2.1 Metamodels

A metamodel is a precise definition of the constructs and rules needed for creating semantic models. It provides the basic information that describes elements, structure, relationships, content and quality of a model.

Moreover, it offers a description of other models or modelling languages, its constructs, and syntax rules [20, 21, 22] that can then be instantiated under any model.

The elementary modelling primitives of the metamodel are defined as:

- *Nodes*: corresponding to constructs that may be present in a model independently of any other construct; for example, E-R model entities may exist without requiring the presence of any other particular constructs.
- *Edges*: corresponding to constructs that can only exist in a model when certain other constructs exist; for example, relationships in the E-R model cannot exist without entities.
- *Constraints*: representing restrictions on the constructs of the model we need to represent through the metamodel; for example, E-R generalization hierarchies or cardinality constraints of E-R relationships and attributes.
- *Attributes*: representing the possible characteristics that each node or edge can have. An attribute can be defined by a name, type, size, etc.

The MOF (Meta Object Facility), an OMG standard, is a well-known metamodeling technique that offers a framework for metadata management with a set of services (these are defined by the Ontology Definition Metamodel). “MOF is an extensible model driven integration framework for defining, manipulating and integrating metadata and data in a platform independent manner” [23].

2.2 Semantic consistency

Gurevych et al. say that semantic consistency is “the hypothesis of recognition of the speech” correctly structured at the abstract semantic level [14].

Semantic consistency expressions are used to validate three levels of consistency: *full consistency*, *semi-consistency* and *inconsistency*. Full consistency occurs when all the concepts are related between them. Semi-consistency occurs when part of the concepts are related to the original hypothesis (or problem). Inconsistency means that the conceptual maps are not semantically related within the domain model.

2.3 Completeness

Completeness can be seen from different perspectives, in particular from a programming and a logic dimension. For example, from a programming perspective, we can define computational completeness

as any computable function. Colomb and Weber present a theorem that expresses this definition [15]:

If the completeness condition is satisfied, then for every partial computable function $f : \mathcal{O}_1 \rightarrow \mathcal{O}_2$ (where \mathcal{O} is a set of objects) there is a program that computes f .

Also they define logic completeness, as a property associated with combining a procedure to build well-formed formulas, a definition of truth that relates two interpretations of models of logical systems, and a proof procedure that allows new well-formed formulas to be derived from old well-defined formulas.

3. Problem Identification

To properly identify the problem in hand, we have studied and evaluated some early aspects approaches [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 25, 26, 27, 28, 29, 30] summarized in [1] to identify missing elements to support a general semantic for traceability and to validate semantic consistency and completeness. The result is shown in Table 1. The characteristics we have considered for our evaluation were those defined in [1], in particular: *traceability through the lifecycle*, *mapping* support and *evolvability*.

Based on this study, we have identified three relevant factors that could raise difficulties and disagreement when more than one traceability model is used:

- In the early aspects framework, the set of tracing elements that each approach offers is different; these can be templates, cross-referenced tables, use cases, XML requirements descriptions, etc. The traceability semantics is interpreted according to each domain element provided by the approach.
- Usually, the semantic consistency property is not used to control traceability in the same phase or between phases.
- Completeness is not handled properly by any of the existing approaches. However, El-Maddah and Maibaum [11] address this problem in the requirements phase.

The above discussion illustrates the need for a formal framework to support traceability. This framework should handle completeness and semantic consistency for crosscutting concerns through the lifecycle. Ideally, the framework should be independent of any particular approach or specific implementation platform.

Table 1. Evaluation of some early aspects approaches, showing contributions marked as ⊕) and deficiencies (marked as ⊖) w.r.t. general semantic traceability, completeness and semantic consistency.

Approach / Criteria	Evaluation summary
1. AORE with Arcade & PROBE [2, 25]	<ul style="list-style-type: none"> ⊕ Provides and XML semantics to express composition and also temporal logic semantics to validate the consistency of the composition rule, and to extend the aspects ontology with those temporal logic rules. ⊖ There is no explicit rule to help deciding the correct mapping elements: decision, a function or an aspect. Moreover, there is no approach at the design level to interpret these elements in the new phase, so that traceability between phases can be supported. ⊖ PROBE offers a mapping between phases; however it does not show how to do this explicitly. ⊖ It does not offer backward traceability. ⊖ It does not support completeness a semantic consistency.
2. Aspects in Requirements Goal Models (ARGM) [12]	<ul style="list-style-type: none"> ⊕ Provides semantic consistency between NFRs inherited from KAOS. ⊖ Despite the formal semantics to handle goals and requirements offered by KAOS, it does not yet address crosscutting concerns. ⊖ It provides elements for traceability, but there isn't yet a rigorous semantics defined. ⊖ It provides elements for traceability, but there isn't yet a rigorous semantics defined. ⊖ It does not support completeness a semantic consistency.
3. AOSD/UC [27], Scenario Modelling with Aspects (SMA) [28], Aspectual Use Case Driven Approach Process (AUCDA) [26, 27]	<ul style="list-style-type: none"> ⊕ The consistency semantics between UML artefacts is defined in OCL. ⊕ The extension points, slice use cases and use case modules offer semantics to build the tracing in a given phase or between phases. (We believe this is the richer approach to support traceability.) ⊖ It does not support completeness a semantic consistency.
4. COSMOS [29]	<ul style="list-style-type: none"> ⊕ It offers a distinction between concerns. This can be taken as reference elements to support traceability. ⊖ It does not offer traceability semantics. ⊖ It does not support completeness a semantic consistency.
5. Concern Oriented Requirements Engineering (CORE) [3, 26]	(What has been said for number one, applies here.)
6. Aspect Oriented Requirements Engineering for components (AOREC) [30]	<ul style="list-style-type: none"> ⊕ Provides a semantics that can be used to support traceability; this is based on the components requirements specification. ⊖ It does not offer traceability semantics. ⊖ It does not support completeness a semantic consistency.
7. Theme/DOC, Theme/UML [5, 6]	<ul style="list-style-type: none"> ⊕ It guarantees traceability by offering automatic mapping between artefacts of the different lifecycle phases. ⊖ It does not support completeness a semantic consistency.

4. MAT: The Vision

The aim of the *Meta Aspect for Traceability* is to offer a framework to provide a general semantics for traceability. This framework should cover the domain elements as well as the analysis and design actions that each early aspect approach defines for the tracing of elements. The

semantics should guarantee completeness and consistency semantics as basic formal validation properties.

The *Meta Aspect for Traceability* consists of three packages: *Tracing Controller*, *Crosscutting Concerns' Models* and *Formal Primitives*. These packages provide domain semantics and its elementary modelling primitives to maintain and control the traceability of crosscutting concerns. The relationships between these packages are

shown in Figure 1. (We have used UML to represent the MAT main packages [24]).

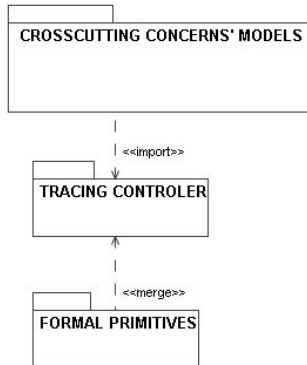


Figure 1. Meta Aspect for Traceability – MAT.

The *Tracing Controller* package acts as a monitor with its own semantics to indicate where and when the tracing of a crosscutting concern must be validated with respect to completeness and semantic consistency. This package weaves the necessary traceability control semantics to the elements of crosscutting concern models.

The *Crosscutting Concerns' Models* package acts as a provider of semantics of artefacts, relationships and rules that existing models of crosscutting concerns in analysis, design and traceability activities. These can provide the refinement (the direction of tracing), mapping and trade-offs of the crosscutting concerns and other elements related to them.

The *Formal Primitives* package introduces formal primitives to validate the completeness and semantic consistency in the tracing of crosscutting concerns.

The domain semantics for traceability in MAT is based on the three dimensions presented in [19]: “(1) the kinds of quantifications allowed, (2) the nature of the actions that can be asserted, and (3) the mechanism for combining base-level actions with asserted actions”.

4.1 Tracing Controller

This package is the “core” of the metamodel since it provides all the descriptions (expressions, rules, domain elements, etc.) needed to combine the semantics of Crosscutting Concerns' Models package with the semantics of the Formal Primitives package so as to maintain and control the tracing of requirements that are crosscutting concerns.

The Tracing controller is composed of two components: *Abstract Monitor of Events* and *Abstract Weaving Semantics*.

The *Abstract Monitor of Events* provides a schema for the semantic of tracing, by specifying the interface of actions associated to them. In other words, it will allow us

to specify which tracing actions and how the tracing actions interact (events, preconditions and post-condition of traceability). All with base elements provided by the crosscutting concerns' models package. Figure 2 shows the abstract monitor that defines the set of entities that are part of the domain semantics for traceability.

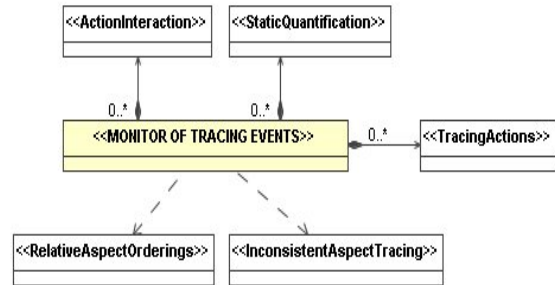


Figure 2. An Abstract Monitor of Events for Traceability

The idea proposed is an adaptation of [19]. In particular we have adapted the notion of actions, quantification, interaction, ordering and inconsistent aspects to be:

- *Tracing actions*: characterize the element models that can denote action.
- *Static quantification*: tracing elements of the models one can quantify, for instance crosscutting concerns, classes, methods, relationships, composition rules, levels, dimension, and themes.
- *Action Interaction*: determines the communication between different tracing actions. Actions on the same level and the same tracing action on different levels may need to exchange information.
- *Relative aspect orderings*: the order in which the tracing operations provided have to be invoked; in other words, it is the specification of the order of multiple tracing actions that can be applied at the same point (level, element model).
- *Inconsistent aspects tracing*: sometimes one tracing action may violate the semantics of another. Specifically, this is the element of the context model that controls the possible mismatch between tracing actions.

The *Abstract Weaving Semantics* provides the semantics necessary for intermixing the tracing actions of the two types of instantiated packages describe in Sections 4.2 and 4.3.

4.2 Crosscutting concerns' models

This package acts as a provider of the base traceability semantics from early aspect approaches. The metamodel must recognize the possible structures, behaviour, templates, ontology, constraint and others elements for

traceability, to determine elements that conform to the crosscutting concerns in different phases of the life cycle for different models (see Figure 3). This should be seen as a set of meta join points where it is possible to know the points where the semantics must be applied to validate completeness and semantic consistency.

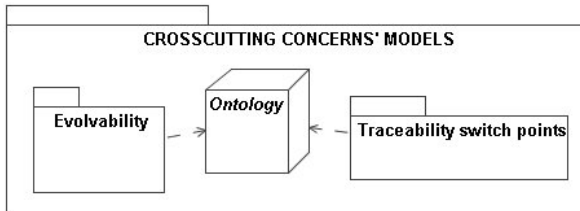


Figure 3. Crosscutting concerns' models components.

Those meta join points of tracing might have some of the following characteristics:

- Crosscutting concerns may have evolved;
- Switch points of traceability direction (vertical, horizontal) and orientation (forward, backward) may have changed;
- Special characteristics that may affect the concerns and element models associated with them (e.g., granularity level, compositions rules, components, architecture style, etc).

The semantic elements can be grouped according to the following dimensions:

- *Tracing provider*: provides the identification of the trace as well as other characteristics of the traced element (for example, location level and relationships).
- *Categories*: tracing elements are grouped according to their function in the crosscutting model.
- *Similarity*: an orthogonal classification of the elements of tracing with respect to that provided by the “categories”.
- *MatchRules*: the set of policies and rules used to match the tracing requirements against the availability.
- *Effects*: the characterization of the tracing in terms of pre- and post-conditions.
- *Purpose*: a high level description of the purpose of the tracing.
- *Way*: to define the possible ways to reach and use the tracing according to the models of traceability.

4.3 Formal primitives

By means of this package we introduce formal primitives for traceability of crosscutting concerns, so as to provide the expressions and elements to validate completeness

and semantic consistency. This package has two components: *Abstract Semantic Consistency* and *Abstract Completeness Elements*. Figure 4 illustrates the idea.

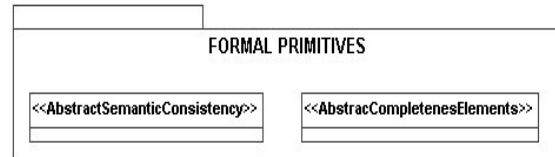


Figure 4. Formal Primitives

The *Abstract Semantic Consistency* acts as a provider of semantics, expressions and rules to validate the levels of semantic consistency of crosscutting concerns. This process evaluates how the composition/decomposition rules transform the source model and validates the form how each rule maintains the semantic consistency of the crosscutting concern through the life cycle.

In situations where it is necessary to introduce new predicates that did not form part of the initial problem, and to guarantee the consistency of the tracing of the crosscutting concerns, MAT must provide the rules to validate the semantic consistency and avoid introducing conflicting predicates.

The classification into one of these three levels will depend on an evaluation of the degree of intermix between the tracing controller package and the formal primitives package.

The semantics requires a common reference of its parts. Thus, in order to validate the semantic consistency it is necessary to consider the following factors:

- *Theoretic body*: common or universe of reference for traceability.
- *Homogeneity*: the predicates must belong to the same semantic family (semantically homogeneous with respect to meaning).
- *Semantic proximity*: it will validate that the predicates of the theory all appear in the initial assumptions and the definitions; (it excludes ad hoc additions).
- *Conceptual connection*: well-structured formulas that incorporate the predicates.

It is necessary to ensure that the expressions maintain states of consistency of the crosscutting concerns in each phase and between phases, thus avoiding the validation of possible intermediate states. The semantic consistency also refers to the integrity of crosscutting concerns between phases.

The *Abstract Completeness Elements* act as a set of statements to verify completeness of crosscutting concern instances' expressions and to perform basic operations of tracing such as declaratively stating when a trace statement requires control. These elements will act as a complementary behaviour that could be added at any time when the trace is applied. This is based on [16] that

consider that completeness relationships “can eliminate elements of arbitrary sufficiency and facilitate a more direct appreciation of the essential nature of a process expression”.

For our case, the well-formed formulas for a logical completeness, must be rules that valid the crosscutting concerns’ expressions conformed by a partial expressions of logical relationships, a partial expression of event relationships and a partial meta-coordination expression that weave the completeness logical and event expressions for some specific of crosscutting concerns.

We will consider the possibility to apply following completeness relationships:

- *Transition*: this relationship defines the values of crosscutting concerns expressions disjoint between different phases.
- *Crosscutting*: this relationship defines a set of criterion for relations between crosscutting concerns and other model elements into the same phase.
- *Participation*: this relationship reflects a general form of the “crosscutting relationship”, where the elements of the relation might not be involved in a crosscutting relationship directly.

Thus, it is necessary to ensure that the completeness of the crosscutting concerns in each phase and between phases can be validated and guaranteed through the formal rules.

5. Related Work

The approaches used for our work are still new, and therefore immature with respect to traceability support. Nevertheless, they are used as basic elements from the metamodel proposed in this paper.

Katz and Rashid provides a framework called PROBE, that presents a precise semantics for the generation of proof obligations for aspect-oriented systems, expressed in standard linear temporal logic, from initial aspectual requirements and associated trade-offs [7]. It also facilitates use of formal methods tools and test case generators. The proposed formal framework is taken as a starting point to propose the semantic consistency and completeness rules offered by the formal primitive’s package.

Bakker discusses an approach for traceability of concerns using XML [8]. This approach allows the determination of the impact of the change scenarios both on architectural concerns and on requirement statements. That defines a concern as an abstract entity, and the work products of the software development process as concrete concerns (also called units). The semantics used is complete with respect to traceability of concerns. However, the crosscutting concerns traceability has not

yet been explored. MAT can take the XML and XQuery semantics defined by Bakker to extend the completeness semantics by adding formality to it.

Ferreira et al. define a meta-aspect repository in the scope of early aspects [13]. The repository stores the necessary information to support navigation over all stored information for future reuse, versioning control and traceability from requirements to implementation. The versioning control can be used as a basic complementary criterion to validate the completeness and consistency semantics.

El-Maddah and Mainbaum propose the GOPCSD tool to trace aspects [11]. This tool permits checking the completeness and consistency of requirements, so as to remove most of their bugs. It also prepares the requirement stage for consistency and completeness checks, so as to bring the different aspects together. The user is able to understand the rationale for the requirements units, as well as to validate the overall operation of the process control application. Finally, the tool translates the corrected requirement into a new specification. For MAT, this approach contributes with the way as requirements can be reviewed from different angles to ensure that they do not prescribe any incomplete nor inconsistent behaviour, and that the views are integrated, using conflict and completeness analysis.

In general, these approaches provide some elements to support traceability during identification and specification of crosscutting concerns. However, they do not directly support and control the traceability of those elements throughout their lifecycle.

6. Conclusions & Further Work

Several early aspects approaches have recognized the need for traceability support within a given lifecycle phase or between different phases. The aim of our work is to develop a general metamodel, the Meta Aspect for Traceability, to support traceability of crosscutting concerns. To validate the metamodel we will be using existing proposals for traceability to guarantee that such concrete models can be derived from our abstract model.

Currently our work is in its infancy. The next step is to define each of the MAT packages, mainly focusing on the “formal primitives’ package” since this one will be responsible to minimize the possible mismatches generated by the models.

For future work we intend to develop the packages that compose MAT specially the *Formal Primitives* package. Also we will study MOF in more detail as mechanism to specify the modelling language.

Acknowledgements

We would like to thank Fernando Arango and Raquel Anaya for fruitful discussions on a previous version of this document. Finally, we thank the reviewers for their detailed comments and useful insights on how to develop this work further.

References

- [1] R. Chitchyan, A. Rashid, P. Sawyer, J. Bakker, M. Pinto Alarcon, A. Garcia, B. Tekinerdogan, S. Clarke, A. Jackson, "Survey of Aspect-Oriented Analysis and Design Approaches", AOSD-Europe-ULANC-9, AOSD-EUROPE network of excellence. May 2005.
- [2] A. Rashid, A. Moreira and J. Araújo, "Modularisation and Composition of Aspectual Requirements", 2nd Int. Conf. On AOSD, USA, ACM Press, pp. 11-20, March 2003.
- [3] A. Moreira, A. Rashid, J. Araújo, "Multi-Dimensional Separation of Concerns in Requirements Engineering," 13th Intl. Conf. on RE, IEEE Computer Society, France, 2005.
- [4] A. Moreira, J. Araújo, and I. Brito, "Crosscutting Quality Attributes for Requirements Engineering," presented at Software Engineering and Knowledge Engineering Conference (SEKE), Ischia, Italy, 2002.
- [5] S. Clarke, E. Baniassad, Aspect-Oriented Analysis and Design. The Theme Approach. Addison-Wesley, Object Technology Series, 2005. ISBN: 0-321-24674-8.
- [6] S. Clarke and R. J. Walker, "Composition Patterns: An Approach to Designing Reusable Aspects". 23rd Intl. Conf. on ICSE, Canada, IEEE CS Press, pp. 5-14, 2001.
- [7] S. Katz, A. Rashid, "From Aspectual Requirements to Proof Obligations for Aspect-Oriented Systems", Proc. RE, 2004, IEEE CS Press, pp. 43-52.
- [8] P.J. Bakker, "Traceability of concerns". Master Thesis. University of Twente. 2005.
- [9] J. Whittle and J. Araújo, "Scenario Modelling with Aspects," IEE Proceedings - Software Special Issue, vol. 151, pp. 157-172, 2004.
- [10] J. Araújo and A. Moreira, "An Aspectual Use Case Driven Approach," presented at VIII Jornadas de Ingeniería de Software y Bases de Datos (JISBD), Alicante, Spain, 2003.
- [11] I. A. El-Maddah and T.S. E. Maibaum, "Tracing Aspects in Goal Driven Requirements of Process Control Systems", 3rd Intl. Conf. on AOSD, March 2004.
- [12] Y. Yu, J. C. S. d. P. Leite, and J. Mylopoulos, "From Goals to Aspects: Discovering Aspects from Requirements Goal Models," presented at International Conference on Requirements Engineering, Kyoto, Japan, 2004.
- [13] R. Ferreira, R. Raminhos, A. Moreira, "Metadata Driven Aspect Specification", AOM Workshop, MoDELS 2005, Jamaica, October 2005.
- [14] I. Gurevych, R. Porzel, M. Strube, "Annotating the Semantic Consistency of Speech Recognition Hypotheses", 3rd SIGdial Workshop on Discourse and Dialogue, pp. 46-49, Philadelphia, PA, July 2002.
- [15] R.M. Colomb, R. Weber. "Completeness and Quality of an Ontology for an Information System". International Conference on Formal Ontology in Information Systems (FOIS'98) Trento, Italy, 6-8 June, 1998. In N. Guarino (ed.) Formal Ontology in Information Systems IOS-Press (Amsterdam) 207-217.
- [16] K.M. Fant, S. A. Brandt, "Considerations of completeness Expression of Combinational Processes", 2002 Theseus Research, Inc. Page 1-11. 2524 Fairbrook Drive. Mountain View, CA 94040.
- [17] B. Tekinerdogan, P. Clements, A. Moreira, and J. Araújo (Editors). Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, Mar. 2004.
- [18] I. Brito and A. Moreira. Integrating the NFR framework in a RE model. In Tekinerdogan et al. [17], pages 27–32.
- [19] R. E. Filman, D. P. Friedman. "Aspect-Oriented Programming is Quantification and Obliviousness", Workshop on Advanced Separation of Concerns, OOPSLA'00, Minneapolis October 2000.
- [20] P. Habela, K. Subieta, "Standard Metamodel for Object Databases: Roles and Issues", Workshop on Object-Oriented Databases at TOOLS Europe 2001 conference.
- [21] J. Evermann, Y. Wand, "Toward Formalizing Domain Modeling Semantics in Language Syntax", IEEE Trans. Software Eng. 31(1): 21-37, 2005.
- [22] E. Damiani, B. Oliboni, E. Quintarelli and L. Tanca, "Modeling semistructured data by using graph-based constraints", On The Move to Meaningful Internet Systems 2003: OTM 2003 WorkshopsItaly, LNCS Vol. 2889, November 2003, pp. 20-21.
- [23] OMG. Meta-Object Facility (MOF™), MOF 1.4 specification; supersedes formal/01-11-02. <http://www.omg.org/technology/documents/formal/mof.htm>
- [24] OMG. Unified Modeling Language (UML), version 2.0. OMG document: formal/05-04-01. <http://www.omg.org/technology/documents/formal/uml.htm>
- [25] J. Araújo, A. Moreira, I. Brito, and A. Rashid, "Aspect-Oriented Requirements with UML," presented at Workshop on Aspect-Oriented Modelling with UML (held in conjunction with the International Conference on Unified Modelling Language UML 2002), 2002.
- [26] A. Moreira, J. Araújo, and A. Rashid, "A Concern-Oriented Requirements Engineering Model," presented at Conference on Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal, 2005.
- [27] I. Jacobson, "Use Cases and Aspects—Working Seamlessly Together," Journal of Object Technology, vol. 2, pp. 7-28, 2003.
- [28] J. Whittle and J. Araújo, "Scenario Modeling with Aspects," IEE Proceedings - Software, vol. 151, pp. 157-172, 2004.
- [29] S. M. Sutton and I. Rouvellou, "Concern Modeling for Aspect-Oriented Software Development," in Aspect-Oriented Software Development, R. E. Filman, T. Elrad, S. Clarke, and M. Aksit, Eds.: Addison-Wesley, 2004, pp. 479-505.
- [30] J. Grundy, "Aspect-Oriented Requirements Engineering for Component-based Software Systems," presented at 4th IEEE International Symposium on RE, 1999.

