PRISMA: Model-Driven Development of Aspect-Oriented Software Architectures*

Jennifer Pérez[1], Isidro Ramos[2], Jose A. Carsí[2], Cristóbal Costa-Soria[3]

Technical University of Madrid (UPM) - ETSISI-CITSEM, Madrid, Spain
jenifer.perez@eui.upm.es
Universidad Politécnica de Valencia (UPV) - DSIC, Valencia, Spain
iramos@dsic.upv.es, pcarsi@dsic.upv.es
Global Metanoia S.L., Paterna Technological Science Park (Valencia), Spain
ccosta@globalmetanoia.com

Abstract. This summary presents a methodology for supporting the development of AOSAs following the MDD paradigm. This new methodology is called PRISMA and allows the code generation from models which specify functional and non-functional requirements.

Keywords: MDD, Aspect-Oriented Software Architectures, code generation

1. Summary

Currently, mature MDD methodologies are required for supporting the code generation from models that specify non-functional requirements. Aspect-Oriented Software Architectures (AOSA) emerged to deal with the design of both, functional requirements and non-functional requirements, which opened an important challenge in the software engineering field: the definition of a methodology for supporting the development of AOSAs following the MDD paradigm. This new methodology should allow the code generation from models which specify functional and non-functional requirements. This contribution presents an approach, called PRISMA, which deals with this challenge. PRISMA follows the MDD approach by enabling software architects to define AOSA models, which allow the generation of the final code of AOSAs. The tasks of the software architect are facilitated thanks to the fact that: (i) the level of abstraction provided by models is higher than the provided by programming languages, and (ii) the code is automatically generated from models. The PRISMA MDD process is based on MOF and its models undergo the different levels of refinement during the process. Therefore, the PRISMA MDD process is divided into two main steps: (i) from the PRISMA Metamodel to PRISMA Type Models, and (ii) from PRISMA Type Models to PRISMA Configuration Models (see Fig. 1)

The PRISMA metamodel defines the PRISMA model and establishes its properties in a precise way (see 1, Fig. 1). To use these primitives in a modeling context, the PRISMA CASE tool provides the PRISMA metamodel through a graphical language to model PRISMA software architectures (see 2, Fig.1). PRISMA type models are

*Published in: Jennifer Pérez, Isidro Ramos, José A. Carsí, Cristóbal Costa-Soria: Model-Driven Development of Aspect-Oriented Software Architectures. Journal of Universal Computer Science, vol. 19, no. 10 (2013), 1433-1473, accepted: 27/5/13 © J.UCS. DOI: 10.3217/jucs-019-10-1433. Rankings: JCR: 0.762 , Journals CORE: A, Journals Microsoft Academic Research/Computer Science: 316 position of 1363

described using the concepts that are defined in the metamodel, guaranteeing that every PRISMA type model is defined conforming to the PRISMA metamodel (see 4, Fig. 1). The PRISMA MDD process assists the architect by providing mechanisms for the verification of models: this allows the detection of modelling mistakes, and keeps them from spreading throughout the rest of the stages. This verification distinguishes two kinds: verification rules that must always be satisfied (hard constraints), and verification rules that must be satisfied once the model has been completely finished (weak constraints). Once the PRISMA type model has been completely described, the architect can proceed to generate both the PRISMA AOADL specifications and the C# code corresponding to this model (see 4, Fig.1).In addition, since a PRISMA type model is a generic system architecture (banking system, tele-operated robot, etc.), the model can be reused for different specific systems. Once its code has been generated, the architect can proceed to define a specific system model, called configuration model. Every configuration model conforms to a PRISMA type model thanks to the fact that PRISMA configurations are defined using the concepts that have been previously defined in its PRISMA type model as modelling primitives. They are provided by PRISMA CASE, which automatically generates a domain-specific graphical modelling tool to configure specific software architectures from the type models, and generating the C# application code corresponding to the configuration model (see 5 and 6, Fig. 1).

The PRISMA MDD process is presented as an important advance in the automatic generation of code from AOSAs thanks to its automatic generation of AOADL specification, C# code and a domain-specific model to configure the final PRISMA software architecture of a system. This domain-specific model reduces the gap between the user and his/her knowledge about modelling. In addition, it has been applied to different kind of applications to validate its code generation capability: the robots TeachMover and Agrobot. The results revealed a high percentage of automatic code generation from PRISMA AOSA models: 94,3% for the TeachMover and 68,1% for the Agrobot, which correspond with 5686 and 1125 Generated Lines of Code. These code generation results reveal that the complete automatic code generation from AOSA models is feasible.

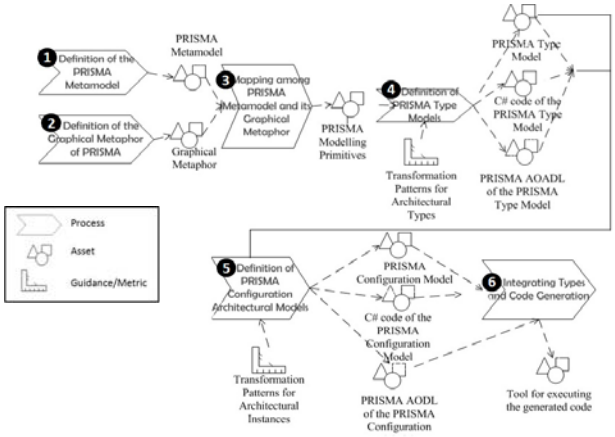

Fig. 1. MDD Process from the PRISMA Metamodel to Applications