

MORPHEUS: a supporting tool for MDD

Elena Navarro¹ & Abel Gómez² & Patricio Letelier² & Isidro Ramos²

¹ Department of Computing Systems, University of Castilla-La Mancha, Spain.
Elena.Navarro@uclm.es

² Dep. of Information Systems and Computation, Polytechnic University of Valencia, Spain.
{agomez, letelier, iramos}@dsic.upv.es

Abstract: Model-Driven Development (MDD) approach is gaining more and more attention both from practitioners and academics because its positive influences in terms of reliability and productivity in the software development process. ATRIUM is one of the current proposals following the MDD principles as the development is driven by models and a tool, MORPHEUS, support both its activities and models. This tool provides facilities for modelling, meta-modelling, and analysis and integrates an engine to execute transformations. In this work, this tool is presented describing both its architecture and its capabilities.

Keywords: Model-Driven Development, Requirements Engineering, Software Architecture

1. Introduction

Software development process is always a challenging activity, especially because systems are becoming more and more complex. In this context, the Model-Driven Development [24] (MDD) approach is gaining more and more attention from practitioners and academics. This approach promotes the exploitation of models at different abstraction levels, guiding the development process by means of transformations, so that traceability and automatic support becomes a reality. MDD has demonstrated positive influences for reliability and productivity of the software development process due to several reasons [24]: it allows one to focus on the ideas and not on the supporting technology; it facilitates not only the analysts get an improved comprehension of the problem to be solved but also the stakeholders obtain a better cooperation during the software development; etc. With those aims, MDD exploits models both to properly document the system and automatically or semi-automatically generate the final system. This is why the software development is shifting its attention [1] from “everything is an object”, so trendy in the eighties and nineties, to “everything is a model.”

ATRIUM [12][15] (Architecture generaTed from Requirements applying a Unified Methodology) has been defined following the MDD principles, as models

drive its application, and the tool MORPHEUS (see [13] for demos) has been built to support its models and activities. This methodology has been defined to guide the concurrent definition of requirements and software architecture, paying special attention to the traceability between them. In this context, the support of MORPHEUS is a valuable asset allowing the definition of the different models; maintaining traceability among them; supporting the necessary transformation, etc. This paper focuses on MORPHEUS and its support to a MDD process.

This paper is structured as follows. After this introduction, a brief description of ATRIUM is presented in section 2. Section 3 describes the supporting tool of ATRIUM, MORPHEUS. Related works are described in section 4. Finally, section 5 ends this paper by presenting the conclusions and further works.

2. ATRIUM at a glance

ATRIUM provides the analyst with guidance, along an iterative process, from an initial set of user/system needs until the instantiation of the proto-architecture. ATRIUM entails three activities to be iterated over in order to define and refine different models and allow the analyst to reason about partial views of both requirements and architecture. Fig. 1 shows, using SPEM [21] (Software Process Engineering Metamodel), the ATRIUM activities that are described as follows:

- *Modelling Requirements*. This activity allows the analyst to identify and specify the requirements of the system-to-be by using the *ATRIUM Goal Model* [18], which is based on KAOS [5] (Knowledge Acquisition in autOmated Specification) and the NFR (Non-Functional Requirements) Framework [2]. This activity uses as input both an informal description of the requirements stated by the stakeholders, and the CD 25010.2 *Software product Quality Requirements and Evaluation Quality model (SQuaRE)* [9]. The latter is used as framework of concerns for the system-to-be. In addition, the architectural style to be applied is selected during this activity [15].

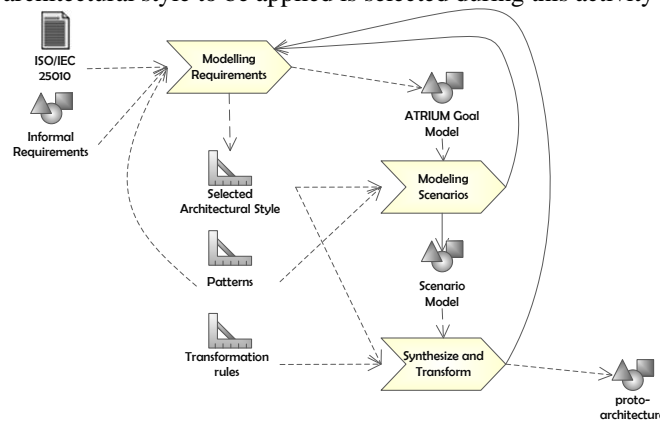


Fig. 1. An outline of ATRIUM

- *Modelling Scenarios.* This activity focuses on the specification of the *ATRIUM Scenario Model*, that is, the set of *Architectural Scenarios* that describe the system's behaviour under certain *operationalization* decisions [16]. Each ATRIUM Scenario identifies the architectural and environmental elements that interact to satisfy specific requirements and their level of responsibility.
- *Synthesize and Transform.* This activity has been defined to generate the proto-architecture of the specific system [14]. It synthesizes the architectural elements from the ATRIUM scenario model that build up the system-to-be, along with its structure. This proto-architecture is a first draft of the final description of the system that can be refined in a later stage of the software development process. This activity has been defined by applying *Model-To-Model Transformation* (M2M, [4]) techniques, specifically, using the QVT Relations language [20] to define the necessary transformations. It must be pointed out that ATRIUM is independent of the architectural metamodel used to describe the proto-architecture, because the analyst only has to describe the needed transformations to instantiate the architectural metamodel he/she deems appropriate. Currently, the transformations [15] to generate the proto-architecture, instantiating the PRISMA architectural model [22], have been defined. PRISMA was selected because a code compiler exists for this model.

ATRIUM has been validated in the context of the tele-operated systems. Specifically, the EFTCoR [8] project has been used for validation purposes. The main concern of this project was the development of a tele-operated platform for non-pollutant hull ship maintenance operations whose main structure is shown in Fig. 2. In this paper, we are going to use the specification made of the Robotic Device Control Unit (RDCU) to show how MORPHEUS provides support to each activity of ATRIUM. The RDCU is in charge of commanding and controlling in a coordinated way the positioning of devices along with the tools attached to them.

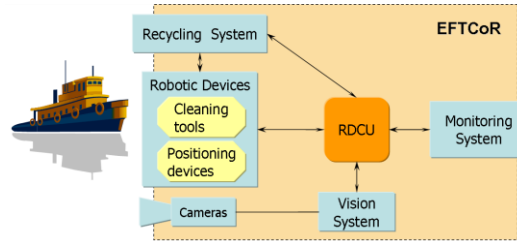


Fig. 2. Describing the EFTCoR platform

3. MORPHEUS: a MDD supporting tool

The main idea behind MORPHEUS is to facilitate a graphical environment for the description of the three models used by ATRIUM (Goal Model, Scenarios Model, and PRISMA Model) in order to provide the analysts with an improved legibility

and comprehension. Several alternatives were evaluated such as the definition of profiles, or the use of meta-modelling tools. Eventually, we developed our own tool in order to provide the proper integration and traceability between the models.

Fig. 3 shows the main elements of MORPHEUS. The *Back-End* layer allows the analyst to access to the different environments, and to manage the projects he/she creates. Beneath this layer, the different environments of MORPHEUS are shown, providing each one of them support to a different activity of ATRIUM. The *RepositoryManager* layer is in charge of providing the different environments with access to the repository where the different models and metamodels are stored. In addition, each one of the graphical environments (Requirements Model Editor, Scenario Editor, and Architecture Model Editor) exploits Microsoft Office Visio Drawing Control 2003 [25] (VisioOCX in Fig. 4, Fig. 9, Fig. 13) for graphical support. This control was selected to support the graphical modelling needs of MORPHEUS because it allows a straightforward management, both for using and modifying shapes. This feature is highly relevant for our purposes because all the kinds of concepts that are included in our metamodels can easily have different shapes, facilitating the legibility of the models. In addition, the user is provided with all the functionalities that Visio has, that is, she/he can manage different diagrams to properly organize the specification, make zoom to see more clearly details, print the active diagram, etc. In the following sections, each one of the identified environments is described.

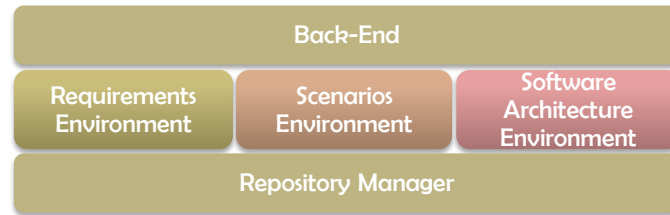


Fig. 3. Main architecture of MORPHEUS

1.1 Requirement Environment

As described in section 2, *Modelling Requirements* is the first activity of ATRIUM. In order to support this activity, the *Requirements Environment* was developed. From the very beginning of the EFTCoR project, one of the main problem we faced was how the requirements metamodel had to change to be adapted to the specific needs of the project. With this aim, this environment was developed with two different work contexts. The first context is the *Requirements Metamodel Editor* (RMME), shown in Fig. 4, which provides users with facilities for describing requirement meta-models customized according to project's semantic needs (see Fig. 5). The second context is the *Requirements Model Editor*

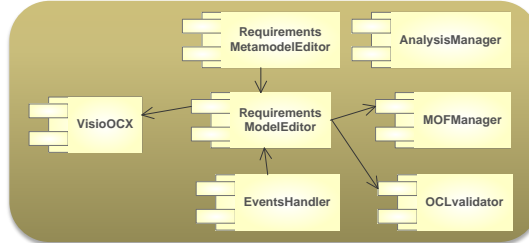


Fig. 4. Main elements of the Requirements Environment

(RME), also shown in Fig. 4, which automatically offers the user facilities to graphically specify models according to the active metamodel (see Fig. 8). These facilities are very useful to exploit MORPHEUS to support other proposals.

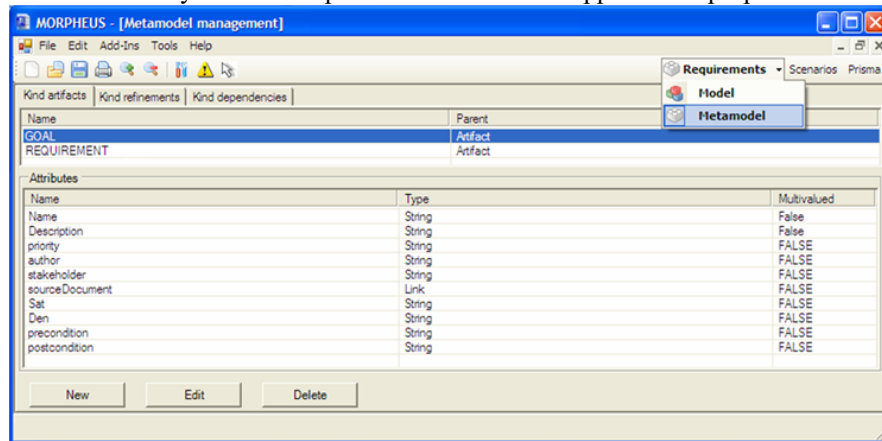


Fig. 5. Meta-Modelling work context (RMME) of the MORPHEUS Requirements Environment

It can be observed in Fig. 5 that the RMME allows the user to describe new meta-elements by extending the core metamodel described in Fig. 6, that is, new types of artefacts, dependencies, and refinements. This metamodel was identified and evaluated its applicability by analysing the existing proposals in requirements engineering [18]. For instance, Fig. 5 shows that the two meta-artifacts (goal, requirement) of the ATRIUM Goal Model were defined using the RMME. In order to fully describe the new meta-elements, the user can describe their meta-attributes and the OCL constraints he/she needs to check any property he/she deems appropriate. Fig. 7 shows how the meta-artifact *goal* was defined by

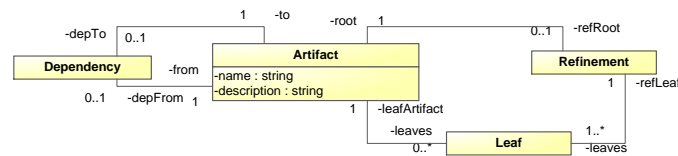


Fig. 6. Core-metamodel for the requirements environment

extending *artifact*; describing its meta-attributes *priority*, *author*, *stakeholder*, etc; and specifying two constraints to determine that the meta-attributes *stakeholder* and *author* cannot be null.

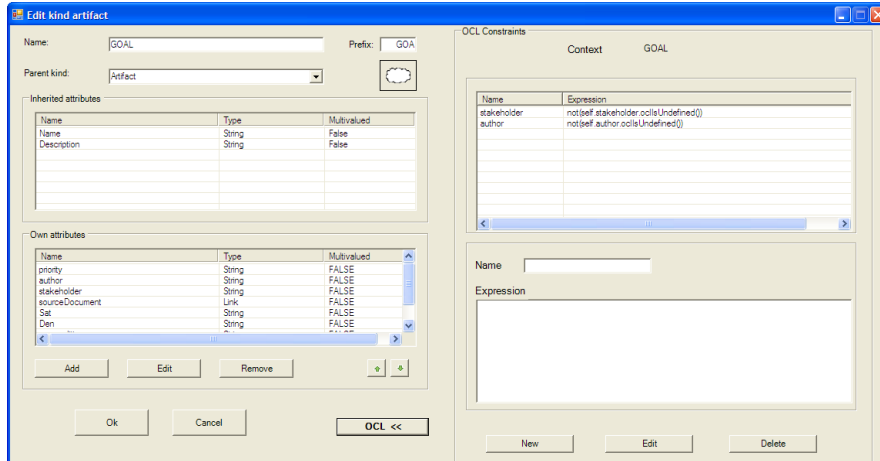


Fig. 7. Describing a new meta-artifact in MORPHEUS

It is worth noting that automatic support is provided by the environment for the evolution of the model, that is, as the metamodel is modified, the model is updated in an automatic way to support those changes, asking the user to confirm the necessary actions whenever a delete operation is performed on meta-elements or meta-attributes. This characteristic is quite helpful because the requirement model can be evolved as the expressiveness needs of the project do.

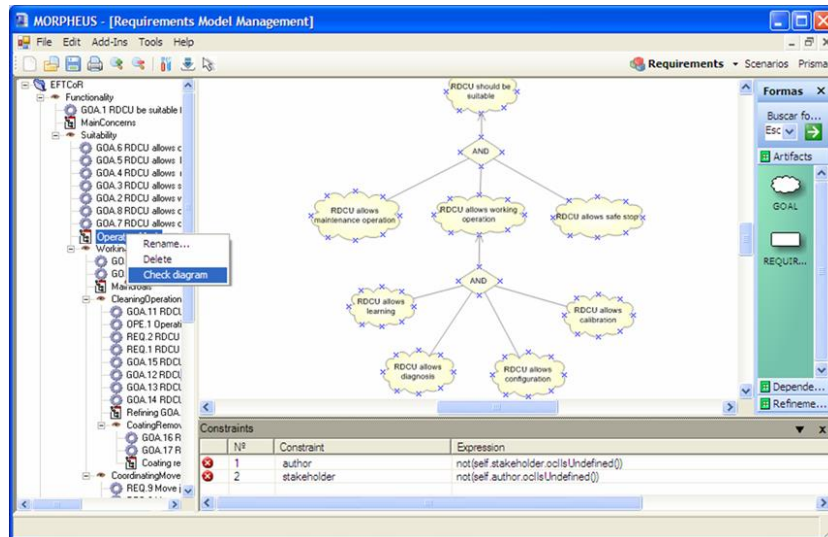


Fig. 8. Modelling work context (RME) of the MORPHEUS Requirements Environment

Once the metamodel has been defined the user can exploit it in the modelling context, RME, shown in Fig. 8. It uses VisioOCX to provide graphical support, as Fig. 4 shows, and has been structured in three main areas. On the right side, the *stencils* allow the user to gain access to the active metamodel. Only by dragging and dropping these meta-elements on the drawing surface in the centre of the environment, the user can specify the requirements model. He/she can modify or delete these elements by clicking just as usual in other graphical environments. For instance, some of the identified goals and requirements of the EFTCOR are described in the centre of the Fig. 8. On the left side of the RME, a browser allows the analyst to navigate throughout the model and modify it. As Fig. 4 illustrates, the *EventHandler* is in charge of manipulating the different events that arise when the user is working on the RME.

In addition, as Fig. 4 illustrates, the RME uses two components to provide support to OCL: *OCLValidator* and *MOFManager*. The former is an engine to check OCL constraints that was integrated in MORPHEUS. The later was developed to allow us to manipulate metamodels and models in MOF [19] format. By exploiting these components the constraints defined at the metamodel can be automatically checked. For instance, when the active diagram was checked, two inconsistencies were found that are shown at the bottom of the Fig. 8.

However, the support of the tool would be quite limited if it only provides graphical notation. For this reason, the *Analysis Manager*, shown in Fig. 4, has been developed to allow the user to describe and apply those rules necessary to analyse its models. These rules are defined by describing how the meta-attributes of the meta-artifacts are going to be valued depending on the meta-attributes of the meta-artifacts they are related to by means of which meta-relationships. Once these rules are defined, the Analysis Manager exploits them by propagating the values from the leaves to the roots of the model [17]. This feature can be used for several issues such as, satisfaction propagation [17], change propagation, or analysis of architectural alternatives [15].

1.2 Scenario Environment

As presented in section 2, *Modelling Scenarios* is the next activity of ATRIUM. This activity is in charge of describing the scenario model. This model is exploited to realize the established requirements in the goal model by describing

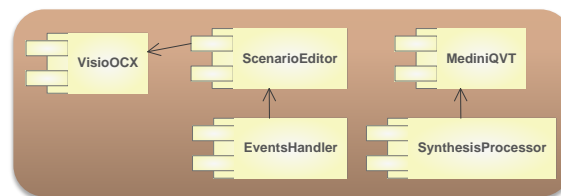


Fig. 9. Main elements of the Scenario Environment

partial views of the architecture, where only shallow-components, shallow-connectors and shallow-systems are described. In order to describe these scenarios, an extension of the UML2 Sequence Diagram has been carried out to provide the necessary expressiveness for modelling these architectural elements [15]. In order to provide support to this activity the *Scenario Editor* (SME), shown in Fig. 10, was developed. The *ScenariosEditor* uses the VisioOCX to provide the user with graphical support for modelling the Scenario Model. The *EventHandler* is in charge of managing all the events triggered by user actions. Fig. 10 illustrates how the SME has been designed. In a similar way to the RMME described in the previous section, it has been structured in three main areas. The *Model Explorer*, on the right, facilitates the navigation through the Scenario Model being defined in an easy and intuitive way and manages (creation, modification and deletion) the defined scenarios. It is pre-loaded with part of the information of the requirements model being defined. For this reason, the selected *operationalizations*, catalogued by their dimensions, are displayed. It facilitates to maintain the traceability between the Goal Model and the Scenario Model. Associated to each operationalization one or several scenarios can be specified to describe how the shallow architectural elements collaborate to realize that operationalization. In the middle of the environment is situated the *Graphical View* where the elements of the scenarios can be graphically specified. In this case, Fig. 10 depicts the scenario “OpenTool” that is realizing one of the operationalizations of the goal model. It can be observed how several architectural and environmental elements are collaborating by means of a sequence of messages. On the right side it can be seen the *Stencil* that makes available the different shapes to graphically describe the ATRIUM scenarios. The user only has to drag and drop on the *Graphical View* the necessary shapes. In addition, below the stencil a control allows the user to introduce the necessary properties for each element being defined.

Another component of the Scenario Environment is the *Synthesis processor* (see Fig. 11). It provides support to the third activity of ATRIUM *Synthesis and Transform* which is in charge of the generation of the proto-architecture. For its

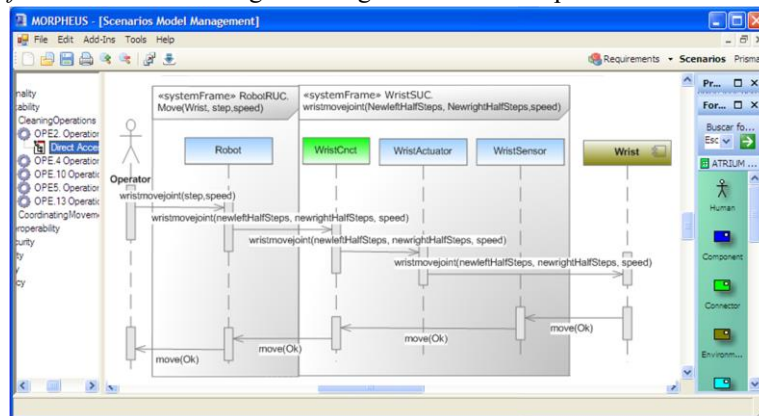


Fig. 10. What the Scenario Editor (SME) looks like

development, the alternative selected was the integration of one of the existing M2M transformation engines considering that it has to provide support to the QVT-Relations language. Specifically, a custom tool based in the *medini QVT* [11] transformation engine (licensed under the *Eclipse Public License*) was integrated as Fig. 11 illustrates. It accepts as inputs the metamodels and their corresponding models in XMI format to perform the transformation. This engine is invoked by the *Synthesis processor* which proceeds in several steps. First, it stores the Scenario Model being defined in XMI. Second, it provides the user with a graphical control to select the destination target architectural model, the QVT transformation to be used and the name of the proto-architecture to be generated. By default, PRISMA is the selected target architectural model because the QVT rules [15] for its generation have been defined. However, the user can define its own rules and architectural metamodels to synthesize the Scenario Model. Finally, the *Synthesis processor* performs the transformation by invoking the QVT engine. The result is an XMI file describing the proto-architecture.

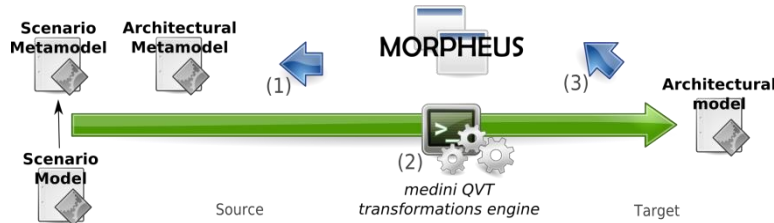


Fig. 11. Describing the Synthesis processor

1.3 Software Architecture Environment

As can be observed, both the Requirements Environment and the Scenario Environment provide support to the three activities of ATRIUM. However, as specified in section 2, a proto-architecture is obtained at the end of its application. This proto-architecture should be refined in a latter stage of development to provide a whole description of the system-to-be. With this aim the *Software Architecture Environment* [23] was developed. It makes available a whole graphical environment for the *PRISMA Architecture Description Language* [22] so that the proto-architecture obtained from the scenarios model can be refined.

As Fig. 13 depicts, this environment integrates VisioOCX for graphical support in a similar way to the previous ones. The *Architectural Model Editor* is the component that provides the graphical support, whose appearance can be seen in Fig. 12. It has three main areas: the stencil on the right where the PRISMA concepts are available to the user, the graphical view in the centre where the different architectural elements are described; and the model explorer on the right. It is worthy of note that this browser is structured in two levels following the

recommendation of the ADL [23]: definition level, where the PRISMA types are defined; and configuration level where the software architecture is configured.

As this environment should allow the user to refine the proto-architecture obtained from the synthesis of the scenario model, it provides her/him with facilities to load the generated proto-architecture if PRISMA was the selected target architectural model. In addition, it also provides an add-in that facilitates the generation of a textual PRISMA specification, which can be used to generate C# code by using the PRISMA framework.

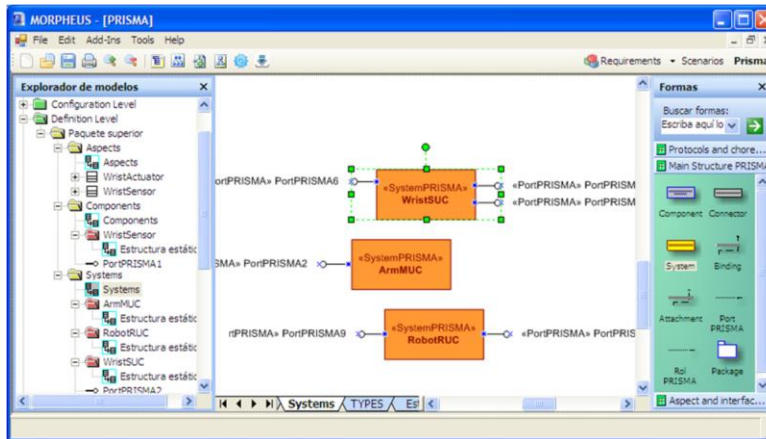


Fig. 12. What Architectural Editor looks like

4. Related works

Nowadays, MDD is an approach that is gaining more and more followers in the software development area, and lots of tools that support this trend have arisen. Nevertheless, none of the existing solutions can completely cover the capabilities of the MORPHEUS tool.

The Eclipse Modeling Framework (EMF) has become one of the most used frameworks to develop model-based applications. EMF provides a metamodeling language, called Ecore, that can be seen as an implementation of the Essential MOF language. Around EMF lots of related projects have grown that complement its modelling and metamodeling capabilities, such as OCL interpreters, model

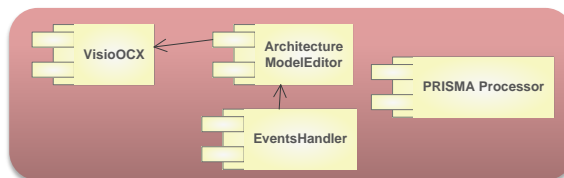


Fig. 13. Main elements of the Software Architecture Environment

transformation engines, or even tools able to automatically generate graphical editors, such as Graphical Modeling Framework (GMF [7]). The advantages are twofold: first they are usually quite mature tools, and second it is easy to interoperate with them by means of the XMI format. That is why the MORPHEUS tool has the *MOFManager* component: it allows us to reuse these tools as is the case of the OCL checker and the model transformations engine. Nevertheless, a solution completely based in EMF has also some important drawbacks. The main one is that, although it is not mandatory, this framework and its associated tools are fundamentally designed to deal with static models that do not change at run time. This factor makes frameworks like GMF completely useless for our purposes, because in MORPHEUS the requirements metamodel is populated with instances during its evolution and it is necessary to be able to synchronize them.

Other analyzed alternatives are the MS DSL Tools [3]. MS DSL tools are a powerful workbench that also provides modelling and metamodeling capabilities to automatically generate both code and graphical editors in Visual Studio. However, it exhibits the same weakness than the previous solution: it is basically designed to deal with models that do not evolve during time, so that, these models can only be modified during design time and not at run time. Moreover, it lacks of the wide community that provides complementary tools to deal, check and analyze models, in comparison with the solution that is completely based on EMF. This disadvantage is also present in other tools, like the ones associated to Meta-CASE and Domain Specific Modelling techniques, such as MetaEdit+ [10].

5. Conclusions and further works

In this work a tool called MORPHEUS has been presented paying special attention to how it provides support to a MDD process, ATRIUM. It has been shown how each model can be described by using this tool and, specially, how traceability throughout its application is properly maintained. It is also worth noting the meta-modelling capabilities it has, providing automatic support to evolve the model as the metamodel is changed. The integration of an OCL checker is also interesting as it allows the user to evaluate the model using the properties he/she deems appropriate.

Several works constitute our future challenges. Although the tool is quite mature, we are considering the development of other functionalities, as for instance, a model checker of the software architecture or a report generator for the requirements environment. It is also among our priorities to deploy the tool in the next future as an open source project to be evaluated and used by the community.

Acknowledgments. This work is funded by the Dept. of Science and Technology (Spain) I+D+I, META project TIN2006-15175-C05-01 and by the UCLM, project MDDRRehab TC20091111. This work is also supported by the FPU fellowship program from the Spanish government AP2006-00690.

References

1. Bézivin, J. (2004). *In search of a basic principle for model driven engineering*, Upgrade 5(2), pp. 21–24, 2004
2. Chung, L., Nixon, B. A., Yu, E., Mylopoulos J. (2000). *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, Boston.
3. Cook, S., Jones, G., Kent, S., Cameron A. (2007) *Domain-specific development with Visual Studio DSL tools*, Addison Wesley Professional.
4. Czarnecki K., Helsen S. (2006) *Classification of Model Transformation Approaches*. *IBM Systems Journal*, 45(3), pp. 621-645.
5. Dardenne A., van Lamsweerde A., Fickas S., (1993) *Goal-directed Requirements Acquisition*, *Science of Computer Programming*, 20(1-2), pp. 3-50.
6. Eclipse Modeling Framework. <http://www.eclipse.org/emf/>
7. Eclipse Graphical Modeling Framework. <http://www.eclipse.org/gmf/>
8. GROWTH G3RD-CT-00794 (2003) *EFTCOR: Environmental Friendly and cost-effective Technology for Coating Removal*. European Project, 5th Framework Prog.
9. ISO/IEC JTC 1/SC 7 N4098 (2008), *Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model*.
10. Kelly S., Lyytinen K., Rossi M., METAEDIT+ *A fully configurable Multi-User and Multi-tool CASE and CAME Environment*. *Proc. of 8th International Conference on Advances Information System Engineering*, LNCS1080, Springer-Verlag, 1996, pp. 1-21.
11. Medini QVT, <http://projects.ikv.de/qvt>.
12. Montero F., Navarro E. (2007), *ATRIUM: Software Architecture Driven by Requirements*, *Proc. 14th IEEE Int. Conf. on Engineering of Complex Computer Systems (ICECCS'09)*, IEEE Press, Jun. 2007, in press.
13. MORPHEUS (2009), http://www.dsi.uclm.es/personal/ElenaNavarro/research_atrium.htm
14. Navarro E., Cuesta C. E. (2008), *Automating the Trace of Architectural Design Decisions and Rationales Using a MDD Approach*, *Proc. 2nd European Conference Software Architecture*, LNCS 5292, Springer Verlag, Sep. 2008, pp. 114-130.
15. Navarro E. (2007), *Architecture Traced from Requirements applying a Unified Methodology*, PhD thesis, Computing Systems Department, UCLM.
16. Navarro E., Letelier P., Ramos I. (2007), *Requirements and Scenarios: playing Aspect Oriented Software Architectures*, *Proc. 6th IEEE/IFIP Conf. on Software Architecture*, IEEE Press, 2007, n. 23.
17. Navarro E., Letelier, P., Reolid, D., Ramos, I. (2007), *Configurable Satisfiability Propagation for Goal Models using Dynamic Compilation Techniques*, *Proc. Information Systems and Development (ISD'07)*, Springer US, Sep. 2007, pp. 167-179.
18. Navarro, E., Letelier, P., Mocholí, J.A., Ramos, I. *A Metamodeling Approach for Requirements Specification*. *Journal of Computer Information Systems*, 2006, 47(5): 67-77.
19. OMG (2006). Meta Object Facility (MOF) 2.0 Core Specification (ptc/06-01-01).
20. OMG (2005). Document ptc/05-11-01, QVT, MOF Query/ Views/Transformations.
21. OMG (2005). Software Process Engineering Metamodel (SPEM), ver. 1.1 formal/05-01-06.
22. Pérez, J., Ali, N., Carsí, J. Á., Ramos, I., (2006). *Designing Software Architectures with an Aspect-Oriented Architecture Description Language*, *Proc. 9th Int. Sym. on Component-Based Software Engineering (CBSE 2006)*, Jun. 2006, pp. 123-138, Springer Berlin/Heidelberg.
23. Pérez, J., Navarro, E., Letelier, P. and Ramos, I. (2006). *A Modelling Proposal for Aspect-Oriented Software Architectures*, *Proc. 13th Annual IEEE Int. Conf. and Works. on the Engineering of Computer Based Systems (ECBS'06)*, IEEE Press, Mar. 2006, pp. 32-41.
24. Selic, B. (2003). *The Pragmatics of Model-Driven Development*. *IEEE Soft.* 20(5), pp. 19-25.
25. Visio 2003 (2009), [http://msdn.microsoft.com/en-us/library/aa173161\(office.11\).aspx](http://msdn.microsoft.com/en-us/library/aa173161(office.11).aspx)