Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 742 – 748

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

NOLE: an AOM Weaver for Aspect Oriented Modeling of Real-Time System

Naoufel Machta[a,*], Mohamed Taha Bennani[a], Samir Benahmed[a]

[a]*University of Tunis el Manar, University Campus , Tunis,Tunisia*

Abstract

Legacy applications that are already designed and maintained could be reused by adding new features like security, temporal constraints, etc. Aspect oriented approaches are an emerging technique that allow separation between functional and non-functional mechanisms. Separation of concerns, in aspect oriented design, en†hances productivity, reduces development costs and improves time to market delivery.
In this paper, we introduce AOMRTSYS an approach for weaving crosscutting concerns on UML and UML MARTE model. Then we focus on the NOLE weaver used by AOMRTSYS. We detail its operations and present the techniques used to implement it.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

Keywords: Aspect Oriented Modeling, Software development, UML MARTE, Real-Time System, Weaver ;

1. Introduction

Aspect Oriented Modeling (AOM) approaches aim to define transversal concerns separately. These are defined by functional and non-functional requirements. For example, billing or bank transaction systems represent functional

* Corresponding Author +216 97644 581
Email : naoufel.machta@isi.rnu.tn, naoufel.machta@yahoo.fr

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.141

requirements. However, authentication or data encryption mechanisms are considered as non-functional requirements. Subsequently, every aspect oriented modeling approach comprises these transversal concerns according to its own weaving process. Within the implementation phase, aspects' weaving uses the following notions: "*Joinpoint*", "*Pointcut*", "*Advice*" and "*Aspect*". A *Joinpoint* describes the location where the advice is added. An advice is a fragment of code, which is inserted before, around or after the defined *Pointcuts*. A *Pointcut* consists of a set of *Joinpoints*. An Aspect is a module that defines the advice and their *Pointcuts*.

Nomenclature

AOM	Aspect Oriented Modeling
AOP	Aspect Oriented Programming
AOMRTSYS	Aspect Oriented Modeling of Real-Time SYStem
UML	Unified Modeling Language
MARTE	Modeling and Analysis of Real Time and Embedded systems
R	Weaving rule
Ri	Instance of the Weaving rule
XML	Extensible Markup Language
XMI	XML Metadata Interchange

There is no consensus on the definition of these notions in the context of AOM. All the approaches dealing with this problem define their self-concepts. Usually, the non-functional requirements are defined using complex and elaborated models, which describe the whole process. The benefit of this choice is to reuse the components modeling the non-functional requirements (i.e. non-functional components). Nevertheless, weaving such components is relatively difficult. A component model could contain sub-models, for example, a structural and a behavioral one. Consequently, defining *Joinpoint* is difficult and potentially heterogeneous. Also, if the component provides many services, the weaving process will be more tedious. Aspect Oriented Modeling of Real-Time SYStem[1] (AOMRTSYS) approach addresses the aspect oriented modeling problem using an elementary and incremental way. According to our approach, a non-functional component is formed by many atomic elements. Every element of the language or modeling formalism is a potential atomic element. In addition, the weaving of a complex functional requirement, using the NOLE weaver, will be done in several steps using an incremental way. This proposition permits the simplification of the weaving step. We only weave atomic elements instead of complex models. This proposition is the cornerstone of an aided modeling system.

This paper is structured as follows. A general overview of AOMRTSYS approach is presented in Section 2. In Section 3, details of the NOLE weaver and techniques for its implementation are given. Related works are discussed in Section 4, and the last section draws our conclusion.

2. Overview of AOMRTSYS

Aspect Oriented Modeling of Real-Time System[1] (AOMRTSYS) is an AOM approach intended for the separation of concern in the modeling and design of real-time systems. AOMRTSYS aims at providing new ways of modularization in order to separate non functional requirement from traditional object-oriented units of decomposition during real-time system software development. Separation of real-time constraint has been the subject of many early research efforts[2, 3, 4] proving its possibility. In fact, a real-time application can be decomposed into functional requirements and non-functional ones. Functional requirements define, as any other domain, application core design. However, non-functional requirements is a set real-time constraints like scheduling, timing, concurrency and resource sharing.

According to our approach, a non-functional component is formed by many atomic elements. Every element of the language or modeling formalism is a potential atomic element. In addition, the weaving of a complex functional requirement will be done in several steps using an incremental way. This new proposition simplifies weaving

process and builds complex systems step-by-step[5] in an incremental way. We only weave atomic elements instead of complex models. This proposition is the cornerstone of an aided modeling system.

AOM concepts are similar to that of AOP[6]. However, those of AOM focus mainly on the composition of structural and behavioral models in the software modeling and simulation stages. In AOMRTSYS, *Joinpoint*, *Pointcut*, *Advice* and *Aspect* have the same intention as in AOP, but they are expressed differently. In AOMRTSYS, we adopt the following definitions:

- *Joinpoint*: refers an element or a set of elements (node or a set of nodes) of the model.
- *Pointcut*: consists of a set of *Joinpoints*.
- *Advice*: can be a model element or an element property, depending on *Joinpoint*.

AOMRTSYS is a symmetric AOM approach; there is no difference between aspect and base model. This approach uses a language for modeling crosscutting concerns (aspects). The language is an OCL-like code using the model name space and founded on the following pattern, which we called "Weaving Rule" ***R***:

```
R = Joinpoint.WeaverAction(Constraint)
```

In ***R***, *Joinpoint* point out where constraints will be woven on the model. *WeaverAction* put down action to be performed by the weaving tool. *Constraint* describes the type of constraint to be performed. As an instance of ***R*** (***Ri***) describes an elementary operation on the model, an advice is carried out by one or more ***Ri***.

Weaving plan, also named composition plan, yields the integration of multiple modular artifacts into a coherent whole[7]. In AOMRTSYS, the weaving plan is generated manually by the designer. It contains a set of instructions about which aspect to weave and where. Instructions are elementary operations each one described by a ***Ri***.

3. The NOLE Weaver

In AOM, the role of the weaver is to perform the integration of the aspects into the base. In the case of AOMRTSYS we started from a UML[8] application model to get UML MARTE[9, 10] application model. This allows to use and maintain old model application and also to generate several models of real-time application with different configuration, since for a single model application we just generate a weaving plan as the requirements (real time) of the environment and the target architecture.

3.1. Principle and features

The principle of the NOLE weaver is illustrated in Figure 1. Starting from a base model (legacy model) and specifications of crosscutting concerns, a plan of weaving is generated manually by the real-time application designer (Fig.1 (a)). NOLE takes as input a legacy application model and a weaving plan and generates as output a target application model (Fig.1 (b)). NOLE weaver contains two main components; the weaver core and the ***Ri*** validation tool.

Fig. 2 gives an overview of the components of NOLE Weaver and gives an idea about how it works and the interaction between its different modules.

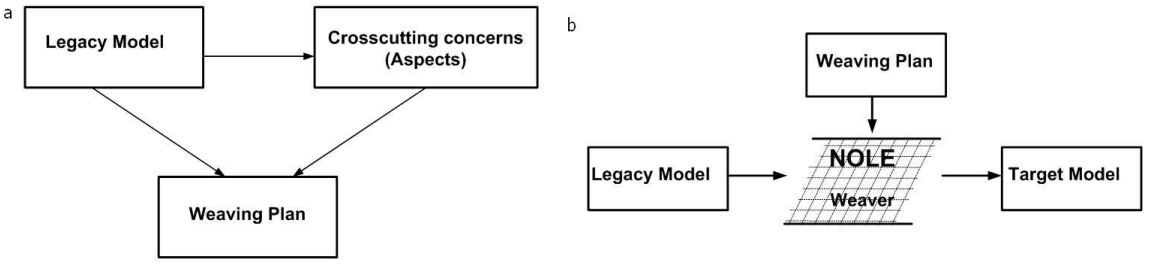

Fig. 1. (a) Weaving plan generation; (b) Nole weaver Input and Output

3.2. ***Ri*** *Validation tool*

As its name suggests, the ***Ri*** validation tool provides verification and validation of each ***Ri*** independently of the others. The verification and validation involve three main steps; the lexical analysis syntactic analysis and semantic analysis.

3.2.1. Lexical analysis

The tool extracts the atomic components of the concerns (i.e. *Joinpoint*, *WeaverAction* and *Constraint*) using a lexical analyzer which is defined by three classes of regular expressions; each one extracts a class of atomic components. As an example we introduce, in this section, some regular expressions which extract *Joinpoints*.

1) JP → JP1 | JP2 | JP3 | JP4 | JP5 | JP6 | JP7 | JP8

2) JP1 → classDiagName | seqDiagName | stateDiagName

3) JP2 → classDiagName DOUBLECOLON className

4) classDiagName → CD_(String)

The first regular expression presents eight *Joinpoint* alternatives. For example, in the regular expression 2 the *Joinpoint* is a diagram and may be a class diagram, a sequence diagram or a state diagram. However, in regular expression 3 the *Joinpoint* is a class that belongs to a class diagram. *classDiagName* (regular expression 4) is preceded by the string *CD* which is a flag fixing the context of the subsequent string (i.e. Class Diagram).

3.2.2. Syntactic analysis

The syntactic analysis of ***Ri***-structure verifies if the rule is trying to make an adjustment on the structure of the application model. For example, in the case of UML, the structural model is described only by the class diagram. This adjustment may be adding, modifying or deleting an element of the application model. We handle, therefore, classes, attributes, class methods, class stereotypes and relationships among classes. The following examples show some of the well-formed rules defined in the tool grammar for the syntactic analysis of ***Ri*** structure.

1 *<Riclass> : <JP1><POINT><ADDCLASS><CST0>*

2 *| <JP2><POINT><ADDMETHOD><CST4>*

3 *| <JP2><POINT><STEREOTYPE><CST8>*

4 *| <JP5><POINT><ADDRELATION><CST6*

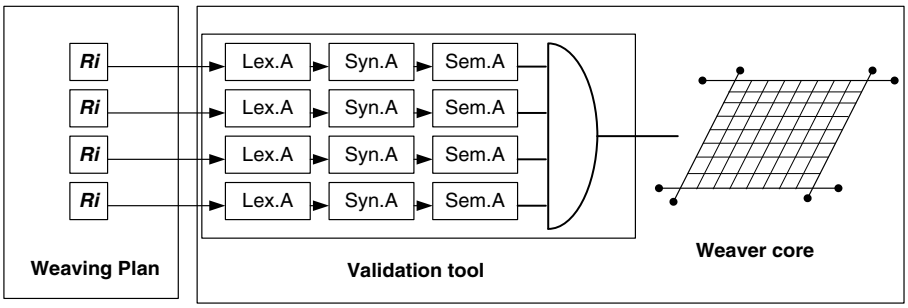

Fig 2. NOLE WEAVER components

3.2.3. Semantic analysis

The concept of semantic analysis and consistency checking of weaving plan as described in this paper is to integrate the information from the legacy model in the individual check of each ***Ri***. Verification of the whole weaving plan (***Ri*** consistency with each other) is constrained by the informal nature of UML. A weaving plan is a coherent set of atomic elements. At this level coherence must be validated by the designer after the generation of the weaving plan. The automation of validation of coherence will be the subject of future work dealing with incremental validation of models[11].

3.3. The weaver core

In AOM approaches, the weaving process leads to a change of model. Several methods accomplish this modification, but the most used concept is a model transformation. The latter facilitates the manipulation and edition of the model. The biggest challenge in transforming the model is to avoid the loss of information. To remedy this problem we used XMI[12] which is an XML[13]-based standard for the interchange of UML models.

The weaver core has to parse XMI files. JDOM[14] is used for this task. JDOM is open source tool and effectively ensures writing, reading, and manipulating XML documents in Java. For the generation of XMI UML model Papyrus[15] modeling tool is used. Papyrus is an open source modeling tool working under Eclipse framework[16]. Papyrus tool is compatible with UML MARTE profile, Since, XMI MARTE model generated by weaver can be exploited by Papyrus .

The model weaver introduces the *Advice*, code realizing the constraint, into places suggested by *Joinpoints*. This is a general definition of *Advice*. However, in some cases, injected code has only a reference on *Joinpoints*.

4. Related works

There are three approaches for modeling aspects in UML. The first approach defines some matching criteria to identify common elements belonging to the two models. It uses a generic merging algorithm to obtain the resulting model. Examples of this approach include Theme/UML[17, 18, 19] and Reddy *et al*.[20]. The second approach tries to adapt aspect oriented programming concepts to design context. Examples of this approach include[21, 22, 23]. The third approach applies the model or graph transformations techniques to tackle model composition. This composition is specified by a graph rules. Examples of this approach include[24 ,25 26].

In the first approach, matching criteria uses model elements signatures. Matching by name could lead to a conflict in the case of method overloading. In Theme/UML[17], the sequence diagram templates model behavioral adaptation to be inserted. It implicitly defines the relative position within the lifeline. This kind of definition is not explicit and may not work at the merging step. The adaptation sequence could interleave the original messages of the lifeline. In France *et al*.[27], merging algorithms are based on directives, which are expressed in a textual form and use algebraic operators[28].

The early implementation of merging directives where in JAVA but the recent one are developed using kermeta[29]. The directive operands handle only structural elements of the application model. Behavioral concerns are not defined yet. In the second approach, not all base model elements could be *Joinpoints*, only a restrictive set is allowed. In fact, some approaches specify events as a *Joinpoints*. Others, however, allow only states to be *Joinpoints*. Also, the advices are limited to before, after and around which is restrictive since it could be intended to weave parallel behaviors. In the third approach, the source system model is transformed into the target one using graph rules. The later could be defined by abstract or concrete syntaxes. The former defines transformation at the meta-level which is the abstract syntax of the modeling language[30]. In the concrete syntax, graph rules are defined over the modeling language which is easier to the modeler[30, 25, 26]. The knowledge of the metamodeling language is not required. Motorola WEAVR[22] is a SDL-based aspect oriented UML-profile presented by Cottenier *et al.* Modeling elements in WEAVR are similar to AspectJ[31] elements, such as *Joinpoint*, advice and *Pointcut*. The weaving tool depends on the underlying platform. MATA[32] (Modeling Aspects using Transformation Approach) defines a layer on top of abstract syntax-based graph transformation (GT) rules. It uses the standard mechanisms provided by GT tools like AGG[33]. However, the matching concept is not well-formalized by the proposed syntax-based language. In general, it makes impossible to determine if a base model has a match. The approaches proposed

by Klein *et al.*[25] and Grnomo *et al.*[26] define different matching strategies. In the first approach, they apply the matching strategy to the entire *Pointcut*. Gronmo *et al.*[26] propose to mix two strategies in the same *Pointcut* since they have introduced an arbitrary events symbol. Unlike our approach, both strategies try to match a set of model elements. It makes the process more complex. In our approach, the matching strategy process one model element at a time.

5. Conclusion and future work

AOMRTSYS approach introduces a weaving language made by instances of the rule pattern. The weaving process of this method runs within the design-time of the system life cycle. This choice provides a language independent and more flexible approach.

AOMRTSYS approach has been applied to object oriented modeling of real-time and embedded system using UML and MARTE profile. A weaver called NOLE has been implemented and tested for a number of case studies. We have used the Papyrus open source tool for import and export models to XMI document. However, since XMI is a standard XML format, any other tool may be used.

NOLE weaver includes weaver core and rule instances verification and validation engine. The former was developed using JDOM and XPATH tools. The latter includes only lexical and syntactic analyzer developed using Flex and Bison tools.

There are a number of interesting avenues for further work that would build upon NOLE. Firstly, the development of ***Ri*** coherence checker to automate the validation of a weaving plan. NOLE covers only class diagram and sequence diagram. The next development steps would extend our approach to cover all UML diagrams.

References

1. Machta, Naoufel, M. Taha Bennani, and Samir Ben Ahmed. Aspect oriented Modeling of Real-Time system with UML and MARTE. *Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on*. IEEE, 2010.
2. NIELSEN, Brian et AGHA, Gul. Towards reusable real-time objects. Annals *of Software Engineering*, 1999, **7**:1-4, p. 257-282.
3. AKSIT, Mehmet, BOSCH, Jan, VAN DER STERREN, William, *et al. Real-time specification inheritance anomalies and real-time filters*. Springer Berlin Heidelberg, 1994.
4. LOPES, Cristina Videira et LIEBERHERR, Karl J. Abstracting process-to-function relations in concurrent object-oriented applications. In: *Object-Oriented Programming*. Springer Berlin Heidelberg, 1994. p. 81-99.
5. MORIN, Brice, BARAIS, Olivier, JÉZÉQUEL, Jean-Marc, *et al.* Weaving aspect configurations for managing system variability. In : *2nd International Workshop on Variability Modelling of Software-intensive Systems*. 2008.
6. KICZALES, Gregor, LAMPING, John, MENDHEKAR, Anurag, *et al. Aspect-oriented programming*. Springer Berlin Heidelberg, 1997.
7. SCHAUERHUBER, Andrea, SCHWINGER, Wieland, KAPSAMMER, Elisabeth,*et al.* A survey on aspect-oriented modeling approaches. *Relatorio tecnico, Vienna University of Technology*, 2007.
8. OBJECT MANAGEMENT GROUP, *et al.* Unified Modeling Language: Superstructure (v 2.4. 1). 2011.
9. SELIC, Bran et GÉRARD, Sébastien. *Modeling and Analysis of Real-time and Embedded Systems with UML and MARTE: Developing Cyber-physical Systems*. Elsevier, 2013.
10. OMG MARTE GROUP, *et al.* A UML profile for MARTE: Modeling and analysis of real-time embedded systems, beta 2 (convenience document with change bars). *OMG MARTE documentation*, 2011.
11. MENET, Ludovic, LAMOLLE, Myiam, et LE DC, Chan. Incremental validation of models in a MDE approach applied to the modeling of complex data structures. In : *On the Move to Meaningful Internet Systems: OTM 2010 Workshops*. Springer Berlin Heidelberg, 2010. p. 120-129.
12. OBJECT MANAGEMENT GROUP, *et al.* XML Metadata Interchange (XMI) version 2.4.2. formal/2014-04-04*, April 2014*.
13. BRAY, Tim, PAOLI, Jean, SPERBERG-MCQUEEN, C. Michael, *et al.*Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, 1998.
14. Jdom api documentation. available at : http ://www.jdom.org/downloads/docs.html, August 2014.
15. CEA. Papyrus UML, available at : http://www.eclipse.org/papyrus/, August 2014.
16. Eclipse framework, available at : http://www.eclipse.org/, August 2014.
17. DRIVER, Cormac, CAHILL, Vinny, et CLARKE, Siobhán. Separation of distributed real-time embedded concerns with theme/UML. In : *Model-based Methodologies for Pervasive and Embedded Software, 2008. MOMPES 2008. 5th International Workshop on*. IEEE, 2008. p. 27-33.
18. CLARKE, Siobhán et WALKER, Robert J. Generic aspect-oriented design with Theme/UML. *Aspect-oriented software development*, 2005, p. 425-458.
19. CLARKE, Siobhàn et BANIASSAD, Elisa. *Aspect-oriented analysis and design*. Addison-Wesley Professional, 2005.

20. REDDY, Raghu, FRANCE, Robert, GHOSH, Sudipto, *et al.* Model composition-a signature-based approach. In : *Aspect Oriented Modeling (AOM) Workshop*. 2005.
21. COTTENIER, Thomas, VAN DEN BERG, Aswin, et ELRAD, Tzilla. Modeling aspect-oriented compositions. In : *Satellite Events at the MoDELS 2005 Conference*. Springer Berlin Heidelberg, 2006. p. 100-109.
22. COTTENIER, Thomas, VAN DEN BERG, Aswin, et ELRAD, Tzilla. Motorola WEAVR: Aspect orientation and model-driven engineering. *Journal of Object Technology*, 2007, vol. 6, no 7, p. 51-88.
23. JACOBSON, Ivar et NG, Pan-Wei. *Aspect-oriented software development with use cases (addison-wesley object technology series)*. Addison-Wesley Professional, 2004.
24. WHITTLE, Jon, JAYARAMAN, Praveen, ELKHODARY, Ahmed, *et al.* MATA: A unified approach for composing UML aspect models based on graph transformation. In : *Transactions on Aspect-Oriented Software Development VI*. Springer Berlin Heidelberg, 2009. p. 191-237.
25. KLEIN, Jacques, FLEUREY, Franck, *et al.* Tissage d'Aspects Comportementaux. In : *Langages et Modèles à Objets: LMO'06*. 2006.
26. GRØNMO, Roy. *Using concrete syntax in graph-based model transformations*. 2009. Thèse de doctorat. University of Oslo.
27. FRANCE, Robert, RAY, Indrakshi, GEORG, Geri, *et al.* Aspect-oriented approach to early design modelling. *IEE Proceedings-Software*, 2004, vol. 151, no 4, p. 173-185.
28. KATZ, Shmuel, OSSHER, Harold, FRANCE, Robert, *et al.* (ed.). *Transactions on aspect-oriented software development VI: special issue on aspects and model-driven engineering*. Springer, 2009.
29. MULLER, Pierre-Alain, FLEUREY, Franck, et JÉZÉQUEL, Jean-Marc. Weaving executability into object-oriented meta-languages. In : *Model Driven Engineering Languages and Systems*. Springer Berlin Heidelberg, 2005. p. 264-278.
30. HENKLER, Stefan, GREENYER, Joel, HIRSCH, Martin, *et al.* Synthesis of timed behavior from scenarios in the fujaba real-time tool suite. In : *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009. p. 615-618.
31. KICZALES, Gregor, HILSDALE, Erik, HUGUNIN, Jim, *et al.* An overview of AspectJ. In : *ECOOP 2001—Object-Oriented Programming*. Springer Berlin Heidelberg, 2001. p. 327-354.
32. WHITTLE, Jon et JAYARAMAN, Praveen. Mata: A tool for aspect-oriented modeling based on graph transformation. In : *Models in Software Engineering*. Springer Berlin Heidelberg, 2008. p. 16-27.
33. TAENTZER, Gabriele. AGG: A graph transformation environment for modeling and validation of software. In : *Applications of Graph Transformations with Industrial Relevance*. Springer Berlin Heidelberg, 2004. p. 446-453.