A Conceptual Formalization of Crosscutting in AOSD

Klaas van den Berg[1] and José María Conejero[2]

[1]Trese Group, University of Twente
k.g.vandenberg@ewi.utwente.nl
[2] Quercus Software Engineering Group, University of Extremadura
chemacm@unex.es

Abstract. We propose a formalization of crosscutting based on a conceptual framework for AOSD. Crosscutting is clearly distinguished from the related concepts scattering and tangling. The definitions of these concepts are formalized and visualized with matrices and matrix operations. This allows more precise reasoning about crosscutting.

Topics of interest: Aspect-Oriented Requirement Engineering, Aspect-Oriented Analysis and Design, Aspect Foundations.

1 Introduction

One of the key principles in Aspect Oriented Software Development (AOSD) is Separation of Concerns (SOC). This principle is described in many publications [4][2]. Related with this principle is the problem of crosscutting concerns. Crosscutting is usually described in terms of scattering and tangling, e.g. crosscutting is the scattering and tangling of concerns arising due to poor support for their modularization. However, the distinction between these three concepts is vague, sometimes leading to ambiguous statements and confusion:

" .. the term "crosscutting concerns" is often misused in two ways: To talk about a single concern, and to talk about concerns rather than representations of concerns. Consider "synchronization is a crosscutting concern": we don't know that synchronization is crosscutting unless we know what it crosscuts. And there may be representations of the concerns involved that are not crosscutting. " (Kiczales, 2005 [5])

The goal of this paper is to come up with general and consistent definitions of above concepts and not to discuss specific examples - although some should fit somehow in this general framework. In this paper, we describe a conceptual framework with precise definitions of scattering, tangling and crosscutting. The description of crosscutting presented here is similar to some descriptions in the work of Masuhara & Kiczales (2003) [7] and of Mezini & Ostermann (2003) [9].

The paper is structured as follows. In section 2, we introduce the crosscutting pattern with definitions about crosscutting, tangling and scattering. In section 3, we show the formalization of crosscutting, how to visualize it in a crosscutting matrix and how to derive this matrix. In section 4 we conclude the paper.

2 Crosscutting Pattern

In this section, we describe the crosscutting pattern. We focus on definitions of crosscutting, tangling and scattering. Our proposition is that tangling, scattering and crosscutting can only be defined in terms of 'one thing' with respect to 'another thing': at least two domains, levels or phases are related with each other. For example:

- The two levels could refer to on one-hand concerns and on the other-hand *representations* of concerns, as stated in the citation in the introduction.
- The term *domain* could be used in mathematical sense where we have a mapping from one domain to another domain.
- The term *phase* could refer to phases in the software development cycle, such as concern modelling, requirements analysis, architectural design, detailed design and implementation.

We use the terms *source* and *target* (as in [8]) to denote two consecutive levels, domains or phases.

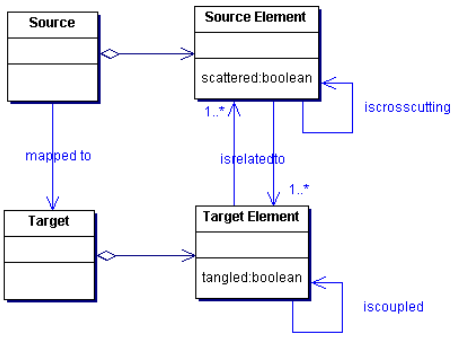

Fig. 1. Concept Diagram of Crosscutting Pattern

In the Crosscutting Pattern, elements in the source are related to elements in the target (see Fig. 1). We use the term *pattern* as in design patterns [3], in the sense of being a general description of frequently encountered situations [7], [9]: e.g. we have phrases as "one thing *with respect to* another thing". Some examples of source and target elements in the crosscutting pattern are the following: concern x module, concern x requirement, concern x architectural element, requirement x module, concern x implementation element.

There is a mapping between source elements and target elements. The terms crosscutting, tangling and scattering are defined as special cases of these mappings. On one hand there is a relation or mapping between source elements and target elements (see Fig. 1). The mapping has a multiplicity. It could be 1:1 or 1:many. In case of 1:many mappings we have scattering, defined as follows: *Scattering occurs when, in a mapping between source and target, a source element is related to multiple target elements.*

On the other hand, there is a relation between target elements and source elements, as a result of a mapping of source to target (Fig. 1). This relation is the reverse of the

mapping above. The multiplicity could be 1:1 or 1:many. In case of 1:many mappings we have tangling, defined as follows: *Tangling occurs when, in a mapping between source and target, a target element is related to multiple source elements.* We say that: *Two source elements are tangled if these elements are mapped onto the same target element.*

There is a specific combination of scattering and tangling which we call crosscutting, defined as follows: *Crosscutting occurs when, in a mapping between source and target, a source element is scattered over target elements and where in at least one of these target elements, some other source elements are tangled*. Now, we can also give a definition for the crosscutting of two source elements: *Source element s1 crosscuts source element s2 if s1 is scattered over target elements, and in at least one of these target elements, s1 is tangled with source element s2*.

We do not require that the second source element is scattered. In that sense, our definition is not symmetric and less restrictive than Masuhara & Kiczales's definition [7].

3 Case Analysis and Matrix Representation

In the previous section, we defined scattering, tangling and crosscutting. Now we discuss a case analysis of possible combinations. Assuming that the properties tangling, scattering, and crosscutting may be true or false, there are 8 combinations (see Table 1). However, crosscutting requires tangling and scattering, which eliminates 3 of these combinations (not feasible). There are five feasible cases listed in the table. Each case addresses a certain mapping from source to target.

Table 1. Combinations of the Properties Tangling, Scattering and Crosscutting

property case	tangling	scattering	crosscutting	feasibility
1.	no	no	no	Feasible
2.	yes	no	no	Feasible
3.	no	yes	no	Feasible
4.	yes	yes	no	Feasible
5.	yes	yes	yes	Feasible
6.	no	no	yes	Not feasible
7.	no	yes	yes	Not feasible
8.	yes	no	yes	Not feasible

In case 4, we have scattering and tangling in which no common elements are involved. Case 5 represents the specific combination of scattering and tangling as defined in the previous section. With our definition of crosscutting, we discriminate between the cases with just tangling, just scattering and on the other hand crosscutting. *Our proposition is that tangling and scattering are necessary but not sufficient conditions for crosscutting*. The rationale for disentangling these concepts is that there may be different solutions for each of these situations.

3.1 Matrix representation

The relation between source elements and target elements can be represented in a dependency matrix. In some sense, the dependency matrix displays the *traceability* between source and target elements. *A dependency matrix (source x target) represents the dependency relation between source elements and target elements (inter-level relationship)*. In the rows, we have the source elements, and in the columns, we have the target elements. In this matrix, a cell with 1 denotes that the source element (in the row) is mapped to the target element (in the column). Scattering and tangling can easily be visualized in this matrix (see the examples below). We define a new auxiliary concept *crosscutpoint* used in the context of dependency diagrams, to denote *a matrix cell involved in both tangling and scattering*. If there are one or more crosscutpoints then we say we have crosscutting.

Crosscutting between source elements for a given mapping to target elements, as shown in a dependency diagram, can be represented in a crosscutting matrix. *A crosscutting matrix (source x source) represents the crosscutting relation between source elements, for a given source to target mapping (represented in a dependency matrix).* In the crosscutting matrix, a cell with 1 denotes that the source element in the row is crosscutting the source element in the column.

A crosscutting matrix should not be confused with a coupling matrix. A *coupling matrix* shows coupling relations between elements at the same level (intra-level dependencies). In some sense, the coupling matrix is related to the design structure matrix [6]. A crosscutting matrix shows relations between elements at one level with respect to a mapping onto elements at some other level (inter-level dependencies).

We now give an example of case 5 presented in Table 1. We use the dependency matrix and the crosscutting matrix to visualize the definitions and examples. For case 5 (tangling (T), scattering (S), crosscutting), an example mapping is shown in Table 2. In this example, we have one scattered source element s[1] and one tangled target element t[3]. Moreover there is one crosscutpoint at matrix cell [1,3] (dark grey cell). We apply our definition of crosscutting and arrive to the crosscutting matrix. Source element s[1] is crosscutting s[3] (because s[1] is scattered over [t[1], t[3], t[4]] and s[3] is in the tangled one of these elements, namely t[3]. The reverse is not true: the crosscutting relation is not symmetric. s[1] is crosscutting s[3], but s[3] is not crosscutting s[1] because s[3] is not scattered (scattering is a necessary condition for crosscutting). This illustrates our proposition about crosscutting not being symmetric. The example is depicted in the following diagrams (Table 2).

Table 2. Example dependency matrix with tangling, scattering and one crosscutting

Dependency matrix

		Target				
		t[1]	t[2]	t[3]	t[4]	
Source	s[1]	1	0	1	1	S
	s[2]	0	1	0	0	NS
	s[3]	0	0	1	0	NS
		NT	NT	T	NT	

crosscutting matrix

		Source		
		s[1]	s[2]	s[3]
source	s[1]	0	0	1
	s[2]	0	0	0
	s[3]	0	0	0

3.2 Deriving Crosscutting Matrices

In this section, we describe how to derive the crosscutting matrix from the dependency matrix. Based on the dependency matrix, we define some auxiliary matrices: the *scattering matrix* (source x target) with just scattering, and the *tangling matrix* (target x source) with just tangling. These two matrices are defined as follows:

- In the scattering matrix, a row contains only dependency relations from source to target elements if the source element in this row is scattered (mapped onto multiple target elements); otherwise the row contains just zero's (no scattering relation).
- In the tangling matrix, a row contains only dependency relations from target to source elements if the target element in this row is tangled (mapped onto multiple source elements); otherwise the row contains just zero's (no tangling relation).

For our example in Table 2, these matrices are the following (see Table 3).

Table 3. Scattering, tangling and crosscutting matrices for dependency matrix in Table 2

Scattering matrix

		Target			
		t[1]	t[2]	t[3]	t[4]
source	s[1]	1	0	1	1
	s[2]	0	0	0	0
	s[3]	0	0	0	0

Tangling matrix

		Source		
		s[1]	s[2]	s[3]
target	t[1]	0	0	0
	t[2]	0	0	0
	t[3]	1	0	1
	t[4]	0	0	0

Crosscutting product matrix

		source		
		s[1]	s[2]	s[3]
Source	s[1]	1	0	1
	s[2]	0	0	0
	s[3]	0	0	0

Crosscutting matrix

		Source		
		s[1]	s[2]	s[3]
source	s[1]	0	0	1
	s[2]	0	0	0
	s[3]	0	0	0

We now define the crosscutting product matrix, showing the frequency of crosscutting relations. *A crosscutting product matrix (source x source) represents the frequency of crosscutting relations between source elements, for a given source to target mapping.* The crosscutting product matrix is not necessarily symmetric. The *crosscutting product matrix* ccpm can be obtained through the matrix multiplication of the scattering matrix sm and the tangling matrix tm: ccpm = sm . tm where $ccpm_{ik} = sm_{ij}\ tm_{jk}$

In this crosscutting product matrix, the cells denote the frequency of crosscuttings. This can easily be used for quantification of crosscutting (crosscutting metrics). The frequency of crosscuttings in this matrix should be seen as an upper bound. In actual situations, the frequency can be less then the frequency from this matrix analysis, because in the matrix we abstract from scattering and tangling specifics.

In the crosscutting matrix, a matrix cell denotes the occurrence of one or more crosscuttings; it abstracts from the frequency of crosscutting. The *crosscutting matrix* ccm can be derived from the crosscutting product matrix ccpm using a simple conversion: ccm_{ik} = if $(ccpm_{ik} > 0) \wedge (i \neq j)$ then 1 else 0

These two crosscutting matrices for the example are given in Table 3. In this example, there are no cells in the crosscutting product matrix larger than 1, except on the diagonal where it denotes a crosscutting relation with itself, which we disregard here. In the crosscutting matrix, we put the diagonal cells to 0. These formulas can be put in an Excel sheet using the function for matrix multiplication. By filling in the cells of the dependency matrix, the other matrices are calculated automatically.

4. Conclusions and future work

In this paper, we proposed a formalization of crosscutting based on a conceptual framework for AOSD. We introduced a crosscutting pattern with a mapping from a source to a target. With source and target, we abstract from specific levels or phases in software development. We defined crosscutting, tangling and scattering as separated cases based on different mappings between source and target. We introduced the dependency matrix and crosscutting matrix to visualize the definitions. We showed that it is possible to formalize these definitions. The proposed definitions are similar to definitions of crosscutting in some other publications, e.g. [7], although our definition is not symmetric and less restrictive.

Usually we encounter a number of consecutive levels or phases in software development, such as Requirement Analysis, Architectural Design, Detailed Design, and Implementation. This can be represented as cascading of crosscutting patterns: the target of the first pattern serves as source for the second one. We plan to apply this framework in different concrete situations in order to establish the suitability of the chosen concepts and definitions.

Acknowledgement

This work has been carried out in conjunction with the AOSD-Europe Project IST-2-004349-NoE [1] and also partially supported by CICYT under contract TIC2002-04309-C02-01. The authors would like to thank Gurcan Gulesir and other members of the Software Engineering Group at the University of Twente for their discussions.

References

1. AOSD-Europe (2004). IST Project Proposal 004349, Annex I - Description of Work, 1 September 2004.
2. Filman, R., et al., *Aspect-Oriented Software Development*. 2004: Addison-Wesley
3. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns. Elements of reusable object-oriented software: Addison-Wesley
4. Hürsch, W. and Lopes, C. (1995). Separation of Concerns. Technical Report, College of Computer Science, Northeastern University

5. Kiczales, G. *Crosscutting*. AOSD.NET Glossary 2005 [cited; Available from: http://aosd.net/wiki/index.php?title=Crosscutting
6. Lopes, C.V. and S.K. Bajracharya. An analysis of modularity in aspect oriented design. In 4th international conference on Aspect-Oriented Software Development. 2005. Chicago, Illinois
7. Masuhara, H. and G. Kiczales. Modeling Crosscutting in Aspect-Oriented Mechanisms. In ECOOP 2003. 2003. Darmstadt
8. MDA (2003). MDA Guide Version 1.0.1, document number omg/2003-06-01
9. Mezini, M. and K. Ostermann. Modules for Crosscutting Models. In 8th International Conference on Reliable Software Technologies. 2003. Toulouse, France: LNCS 2655
10. Tekinerdogan, B. ASAAM: Aspectual Software Architecture Analysis Method. in WICSA 4th Working IEEE/IFIP Conference on Software Architecture. 2004: IEEE