Towards developing generic solutions with aspects

A. M. Reina
Languages and Computer Systems Department
Avda. Reina Mercedes, s/n
41012 Seville, Spain
reinaqu@lsi.us.es

J. Torres
Languages and Computer Systems Department
Avda. Reina Mercedes, s/n
41012 Seville, Spain
jtorres@lsi.us.es

M. Toro
Languages and Computer Systems Department
Avda. Reina Mercedes, s/n
41012 Seville, Spain
mtoro@lsi.us.es

ABSTRACT

Software industry has to face up to continuous and fast changes of technology as well as varying customer's requirements. In order to adapt software for these new platforms and technologies at a minimum cost some proposals like MDA have been brought up, but also, simultaneously, others like aspect-oriented programming are addressing the changeability of customer's requirements. We propose the use of the MDA philosophy to raise the level of abstraction of current aspect-oriented design modelling languages, because most of them are platform dependent. Thus, we suggest the use of concern-specific modelling languages to specify concerns in a platform independent way. And, we also propose to tailor UML to the specific requirements of each aspect, so that, UML extension mechanisms (MOF metamodels and UML-profiles) are needed to define semantics of new aspect-specific constructs. Finally, a rational is given to determine which extension mechanism is the best for specifying each concern-specific language.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

Design, Languages

Keywords

advanced separation of concerns, MDA, UML

1. INTRODUCTION

Software industry has one special feature that makes it different from other industries: the high rate of changes that are produced in a relatively short period of time. In the last few years, additional technologies and platforms have been brought up, and we have become familiar with terms such as Java, Linux, HTML, XML, SOAP, UML, J2EE, .NET, JSP, Flash, web services, and so on. In this changing world, many companies have had to be compliant with these new technologies due to customer's requirements or, also, because they need some tools based on these new platforms. Therefore, to face up to technological changes and the variability of customer's requirements at a minimum cost has become a key point for the survival of any company.

Researchers are aware of these needs, so that, different proposals which are trying to solve these problems can be found in literature. On the one hand, MDA (Model Driven Architecture) [31] is facing up to technological changes. On the other hand, aspect-oriented programming [19] improves software evolution because it localizes changes in concrete points and minimizes the dependencies among the different aspects that compose the whole system.

Although the first proposals in aspect-orientation community were mainly for separating concerns at programming level [17, 32, 7, 22], researchers felt the need for raising the level of abstraction, thus new proposals have been brought up to specify concerns at design-time. Most of these proposals [6, 13, 37, 36, 1, 41] are inspired by AspectJ [33]. Moreover, many of them has as a main aim the code generation, and they end up generating AspectJ code.

We think that the previous proposals are too dependent on a concrete platform and they have been thought having in mind the generation of code for aspect-oriented languages. But many times customer's requirements impose the choice of a platform that is not aspect-oriented. Therefore, we think that the level of abstraction should be higher at design time, in such a way that although we work with separated concerns, the system can be adapted to any platform, whether aspect-oriented or not.

This paper proposes the use of MDA philosophy to achieve this platform independence. If MDA ideas are followed, models to specify concerns platform-independently will be needed. We think that UML and its extensions (UML-profiles or MOF metamodels) are enough to specify these concerns, and as a consequence, we propose the definition of domain-specific languages based on these UML-extensions to model concerns.

The rest of the paper is organized as follows: in section 2, a general overview of MDA is given, in order to clarify some terms and define some terminology used in the rest of the paper. After that, an analysis of the different proposals for defining concerns at the design level is made in section 3. Then, in section 4, the details of our proposal are introduced, and finally, the paper is concluded and our future lines of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UML '04 Lisbon, Portugal
Copyright 2004 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

work are pointed out.

2. MDA

MDA [31] is a framework defined by the OMG (Object Management Group) for software development. The foundation of this proposal is the importance of models in the software development process. In order to face up to technological changes, MDA proposes the separation of the specification of the functionality and the specification of the implementation of the system on a specific platform.

The main steps in the MDA development process are depicted in Figure 1, where three different layers of modelling are shown: PIM, PSM and code. According to the OMG [31], a *platform independent model* (PIM) is a view of a system from the platform independent viewpoint, while a *platform specific model* (PSM) is a view of a system from the platform specific viewpoint.

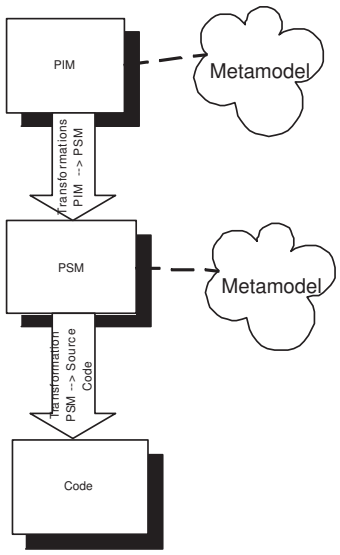

Figure 1: Main steps in the MDA development process

MDA does not require the use of UML to specify PIMs or PSMs, but in case other languages are required, the MOF metamodels for them should be indicated. The *Meta Object Facility*(MOF)[30] is a standard maintained by the OMG for the description of modelling constructs, and it can be considered as one of the key foundations of MDA.

The development process proposed by MDA starts with the specification of a platform independent model. Then, a set of transformations are needed in order to obtain the specification of the platform specific model, that is, the model but custom-designed for the specific platform where the application is going to be deployed. After that, another set of transformations are applied to the PSM to get the final source code.

3. MODELLING ASPECTS

Although the first proposals formulated for separating concerns were focused on implementation [17, 32, 7, 22], the level of abstraction has risen more and more. Thus, nowadays we can find proposals that pursue the separation from the very beginning of the software development process, and, as a consequence, a new term has been coined for naming these concerns obtained during the first phases of the software development process: *early aspects*. Nearly at the same time, many proposals have been brought up to model concerns.

First of all these proposals for specifying concerns at design time have been surveyed and analyzed, and as a result, its main features and contributions have been summarized in Table 1. This table shows the different proposals ordered by its year of publication. Thus, the first column of the table represents the year of publication, and the second one, its reference. The rest of the columns reflect a set of features that are interesting to compare all of them. They are:

Abstraction Level This property can have two values: high or low. The level of abstraction represents if the constructs proposed in the modelling language are close to concepts of programming languages or not. A proposal with a low level of abstraction use constructs that are near to the concepts managed by a programming language.

Inspired by Many of the modelling languages have been inspired by proposals which separate concerns at programming. Thus, this column shows the name of the proposal that has inspired the concepts expressed in the language.

Extension Mechanism In one way or another, all of the proposals use the UML as base language, and they suggest UML extensions to express concerns. This column represents the kind of extension mechanism used, whether a UML profile, a metamodel or others.

Purpose This property can have two different values: general or specific. A general purpose proposal means that a general purpose language is proposed for modelling all kind of aspects, while a specific purpose one means that one language should be specified for every aspect.

Main Contributions It shows a list with a brief description of the main contributions of the proposal.

Analyzing Table 1, it can be seen that one of the first proposals which has incorporated aspects at design is [37], where an extension to the UML metamodel is suggested in order to support aspects. Otherwise, in [6] new constructs are incorporated to UML (pointcuts and aspects) and the use of packages to separate and encapsulate aspects is proposed. Moreover, in [13] an extension to UML by means of stereotypes, tagged values and constraints is proposed.

UML has evolved along with aspect orientation, and more formal mechanisms to its extension have been brought up. Thus, there are proposals which use *light-weight* UML extensions, such as [1], where a profile for AOM is given and stereotypes for aspects and crosscuts are introduced. But also, there are others like [9], that extend UML in a *heavy-weight* way, that is, they propose a metamodel to express aspects.

Table 1 also shows that most of the proposals work with constructs that are very close to the ones defined in programming, and also, it reveals that those constructs are inspired mainly by Aspect/J (only [10] and [14] are based on HyperJ

Year	Ref.	Abstraction Level	Inspired by	Extension Mechanism	Purpose	Main Contributions
1999	[37]	low	Aspect/J	metamodel	general	* It adds new elements for aspect and woven class to the UML metamodel. * Model of relation aspect-class as an abstraction dependency relationship.
2002	[10]	low	Hyper/J	metamodel	general	* It adds new decomposition capabilities in order to align design models with individual requirements. * New abstract construct for `ComposableElement`. * New composition relationship.
2002	[36]	low	Aspect/J	stereotypes	general	* It provides representations for all language constructs in AspectJ. * It specifies an UML implementation of AspectJ's weaving mechanism. * Meta-attributes are defined to hold the weaving instructions.
2002	[41]	low	Aspect/J	profile	general	* Stereotypes of `Class` for modelling aspects and pointcuts. * Tagged values of `Association` for modelling relations between classes and aspects. * Stereotype of `Association` for modelling the relation between aspects. * Stereotypes of `Operation` for modelling before, after, around, ...
2002	[23]	low	Aspect/J	metamodel	general	* Definition of a metamodel to express aspects.
2002	[9]	low	Aspect/J	metamodel	general	* Definition of a metamodel to express aspects.
2002	[14]	high	Aspectual Collaborations	metamodel	general	* Packages as first-class citizens. * It refines the binding by which a complex behavior is applied to a core module. * It gives a new semantics to packages, adding them hig h-level properties.
2003	[6]	low	Aspect/J	profile	general	* Use of packages to separate concerns. * New constructs for pointcut and aspects. * New dependency relation, to link aspects and components to pointcuts.
2003	[13]	low	Aspect/J	stereotypes, tagged values and constraints	general	* Base package, aspect package and connector package.
2003	[1]	low	Aspect/J	profile	general	* Stereotypes of `Class` for modelling aspects. * Stereotype of `Association` for modelling crosscut. * Stereotypes of `Operation` for preactivation and postactivation ...
2003	[34]	low	Hyper/J	profile	general	* Extension provided specially for the development of product lines.
2003	[26]	high	none	profile	general	* It incorporates AO concepts into modelling and MDA.
2003	[21]	high	none	none	general	* Specification of aspects as models. * Weaving as model transformation.

Table 1: Classification of the different proposals for aspects modelling at design level

and Aspectual Collaborations [15], respectively). Furthermore, most of them suggest the use of a general purpose modelling language, that is, by means of a metamodel or a profile, they express all kind of aspects.

Finally, we think that the proposals which are closer to ours are [26] and [21]. In the first one, a framework for incorporating AO concepts into modelling and MDA is proposed. Weaving is made by means of model transformations, and, finally, a model which includes structure, behavior and logic is obtained. In this proposal, executable models are built for one single subject matter. These executable models are defined with the profile Executable UML [27].

On the other hand, in [21], the MDA approach and the specification of components and aspects in their own modelling languages are proposed. They also see the weaving of an aspect into the component as a transformation of the component model. Thus, a weaver in this context, will be a model transformer that will take the aspect model and the component model as input and will produce a component model transformed. It also proposes a need to investigate how aspects can be modelled.

4. SEPARATING CONCERNS

If the different proposals introduced in the previous section are analyzed, it turns out that most of them specify aspects by means of a general-purpose design modelling language in such a way that the same UML extension is used for expressing every kind of aspect. There is also a general trend: the inspiration on terms and concepts of AspectJ [18]. We think that these proposals are valuable, but they are platform-specific. That is, they were thought pursuing the generation of aspect-oriented code. But many times, customer's requirements impose platforms that are not aspect-oriented. Therefore, we need a way to specify concerns independently of the specific platform, in a higher level of abstraction.

We are not discarding the profiles and metamodels analyzed in the previous section, but we are leaving them for PSMs (Platform Specific Models), and in order to separate concerns at PIMs we propose the use of metamodels and UML profiles, in such a way that a UML-based language should be specified for each high-level aspect.

We think that nowadays we are in a point very similar to the one we were at the beginning of aspect-oriented pro-

gramming. At those times, many specific-purpose aspect languages proposals were brought forth, such as Cool and RIDL [25]. One of the main disadvantages of these kinds of languages is that they couldn't support aspects different to the ones they were designed for. However, they have a higher abstraction level than the *base language*, that is, the language used for the implementation of the basic functionality.

But now, this inconvenience can be overcome at the design level using UML as the modelling language, because it is a general purpose language which can be extended by way of two extension mechanisms. These mechanisms (metamodels and UML-profiles) let us express the different aspects by means of specific terms of the aspect's domain. We think that a combination of both of them can be used to specify concerns at the PIM level and although nowadays, an architect has to choose which is the best extension mechanism to model a specific domain, in the future, the distinction between them seems to be erased. Thus in [12], it is stated: *'We envision MDA tools on the horizon that provide much more freedom and eliminate the distinction between metamodels and profiles'.*

But now, the first step to develop an application is to determine the number of concerns to be separated. Once the concepts are clear, we should decide if we are going to specify every one by means of an UML profile or a metamodel. In this sense, in [11] Desfray gives a rational in order to choose the right metamodeling technique. We propose this rational to choose if an aspect should be specified with a metamodel or a UML-profile. Thus, if we adapt Desfray's proposal to concerns, it can be stated that they should be specified with a MOF based technique, if

- the domain is well defined and has a unique, well-accepted main set of concepts, or
- the model is not subject to be transferred into other domains, or
- there is no need to combine the aspect domain with other domains.

On the contrary, a UML-profile based technique, should be chosen when

- the aspect domain is not subject to consensus, or
- many changes and evolutions may occur, or
- the aspect domain may be combined with other domains in an unpredictable way, or
- models defined under the domain may be interchanged with other domains.

Another important advantage of choosing a profile extension is that generic UML tools can be used, whereas heavyweight extensions are likely to be too dependent on vendors and the revision of their tools. The main disadvantage of the profile approach is that it is not as semantically powerful as the metamodel approach.

Let's see an example in order to clarify our proposal, we have chosen a generic distributed web application. Firstly, we should decide which high level concerns are going to be specified separately. A typical distributed web application is composed of the following concerns, at least: user interface, navigation, security, distribution and persistence. Thus, separate models should be maintained for every concern.

Figure 2 depicts the structure of the distributed web application with our proposal. As we propose an MDA approach there are three different layers of models: PIMs, PSMs and code. The first layer, the platform-independet one, shows the five different models corresponding to the high level concerns enumerated previously. As well as [14], we think that packages should be first-class citizens. Thus, we have one package for every concern.

In an ideal situation this separation obtained in the first layer should be maintained from PIMs to source code, however, we should take into account that many times we have to deal with specific platforms which don't separate concerns adequately, due to customer's impositions. Thus, in Figure 2, some transformations are applied to the PIMs specifying aspects of navigation and user interface, and PSMs for the JSF (Java Server Faces) [8] platform are obtained. In this case, JSF does not provide a complete separation of navigation and user interface, so we'll have to express them at PSM level using the same metamodel or profile.

There are many proposals for the modelling of navigation and user interface [28, 5, 20], but there is a general agreement for modelling user interface and navigation by means of a metamodel. Moreover, in [28], a discussion about choosing a metamodel or a profile to model web applications is made, and it is concluded that the metamodel is the best option for modelling web applications because the semantic distance between UML elements and web modelling elements is too large. In this case, we also propose following the general trend and use models based on one of the metamodels mentioned previously to specify user interface and navigation.

However, if we look at the persistence aspect, we can check that most of the proposals [40, 3] have opted for a UML-profile, because the semantic distance is less. In [40] a profile for persistence in PIMs is proposed whereas in [3] a relational persistence profile is suggested. This profile can be considered technology-specific, because it has been thought to model a concrete technology: relational databases. But it should be noticed that there are other options in object persistence, for example, data might be stored in XML in a file system.

With regards to distribution, in [35] a profile for distribution is defined, valuable to be used in PIMs. In relation to PSMs, the OMG has adopted a MOF metamodel of Java and EJB.

Finally, in [24] a new modelling language named SecureUML is specified as a metamodel extension to UML. This language is focused on access control.

To conclude this section, our vision of high-level concerns is composed of a set of packages that are related to a set of metamodels and UML profiles stored in MOF and profiles repositories.

5. CONCLUSIONS AND FURTHER WORK

After surveying many of the design modelling languages defined to express concerns, it turns out that most of them are general purpose and low level languages, which means that most of them are looking for a language which expresses every kind of aspect, but also they use constructs that are very close to the ones used in aspect oriented programming languages.

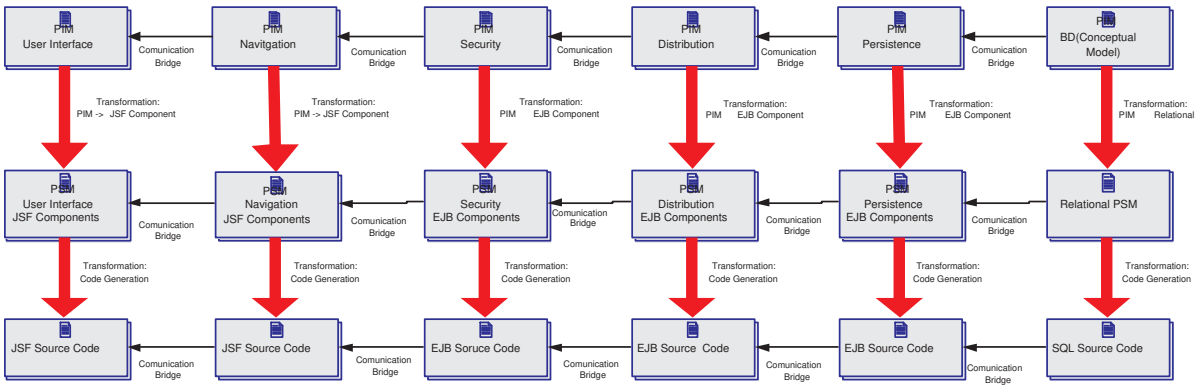

Figure 2: Structure of a typical distributed web application

We think that there is a need for raising the abstraction level of these modelling languages, because most of the surveyed have been thought to produce aspect-oriented source code, but sometimes, we have to work with non aspect-oriented platforms due to customer's requirements.

In order to increase the level of abstraction, and therefore, to adapt more easily to changes of requirements and technology, the MDA philosophy has been proposed. Thus, adjusting to this philosophy two levels of models and a set of transformations are needed. Firstly, we will have a set of models that are platform independent. By means of a set of transformations, we will obtain a set of models that are specific to the platform where the system is going to be deployed.

To separate concerns adequately, we propose the extension of UML by means of a metamodel or a UML profile for every concern. That is, we propose the use of domain specific web modelling languages, because they let us reason easily about concerns. We also propose the use of packages as first class citizens to group all models related to one specific aspect.

As a future line of research we are working on a framework to support all the ideas proposed in this paper. In this first stage, we are focusing specifically on web applications, so we'd like to test this framework with this kinds of applications. We also want to define repositories of metamodels with some of the ones proposed in this paper. Some of these extensions, need to be completed, and other new ones have to be developed.

Finally, another important focus of our research is the study of the incompatibilities and relationships among different high-level aspects, and the resolution of conflicts when they are weaved. We would like to provide the right modelling mechanism to specify the order of weaving or model transformations.

6. ACKNOWLEDGMENTS

This work has been partially supported by the Spanish *Ministery of Science and Technology* and FEDER funds: TIC 2003-369.

7. REFERENCES

[1] O. Aldawud, T. Elrad, and A. Bader. UML profile for aspect-oriented software development. In Aldawud et al. [2].

[2] O. Aldawud, M. Kandé, G. Booch, B. Harrison, and D. Stein, editors. *Third International Workshop on Aspect Oriented Modeling*, Mar. 2003.

[3] S. W. Ambler. Towards a Relational Persistence Model Profile for UML 2.0. In COM00 [29].

[4] *Workshop on Aspect-Oriented Modeling with UML (AOSD-2002)*, Mar. 2002.

[5] L. Baresi, F. Garzotto, L. Mainetti, and P. Paolini. Meta-modeling Techniques Meet Web Application Design Tools. In *Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering*, volume 2306 of *LNCS*, pages 294–307. Springer-Verlag, 2002.

[6] M. Basch and A. Sanchez. Incorporating aspects into the UML. In Aldawud et al. [2].

[7] L. Bergmans and M. Akşit. Composing crosscutting concerns using composition filters. *Comm. ACM*, 44(10):51–57, Oct. 2001.

[8] P. S. Bhogill. An Introduction to Java Server Faces. *Java News Brief*, Aug 2003.

[9] C. Chavez and C. Lucena. A metamodel for aspect-oriented modeling. In AOSD-UML02 [4].

[10] S. Clarke. Extending standard UML with model composition semantics. *Science of Computer Programming*, to appear.

[11] P. Desfray. UML Profiles versus Metamodeling Extensions... an Ongoing Debate. In COM00 [29].

[12] D. S. Frankel. *Model Driven Architecture. Applying MDA to Enterprise Computing.* Wiley Publishing, Inc., 2003.

[13] I. Groher and S. Schulze. Generating aspect code from UML models. In Aldawud et al. [2].

[14] S. Herrmann. Composable designs with UFA. In AOSD-UML02 [4].

[15] S. Herrmann and M. Mezini. Combining composition styles in the evolvable language LAC. In *Workshop on Advanced Separation of Concerns in Software Engineering (ICSE 2001)*, May 2001.

[16] M. Kandé, O. Aldawud, G. Booch, and B. Harrison, editors. *Second International Workshop on Aspect-Oriented Modeling with UML (<<UML>>2002)*, Sept. 2002.

[17] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. Getting started with

AspectJ. *Comm. ACM*, 44(10):59–65, Oct. 2001.

[18] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of AspectJ. In J. L. Knudsen, editor, *Proc. ECOOP 2001, LNCS 2072*, pages 327–353, Berlin, June 2001. Springer-Verlag.

[19] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin. Aspect-Oriented Programming. In M. Akşit and S. Matsuoka, editors, *11th Europeen Conf. Object-Oriented Programming*, volume 1241 of *LNCS*, pages 220–242. Springer Verlag, 1997.

[20] N. Koch and A. Kraus. Towards a Common Metamodel for the Development of Web Applications. In J. M. C. Lovelle, B. M. G. Rodríguez, L. J. Aguilar, J. E. L. Gayo, and M. del Puerto Paule Ruíz, editors, *Web Engineering, International Conference, ICWE 2003, Oviedo, Spain, July 14-18, 2003, Proceedings*, volume 2722 of *Lecture Notes in Computer Science*, pages 497–506. Springer, 2003.

[21] V. Kulkarni and S. Reddy. Supporting Aspects in MDA. In WISME-UML03 [39].

[22] K. Lieberherr, D. Orleans, and J. Ovlinger. Aspect-oriented programming with adaptive methods. *Comm. ACM*, 44(10):39–41, Oct. 2001.

[23] J. M. Lions, D. Simoneau, G. Pitette, and I. Moussa. Extending OpenTool/UML Using Metamodeling: An Aspect-Oriented Programming Case Study. In Kandé et al. [16].

[24] T. Lodderstedt, D. A. Basin, and J. Doser. SecureUML: A UML-Based Modelling Language for Model-Driven Security. In *Proceedings of the 5th International Conference on the Unified Modelling Language*, 2002.

[25] C. V. Lopes. *D: A Language Framework for Distributed Programming*. PhD thesis, College of Computer Science, Northeastern University, 1997.

[26] S. J. Mellor. A Framework for Aspect-Oriented Modeling. In UML-AOM03 [38].

[27] S. J. Mellor and M. J. Balcer. *Executable UML: A Foundation for Model-Driven Architecture*. Addison Wesley, 2002.

[28] P. Muller, P. Studer, and J. Bézivin. Platform Independent Web Application Modeling. In P. Stevens, J. Whittle, and G. Booch, editors, *UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference, San Francisco, CA, USA, October 2003, Proceedings*, volume 2863 of *LNCS*, pages 220–233. Springer, 2003.

[29] OMG. *Proceedings of the First Workshop on UML in the .COM Enterprise: Modeling CORBA, Components, XML/XMI and Metadata*, 2000.

[30] OMG. Meta Object Facility Specification Version 1.4. Technical Report OMG document formal/02-04-03, OMG, 2002.

[31] OMG. MDA Guide Version 1.0. Technical Report omg/2003-05-01, OMG, May 2003.

[32] H. Ossher and P. Tarr. The shape of things to come: Using multi-dimensional separation of concerns with Hyper/J to (re)shape evolving software. *Comm. ACM*, 44(10):43–50, Oct. 2001.

[33] X. PARC. Aspectj home page. web, 2002.

[34] I. Phillipow, M. Riebisch, and K. Boellert. The Hyper/UML Approach for Feature Based Software Design. In UML-AOM03 [38].

[35] R. Silaghi, F. Fondement, and A. Strohmeier. Towards an MDA-Oriented UML Profile for Distribution. Technical Report IC/2004/49, Swiss Federal Institute of Technology in Lausanne, Lausanne, Switzerland, May 2004. Also to be published in the Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference, EDOC, Monterey, CA, USA. September 20-34, IEEE Computer Society, 2004.

[36] D. Stein, S. Hanenberg, and R. Unland. An UML-based Aspect-Oriented Design Notation. In G. Kiczales, editor, *Proc. 1st Int' Conf. on Aspect-Oriented Software Development (AOSD-2002)*, pages 106–112. ACM Press, Apr. 2002.

[37] J. Suzuki and Y. Yamamoto. Extending UML with aspects: Aspect support in the design phase. In *Int'l Workshop on Aspect-Oriented Programming (ECOOP 1999)*, June 1999.

[38] *Fourth International Workshop on Aspect-Oriented Modeling with UML (<<UML>>2003)*, Oct. 2003.

[39] *Proceedings of the Workshop in Software Model Engineering held in conjunction with the UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference*, oct 2003.

[40] W. Witthawaskul and R. Johnson. Specyifing persistence in platform independent models. In WISME-UML03 [39].

[41] A. A. Zakaria, H. Hosny, and A. Zeid. A UML Extension for Modeling Aspect-Oriented Systems. In Kandé et al. [16].