An Open Source Domain-Specific Tools Framework to Support Model Driven Development of OSS

Achilleas Achilleos[1], Nektarios Georgalas[2], and Kun Yang[1]

[1] University of Essex, Dept. of Electronic Systems Engineering, UK
{aachila,kunyang}@essex.ac.uk
[2] British Telecom Group, UK
nektarios.georgalas@bt.com

Abstract. Telecommunications companies undergo massive transformations which reflect onto exacting requirements for controlling the costs of new Operation Support Systems (OSS) development and integration. This calls for the adoption of new approaches, which improve agility and reusability. Model Drive Development (MDD), as specified by OMG, can drastically tackle these issues and has, therefore, attracted the interest of the telecommunications industry. Equally important is the Open Source paradigm. For MDD to gain wide industrial adoption, tools should be available to facilitate the OSS development process. In this paper, we specify requirements MDD tools should meet for effective application of the approach. An extensive survey is then carried out to evaluate existing meta-modelling frameworks over the identified tools requirements. Eventually, we present the Integrated Eclipse Model driven Environment (IEME), which comprises a unified environment of bundled Eclipse-based MDD facilities that also supports the automatic generation of domain-specific tools.

Keywords: Meta-modelling, MDA, model-driven development, domain specific languages, modelling editor tools.

1 Introduction

Telecommunications companies undergo massive transformations to become agile organizations shifting from their traditional profile as telephony operators into providers of networked-based ICT services. Some of their main challenges are to reduce the costs of their IT operational support systems (OSS), increase agility, and design the OSS infrastructure to support fast delivery of new services and products. A major contributing factor in the cost of running OSS is the integration tax incurred when legacy is integrated with new OSS Components Off-The-Shelf (COTS). This is mainly due to the multiplicity of platforms and middleware used by the OSS. Although advances are being made by the OSS industry towards standardising OSS capabilities and information models, through the OSS/J initiative [1] and the TeleManagement Forum (TMF) [2], systems are still designed around specific middleware technologies such as CORBA, J2EE, and .NET. This preserves the problem of heavy costs and lengthy process cycles of OSS to new versions of such platforms.

D.H. Akehurst, R. Vogel, and R.F. Paige (Eds.): ECMDA-FA 2007, LNCS 4530, pp. 1–16, 2007.

MDD of software, as specified by OMG [3], provides an approach that can drastically tackle the aforementioned issues. MDA provides clear distinction between models independent of technical details, namely, *Platform Independent Models (PIMs)* and models that include detail of the implementation technology, namely, *Platform Specific Models (PSMs).* MDA comprises a set of standards that enable the definition of Domain Specific Languages (DSLs) used to specify a system's structure and behaviour. DSLs are represented as meta-models based on the Meta-Object Facility (MOF) and can be precisely defined using the Object Constraint Language (OCL) [4] for defining constraints over meta-models as well as actual models. With sequential transformations among various DSLs, using the Query/View/Transform (QVT) standard, system implementations can be produced for particular platforms. MOF expressed model data can be exchanged between compliant tools using XML Metadata Interchange (XMI) technology.

We have practically applied MDA in several case-studies that demonstrated the advantages it offers in the process of designing, developing and integrating OSS in terms of improved quality and lower costs [5], [6], [7], [8]. In order to capitalize on these advantages it is instrumental that the MDA standards are implemented in the tools used for OSS development. In the context of the NGOSS/MDA TMF Catalyst project [9], a practical evaluation of mainstream commercial tools showed limited, if any, implementation of the MDA standards with a strong proprietary flavour. That is, service providers are currently bound to limited exploitation of the MDA potential. Furthermore, service providers expend considerably on costly licenses and training in order to put such tools to enterprise-wide use, driving overall costs even higher. This is the reason why open source paradigms, such as Eclipse [10], have recently received significant attention and strong industrial support. In the TMF OpenOSS Catalyst [11] the general benefits of open source were investigated in the direction of driving down OSS development costs. However, OpenOSS did not cover the area of tooling.

This paper specifies high-level principles, which should be complied with by tools for sufficient facilitation of meta-modelling and MDA. These principles get refined into more concrete requirements through a state-of-the-art survey of meta-modelling tools. Based on these requirements, the paper presents IEME, an Eclipse-based open source environment providing key MDA facilities for the development of software systems. IEME brings together Eclipse initiatives that individually implement a certain MDA aspect. Specifically, IEME uses: (i) the Eclipse Modelling Framework (EMF) [12], the Eclipse implementation of MOF, for specifying the abstract syntax of meta-models; (iii) the Graphical Modelling Framework (GMF) [13] for specifying the concrete syntax of meta-models and generating dedicated graphical tools that support both the abstract and concrete semantics of a meta-model; (iv) the Atlas Transformation Language (ATL) [14] for specifying transformation rules among meta-models; and (v) the openArchitectureWare (oAW) [15] for building code generators and specifying constraints on meta-models and models .

The rest of the paper is organised as follows. Section 2 investigates the relationship between meta-modelling and DSL specification and lays out a rigorous set of requirements that an effective MDD tools framework should satisfy. Section 3 conducts a survey of tools with MDD capabilities and evaluates them using the identified requirements. Next, section 4 presents IEME and how it fulfils the identified requirements. Finally, section 5 discusses conclusions and future work.

2 Meta-models as Domain Specific Languages: MDD Tools Framework Requirements

According to Nytun, Prinz and Tveit [16]: *A metamodel is a model that defines a language completely including the concrete syntax, abstract syntax and semantics.*

Another definition describes meta-modelling [17] as: *The construction of an object oriented-model, which represents the abstract syntax of a language.*

Our view on meta-modelling aligns with the former statement. That is, meta-modelling is the process of complete and precise specification of a *domain-specific* modelling language, which in turn can be used to define models of that domain. This treatment places a meta-model one abstraction layer higher than domain models. This way, an indefinite hierarchy of abstraction layers can be built, where models at layer *n* are specified using the precise semantics of the language defined as a meta-model at layer *n+1*. In this setting, models situated at layer n are instances of meta-models at layer n+1.

MDA provides such a layered architecture limiting the number of abstraction layers to four. At the top level, M3, the meta-meta-model of MOF is situated, which provides a generic language for the definition of domain-specific languages. Layer M2 is populated by meta-models that represent MOF-defined domain-specific languages. Layer M1 hosts domain models written in M2-defined DSLs. Finally M0 hosts runtime domain objects that instantiate M1 domain entities.

Due to their focus on a certain domain, as opposed to generic modelling languages such as UML [18], it is necessary to produce a precise definition for the domain specific semantics of a DSL. This requires a DSL specification paradigm that can adequately facilitate the rigorous definition of the DSL abstract syntax, comprising a meta-model of the domain-specific concepts and constraints for precisely defining the domain concepts semantics. MDA is such a paradigm, as it provides MOF for meta-model specification and OCL for the specification of meta-model constraints. Furthermore, MDA can facilitate the definition of mappings between DSLs using QVT and the exchange of meta-model data using XMI.

In order to render practical the use of a DSL, a high-level notation should be available, allowing designers to produce models in this DSL. Therefore, alongside its abstract syntax, a DSL should encompass a concrete syntax definition specifying the way DSL abstract concepts can be represented within a design in a consistent manner. For easier use of the language, such DSL concrete syntax may be specified through a graphical notation, which can drive the development of DSL-specific graphical modelling tools. Developing such DSL tools can be a laborious and costly process, especially when considering the need of these tools to constantly evolve alongside modifications and extensions the DSL may incur in time. Therefore, looking at automating the process of generating DSL tools can be very beneficial. Automatic tool generation will require meta-tools, ie. more abstract DSL tool specification environments, providing a framework of meta-modelling and graphical facilities to precisely specify DSL abstract and concrete syntax.

All aforementioned features, also grounded on our previous work and MDA case-studies [5], [6], [7], [8], [9], describe general principles for the way we view MDD process applying in practice and the facilitation we believe is required by way of a MDD supporting tools framework. These principles are shaped into a specific set of requirements necessary to render a MDD framework practically efficient. The requirements are outlined below:

[R1] *Abstract syntax:* Any DSL shall be specified as a M2 meta-model using a semantic meta-meta language, such as MOF. An effective MDD framework must ensure completeness of the new modelling language through its meta-meta language.

[R2] *Concrete syntax:* A DSL shall additionally specify a notation, preferably graphical, to allow the concrete representation of its abstract concepts. This will enable better understanding of the language and will make its use easier in developing domain models.

[R3] *Meta-model level constraints:* Precision in the DSL semantics shall be provided by the specification of constraints onto the M2 meta-model (DSL abstract syntax) to ensure correctness of the language.

[R4] *Domain-specific modelling tools generation:* One to one mapping must be enabled between the DSL abstract concepts and their corresponding concrete representation, which shall lead to the generation of a DSL modelling tool environment. The tool will be used for the specification in DSL and management of M1 domain PIMs.

[R5] *Model level constraints:* It shall be possible to specify constraints onto the actual M1 domain PIMs. Therefore, domain-specific tools must provide a constraints specification facility.

[R6] *Model Transformations:* It shall be possible to transform a PIM to another PIM or PSM. This must be driven by mapping rules defined at M2 between the meta-models which represent the abstract syntax of the DSLs used to specify the original and resulting M1 domain models. The mapping rules should be embedded in the generated DSL modelling tool. In its M1 model manipulation capacity, the domain modelling tool should be able to execute the transformation but should provide no facility to change the mapping rules.

[R7] *Text-based generation:* A MDD framework shall generate text-based output from M1 domain models. This can lead to code generation in a programming, such as Java, or a markup language, such as XML.

[R8] *Standards Conformance:* Any MDD and supporting tools framework should be conformant to OMG's MDA standards, namely, MOF, XMI, QVT and OCL.

[R9] *Accelerated adoption:* Generated tools should be easy to use by the designers. In the context of this paper, we will restrain to assessing accelerated adoption by the extent the environment of the generated tool is compatible to a widely adopted and used development environment, such as Eclipse.

From a conceptual perspective, the aforementioned requirements drive a certain flow of steps, a way of working in other words, that characterise a structured and practically effective MDD process. Figure 1, illustrates thoroughly the proposed MDD process flow over OMG's 4-level architecture and the way each requirement matches the consecutive flow steps.

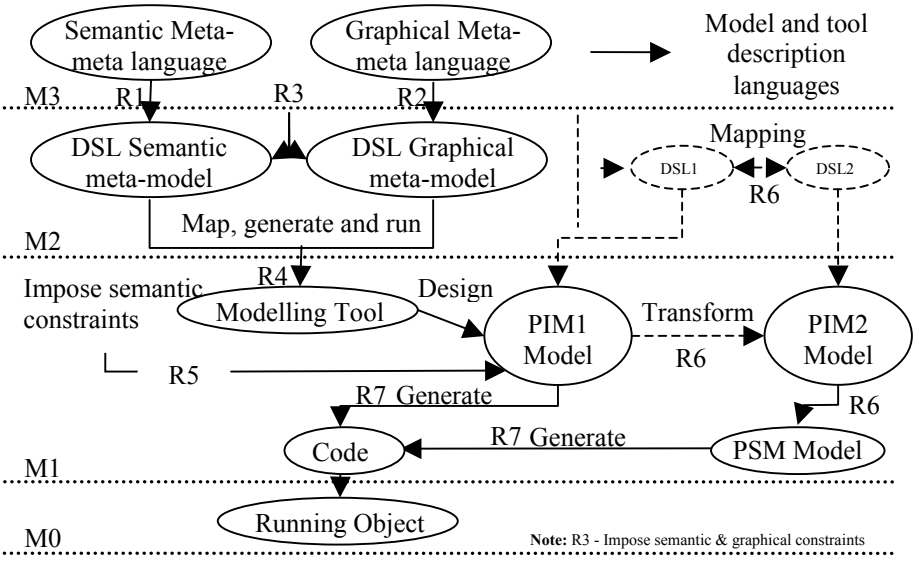

Fig. 1. Meta-modelling and frameworks

From a practical perspective, the requirements introduce a two-layered MDD tools framework, as illustrated in Figure 2. The top layer refers to meta-tool environments with capacity to specify abstract and concrete syntax of a DSL, meta-model level constraints and DSL transformations. The meta-tools generate graphical modelling tools, which occupy the lower layer and conform to the DSL specifications. The offspring tools can be used to develop domain models using the DSL, to specify domain model constraints and to automatically generate out of the domain models either code or other PIMs, as dictated by the embedded meta-model transformations.

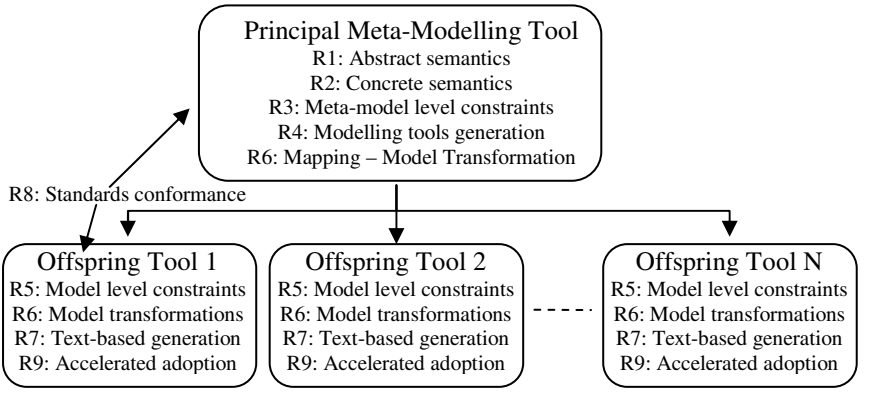

Fig. 2. Generation of offspring modelling editor tools

3 Comparative Study of MDD Tool Frameworks

In this section we provide an extensive discussion on major meta-modelling frameworks. We check how these conform to the requirements set and to the procedures outlined in Figure 1 and Figure 2. It must be denoted that those requirements were extracted based on our experience from previous case studies [8], [9] performed using such frameworks (see XMF-Mosaic). The frameworks under study are two research tools; (i) Generic Modelling Environment (GME) [19], (ii) DOmain Modelling Environment (DOME) [20], two commercial tools; (iii) MetaEdit+ [21], (iv) XMF-Mosaic [22] and the open source framework project AndroMDA [23].

The first four frameworks selected are considered as the most suitable ones for contacting the survey. This is due to the fact that these are the most dominant ones and provide extensive *Domain Specific Modelling (DSM)* support; DSM Forum [24] supports also this idea. Although AndroMDA falls out of the DSM Forum group of tools is used in the context of this paper to demonstrate the high value of an open source project. DSM Forum focuses on DSL development and expresses the necessity for production of DSL supporting tools. DSM Forum industrial experiences and case studies [25] revealed that the use of DSM supplies the aforementioned benefits. Furthermore, related work [26] yet again acknowledged that DSM in conjunction with the MDA paradigm increase productivity significantly. The major problem though with such an approach was the inability of frameworks to generate the appropriate modelling tools. Nowadays meta-modelling frameworks have significantly improved and provide many of the required capabilities to the developers.

Primarily a framework needs to supply a precise meta-meta modelling language for the production of DSLs. This should cover both the definition of abstract properties and their concrete graphical representation. GME supports a proprietary meta-modelling language called MetaGME [19] that is based on UML class diagram notation for the creation of new DSLs. Meta-models and models are represented and can be imported/exported using an XML format. MetaGME allows additionally definition of meta-model level constraints compliant with OCL 1.4. MetaGME's OCL implementation allows the developer to generate a consistent DSL and its corresponding modelling editor. Additionally the framework allows the definition of OCL constraints at the model level for checking low-level model attributes.

GME does not provide any explicit support for defining and executing model-to-model transformations. The code generation functionality is restricted since it only allows the developer to integrate its on generator as an API add-on. GME has been recently incorporated into a new Eclipse project called Generic Eclipse Modelling System (GEMS) [27]. Its goal is to bridge qualified meta-modelling projects, such as GME, with the Eclipse platform and its related modelling projects; EMF and GMF. The aim is similar to the integration we have performed by bridging several Eclipse based modelling projects together to form an effective meta-modelling environment.

DOME has its own proprietary tool specification language [28], which relies on concepts similar to UML. It covers abstract semantics definition but provides only basic support for concrete semantics since the graphical appearance cannot be edited in a visual manner. Although the graphical representation is not very powerful it still enables tools generation for the defined DSLs. DOME does not support explicitly the OCL language but provides built-in support for certain types of frequently used

meta-model level constraints; with the Alter language. Additionally Alter allows the developer to build code generators on the basis of the domain models defined. Concerning model transformations the tool does not provide any support. Models and meta-models are also expressed using XML syntax.

MetaEdit+ includes several tools that compose its MetaEngine and provides in overall a framework that minimizes the developer's workload. It implements a meta-meta language called GOPRR [29]. Each letter corresponds to an element of the language. The framework allows defining both the conceptual and graphical properties. From the DSL definition the modelling tool can be automatically generated including facilities such as diagramming editors, browsers and generators. Constraints can be also defined as data incorporated onto the meta-model definition. GOPRR meta-modelling definition enables MetaEdit+ to identify several rules from which a user can select the most appropriate ones. The user can even alter those rules to conform better to its requirements. Framework's support for model level constraints definition is limited. It provides also a Generator editor that facilitates both basic model-to-model transformations and code generation. The framework grants to the developer the ability to integrate its own model transformation engine or code generator package as an API add-on. In general it facilitates many of the requirements and it seems that its future direction aims towards the integration of the entire set of those features.

Table 1. Meta-modelling frameworks requirements conformance

RS	Tool features	GME	DOME	MetaEdit+	XMF-Mosaic	AndroMDA
R1	**Abstract semantics**	UML MetaGME	√	GOOPPR	XCore	UML 1.4 or MOF XMI
R2	**Concrete semantics**	UML MetaGME	Partially	√	√	×
R3	**Meta-model level constraints**	OCL 1.4 in MetaGME	Alter language	As data in meta-model	XOCL	OCL translated to Java, EJB-QL and HQL
R4	**Modelling tools generation**	√	√	√	√	×
R5	**Model level constraints**	OCL 1.4	×	Limited support	XOCL	×
R6	**Transformations**	×	×	Generator Editor or add-ons	XMap language	Defined in Java
R7	**Code generation**	As GME add-ons	Alter language	Generator Editor or add-ons	XMap language	Template-based
R8	**MOF Compliant**	Proprietary language	Proprietary language	Proprietary language	√	Limited support
	XMI-Compliant	XML format	XML format	XML format	XML format	√
	QVT Compliant	×	×	×	√	×
	OCL Compliant	√	×	×	Executable OCL	Limited support
R9	**Accelerated adoption**	×	×	×	Eclipse build	×

XMF-Mosaic is the last framework of the DSM Forum group studied in this survey, which is build onto the Eclipse platform. It incorporates all requirements identified and illustrated onto Figures 1 and 2. Table 1 also reveals that fact and displays the features each tool provides. XMF-Mosaic uses XCORE [30] as its meta-modelling language, which is based on the Meta Object Facility (MOF) [31] specification. XCORE is used for the specification of the meta-model properties. It also supplies tools like XBNF and XTools that facilitate the representation of the meta-model concepts into a so-called user interface model. XBNF is a grammar language and XTools is used to map domain concepts graphically. Additionally XTools specifies the tooling for the user interfaces of the generated modelling editor. With the use of two powerful languages XMap and executable OCL (XOCL) it fulfils the aspects of constraint checking, model transformations and text-based generation. XMap is used to define the mappings between DSLs to perform the model transformation and additionally can be used to define code generators. XOCL enables constraint definition on both meta-models and models. XMF Mosaic is a very powerful meta-modelling framework, which conforms to OMG specifications more than any other framework. Despite that fact XMF-Mosaic is a commercial product, which is not freely available and cannot be extended or modified unless the company releases a new version.

This is where the open source software community comes into action, since it covers the additional aspect of extensibility and subtracts software licence costs. AndroMDA [31] is a very good example of an open source extensible generator framework that adheres to the MDA paradigm. Its core features endeavour most of the requirements. It provides UML 1.4 meta-modelling language support and alternatively it allows using your own MOF XMI meta-model. Comes with pre-configured OCL constraints and allows the addition of own specific project constraints, which are translated into Java, Hibernate Query Language (HQL) and Enterprise JavaBeans Query Language (EJBQL) validation code. Constraints are enforced onto the meta-models. Additionally it provisions to define model-to-model transformations using Java and it is planned as part of the next major release of AndroMDA to provide support for the powerful QVT based ATL language. It provides also generation of text-based output using well known template engines.

Another valuable aspect of AndroMDA, along with the fact that it is open source, is its modular design that supplies a plug-in based architecture. This allows the developer to compose its own environment from various project blocks to suit specific requirements and needs. AndroMDA covers most of the requirements but it lacks support of the essential facet of modelling tools generation, which is an imperative feature of an MDA framework with DSM support. Furthermore, current support for constraint validation is deficient and model-to-model transformation using Java is not as powerful as with the use of a QVT based language. Finally setup of the development environment for AndroMDA and configuration of its building blocks can be quite tedious and troublesome. Due to that fact, an Eclipse-based integration is one of the primary objectives set by the AndroMDA project and is currently under the development process.

Eclipse open source platform provides solutions to the issues identified from the evaluation of the aforementioned frameworks and guides the effort in the formulation of a coherent MDA environment. Recognising its tremendous impact on the industrial world (e.g. XMF-Mosaic is build on the Eclipse platform) and the high-value of its

related modelling projects we proceeded in formulating an open source environment, which integrates the necessary meta-modelling features. The environment is composed by several Eclipse projects in an intuitive manner that allows efficient application of model driven development. Further reference to the environment in this article is done in terms of its devised name that is Integrated Eclipse Modelling Environment (IEME).

4 Integrated Eclipse Modelling Environment

The development of IEME was mainly driven by the need to produce a meta-modelling framework satisfying the requirements identified in section 2. Due to the interest in keeping IEME an open source environment, relevant MDA project initiatives of the Eclipse platform were considered. These initiatives were carefully evaluated in practice as per their ability to best meet the aforementioned requirements and were suitably tailored and integrated in a coherent environment.

There are many reasons for choosing Eclipse as the platform for creating the framework. Primarily is its wide acceptance amongst developers and the fact that it is an extensible open source development platform (*R9*). This provides the ability to the developer to modify any of its features or tools and extend or add new tools to serve its company specialised needs. Additionally it offers the possibility to integrate IDEs such as Java and C/C++ that facilitate in increasing software systems productivity.

Figure 3 shows how the MDD process requirements are mapped accordingly to the capabilities of the integrated environment. Initially with the use of the Graphical Modelling Framework (GMF) diagram editor the domain meta-model can be defined

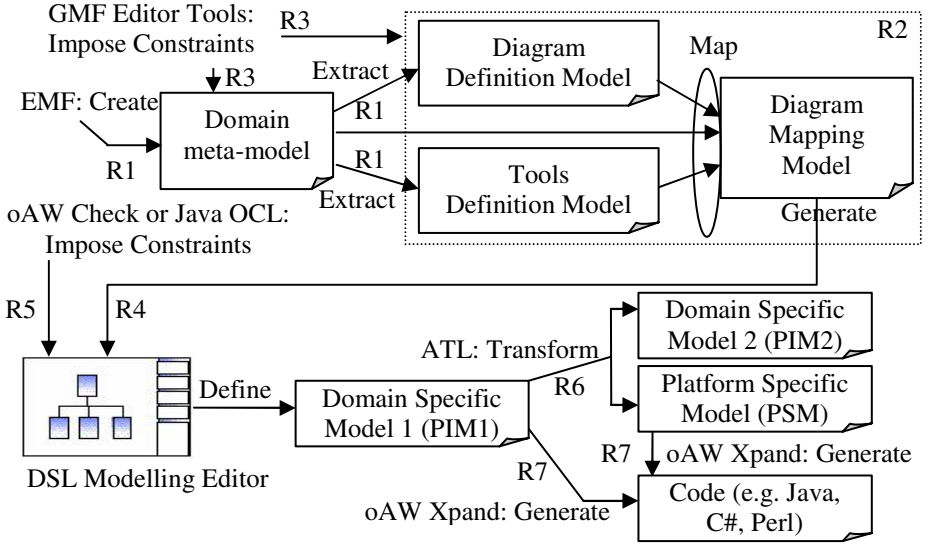

Fig. 3. Model driven development using the IEME

using the *ECore meta-meta modelling language* (*R1*). Subsequently, the diagram and tools definition models can be extracted automatically from the domain meta-model. GMF translates the abstract syntax of the domain meta-model and generates the concrete syntax models. These are the *Diagram Definition Model (R2)*, which presents the diagrammatic figures and the *Tools Definition Model (R2)*, which presents the palette element tools. Both models represent the graphical elements of the generated modelling editor tool. The models can be further enhanced through the use of the GMF modelling editor tools provided.

Binding of the abstract and concrete syntax produces the *Diagram Mapping Model (R2)*, which includes the distinct concepts found in the three models. The generation model is extracted from this mapping model and the Eclipse Modelling Framework (EMF) Java Emitter Templates (JET) generation engine is used to generate the *DSL Modelling Editor* (*R4*) of the domain language. The modelling editor can be used to define graphically application domain specific models with their semantics set accordingly through the use of the properties view of the editor. Models can be further improved by the imposition of constraints using the *openArchitectureWare (oAW) Check language* or by *applying OCL statements through Java* (*R5*). It must be denoted that *GMF modelling editor tools* (*R3*) provide capabilities for assigning both graphical and semantics constraints at the meta-model level during the editor's development process.

Atlas Transformation Language (ATL) is another component that can be used to define model-to-model transformations (*R6*). The transformations are written in an ATL textual editor and are purely based onto the meta-model. Transformations can be from a platform independent model (PIM) to another platform independent model or even a platform specific model (PSM). Thus the only remaining aspect is the generation of the implementation from the model. The framework grants that capability with the use of the *oAW Xpand language* (*R7*). Actually the generator is build out of templates written in the XPand language something that enables code generation in any possible language.

Table 2. IEME framework characteristics conformance

Reqs	Tool features	IEME
R1	**Abstract semantics**	ECore (EMF) – ECore Diagram (GMF)
R2	**Concrete semantics**	GMF (GMFGraph, GMFTool, GMFMap, GMFGen)
R3	**Meta-model level constraints**	GMF modelling editor tools
R4	**Modelling tools generation**	EMF JET Engine
R5	**Model level constraints**	oAW Check language or Java
R6	**Transformations**	Atlas Transformation Language (ATL)
R7	**Code generation**	oAW Xpand language
R8	**MOF Compliant**	EMF – implementation of MOF
	XMI Compliant	XMI meta-models and models
	QVT Compliant	ATL – implementation of QVT
	OCL Compliant	GMF, oAW Check, Java OCL
R9	**Accelerated adoption**	Build on the Eclipse Platform

The procedure described before and illustrated in Figure 3 shows that IEME is in fact a complete and coherent MDA/MDD framework. Additionally Table 2 presents how the framework reflects each of the requirements set and attests to the claim that it provides a pure meta-modelling environment. Following we give a more detailed overview of the core components of the environment to enable better understanding of its features and capabilities. It must be stated that the framework provides Java and C/C++ IDEs that are also built-in the environment using the extensible Eclipse plug-in architecture. IEME is composed by the following modelling core components delivering a versatile environment.

Eclipse Modelling Framework (EMF) [12] is the core of the environment. It's an extended implementation of MOF and lies at the meta-meta level. It is a modelling framework and code generation facility that serves as the meta-meta language for defining domain meta-models (ecore). EMF supplies its own tree-based editor for meta-model definition. The domain meta-model includes the semantics of the defined DSL. Furthermore it provides the facilities that are essential for the automatic generation of the corresponding tree-based editor tool. This tool enables the creation of models of the defined domain specific language.

Graphical Modelling Framework (GMF) [13] provides a generative component and runtime infrastructure for developing graphical editors. GMF provides graphical editor tools that allow the definition of the visual domain meta-model (ecorediagram), the diagram definition model (gmfgraph), the tooling definition model (gmftool) and the mapping model (gmfmap). The visual domain meta-model is the diagrammatic view of the domain meta-model, which provides better understanding of the defined DSL. Therefore definition of the visual domain meta-model using the GMF editor is preferred than the definition of the domain meta-model using the EMF tree-based

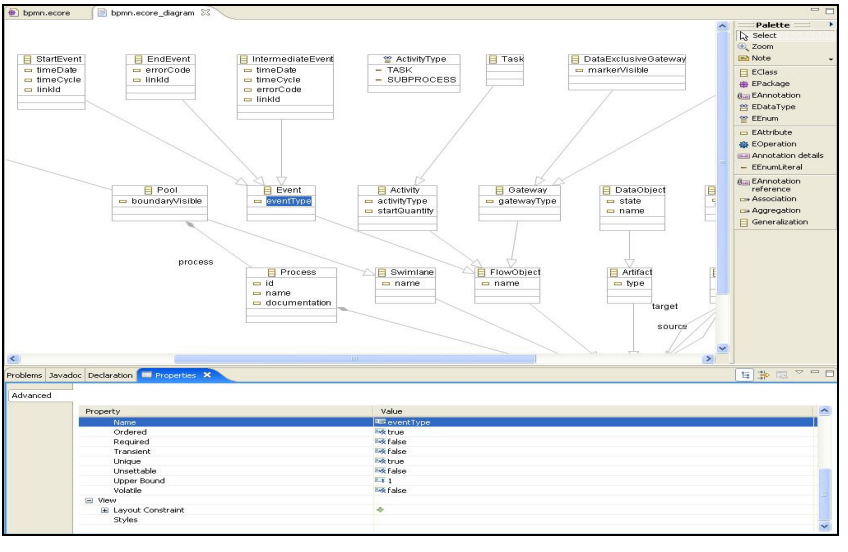

Fig. 4. BPMN Meta-model definition using GMF graphical editor

editor. Figure 4 shows an example of a visual domain meta-model definition using the GMF editor. It represents the domain meta-model semantics of the Business Process Modelling Notation (BPMN).

EMF and GMF combined and used in parallel provide the essential meta-meta languages and the supporting tools that drive the entire language development process. Most importantly they facilitate the generation of the appropriate tools to support the modelling language. Both projects use XMI as the common syntax for their models. Next we present the projects that cover the rest of the requirements namely constraint checking, model transformations and code generation. The entire set of core components are plug-ins integrated into the Eclipse platform something that preserves the stability and extensibility of the overall environment.

Atlas Transformation Language (ATL) [14] is a project developed at INRIA French National research institution and aims to provide model-to-model transformations. ATL is also an Eclipse plug-in that implements the Query/View/Transformation (QVT) language standard. ATL is actually a domain specific language build for transformations. Basically it allows defining model-to-model transformations with the use of appropriate editor tools. It must be denoted that the transformations are defined based on the meta-models. Currently ATL is adopted as the basic component, along with the QVT standard, of the Eclipse Model-to-Model transformation (M2M) project. There is a variety of defined transformations already available online onto the Eclipse ATL ZOO something that reveals the popularity of ATL as the major transformation language.

openArchitectureWare (oAW) [15] is a framework that provides a set of modelling tools integrated into a coherent model driven development environment. Careful study of the tools developed within this project reveals that it follows exactly the same guidelines and characteristics for working in the context of MDA. Additionally it provides the capability to choose selectively components from the overall framework since these are built as Eclipse plug-ins. Some of the components of the project, which perform specific tasks are not quite as powerful as their counterparts; EMF, GMF and ATL. Therefore these were not selected. oAW though contains other powerful tools with their accompanying languages that can assist in the complete and coherent integration of an MDA framework.

Foremost is the Xpand template language, an extract of which is shown onto Figure 5, which supports advanced features for building code generators in any programming language. It must be denoted that the environment provides an alternative code generation facility, which is EMF Java Emitter Templates (JET) [32], [33]. Although JET is very powerful itself and can be used to generate code in any language, it is more focused and best to use for generating Java code. Therefore Xpand template language is found to be more competent for text-based generation to any particular language.

Another component is the Check language, which is an OCL based language that allows definition of constraints onto the EMF meta-model and directly onto the models. oAW has strong support for EMF based models but can work also with other models (e.g. UML2). Additionally it even allows the definition of OCL constraints using Java. A core workflow engine controls the generator's workflow, as specified in an XML format. The workflow definition drives the execution by invoking the corresponding components for reading and instantiating models, checking for constraint violations and then finally, for generating code.

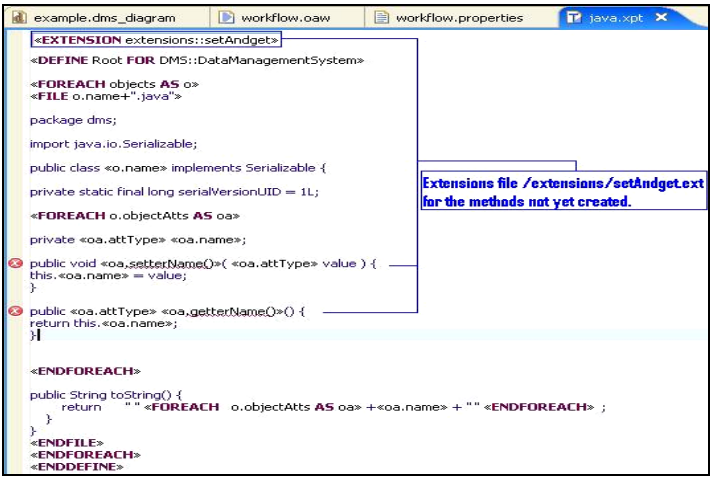

Fig. 5. Building code generators using XPand template language

5 Conclusions and Future Work

In this article we stress out the importance of tools for industrial use of meta-modelling frameworks. Such type of frameworks should provide the ability to define and generate modelling tools. Additionally they must comply with some other fundamental requirements. There are various initiatives to formulate a coherent environment and each of them comprise of some powerful tools. Study of those frameworks exposed that each initiative follows the same guidelines to deliver these essential features. Despite that fact none of them provides a complete solution to the problem. The commercial XMF-Mosaic covers most of the important aspects but it also lacks in terms of the generation of stable modelling editor tools using XTools.

IEME is our proposition of a framework that covers all these aspects by integration of the most powerful tools and languages. It must be denoted that there might be other dominant tools and languages that also fulfil their specific goals. The most important reason for the selection of these is their smooth ability to integrate as plug-ins to the extensible open source Eclipse platform. Eclipse platform provides the flexibility and the dynamics required by such an environment. A developer might require adding or altering a facility to suit its specific company requirements. It is no secret that major frameworks and industrial organisations use Eclipse as their development platform.

There are also other benefits provided by the integrated environment. These are namely its precise meta-meta language that is based on OMG MOF, its support for a common interchangeable syntax such as XMI. Someone might argue that there are so many versions of XMI that interoperability is not easy to achieve. The attempt though is to at least provide a common widely acceptable syntax for the models in order to be compliant with other frameworks that adopt this standard syntax. Furthermore EMF and GMF projects are now very mature offering remarkable capabilities for the generation of stable and user friendly modelling tools with a simplistic procedure.

Constraint checking, transformations and code generation are covered in great depth by the appropriate projects.

An important aspect that was identified and remains to be tackled, as future work, is the configuration management when porting to newer versions of the modelling language. Transition to another meta-model by incarcerating new semantics requires adjusting the modelling language's editor tools. Therefore it can be realised that models designed using the new version of the language are dissimilar to previous models. The framework needs to provide the capability to convert the latter to reflect the improved modelling language. An initial consideration is the usage of the powerful ATL transformation language to define transformations between the different versions of the models. Such an improvement to the framework will be very beneficial reducing further transition costs.

The environment is an initial effort to provide a complete solution to the MDA paradigm, covering all necessary aspects. Further research and testing of the environment with examples and case studies will detect any deficiencies to further improve it. Since the environment is characterised by its extensibility and adaptability such alterations can be easily implemented. Finally another important aspect that remains open is the interoperability of IEME with the rest of the major frameworks presented in this article. MDA growth requires this interoperability amongst powerful frameworks since it will provide the ability to the developer to make use of the most appropriate tools to accomplish its objectives.

References

[1] The OSS through Java Initiative, [Online] Available: http://www.tmforum.org/browse.aspx?catID=2896

[2] The TeleManagement Forum, [Online] Available: http://www.tmforum.org

[3] Model Driven Architecture (MDA), Specification Guide V1.0.1, Object Management Group (OMG), [Online] Available: (June 2003), www.omg.org/docs/omg/03-06-01.pdf

[4] Object Constraint Language (OCL) Specification, version 2.0, Object Management Group (OMG), [Online] Available (June (2005), http://www.omg.org/docs/formal/06-05-01.pdf

[5] Ou, S., Georgalas, N., Azmoodeh, M., Yang, K., Sun, X.: A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, Springer, Heidelberg (2006)

[6] Azmoodeh, M., Georgalas, N., Fisher, S.: Model-driven systems development and integration environment. In: BT Technology Journal, vol. 23(03), Springer, Berlin Heidelberg (2005)

[7] Georgalas, N., Azmoodeh, M., Ou, S.: Model Driven Integration of Standards Based OSS Components, In: Proceedings of the Eurescom Summit on Ubiquitous Services and Application, Heidelberg, Germany (2005)

[8] Georgalas, N., Azmoodeh, M., Clark, T., Evans, A., Sammut, P., Willans, J.: MDA-Driven Development of standard-compliant OSS components: the OSS/J Inventory Case-Study, In: Proceedings of the Second ECMDA with emphasis on Methodologies and Transformations, Canterbury, UK (September 2004)

[9] Georgalas, N.: NGOSS/MDA: Realising NGOSS as a Model Driven Approach, Catalyst project, TeleManagement World Conference, Nice, France (2005)

[10] Eclipse – an open development platform, [Online] Available: http://www.eclipse.org/
[11] The OpenOSS Programme, TeleManagement Forum, [Online] Available: http://www.tmforum.org/browse.aspx?catID=2602&linkID=31021
[12] Eclipse Foundation Inc. Eclipse Modelling Framework (EMF), [Online] Available: http://www.eclipse.org/emf/
[13] Eclipse Foundation Inc. Graphical Modelling Framework (GMF), [Online] Available: http://www.eclipse.org/gmf/
[14] INRIA Research Institution, Atlas Transformation Language (ATL), [Online] Available: http://www.eclipse.org/m2m/atl
[15] openArchitectureWare.org, openArchitectureWare (oAW), [Online] Available: http://www.eclipse.org/gmt/oaw
[16] Nytun, J.P., Prinz, A., Tveit, M.S.: Automatic Generation of Modelling Tools. In: ECMDA-FA: Proceedings of Second European Conference, pp. 268–283. Springer, Berlin/Heidelberg (2006)
[17] Greenfield, J., Short, K., with contributions by Cook, S., Kent, S.: Software Factories: Assembling Applications with Patterns, Frameworks, Model and Tools. John Willey and Sons, New York (2006)
[18] Unified Modelling Language (UML), version 2.0, Object Management Group (OMG), [Online] Available: (June 2004), http://www.omg.org/technology/documents/formal/uml.htm,
[19] Vanderbilt University, A Generic Modelling Environment, GME 5 User's Manual, [Online] Available: http://www.isis.vanderbilt.edu/projects/gme/GMEUMan.pdf
[20] Honeywell Labs, DOmain Modelling Environment (DOME), [Online] Available: http://www.htc.honeywell.com/dome/index.htm.
[21] Metacase, MetaEdit+ Version 4.5 User's Guide, [Online] Available (2006), http://www.metacase.com/support/45/manuals/meplus/Mp.html
[22] Xactium, Language Driven Development and XMF-Mosaic, White papers, [Online] Available: (March 2005)
http://whitepapers.zdnet.co.uk/0,1000000651,260134763p,00.htm,
[23] AndroMDA.org, Open Source MDA Generator Framework, [Online] Available: http://www.andromda.org
[24] DSM Forum.org, [Online] Available: http://www.dsmforum.org/
[25] DSM Forum.org, DSM Case Studies and Examples, [Online] Available: http://www.dsmforum.org/cases
[26] Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J., Neema, S.: Developing Applications Using Model-Driven Design Environments. IEEE Computer Society Journal, Vanderbilt University (2006)
[27] SourceForge.net, Generic Eclipse Modelling System (GEMS) User's Guide, [Online] Available: 540131&big_mirror=0 (1159), http://downloads.sourceforge.net/gems/gems_user_guide_2_0_5_01.pdf?modtime=,
[28] Honeywell Labs, DOME User's Guide V.5.2.2, [Online] Available (1999), http://www.htc.honeywell.com/dome/DOMEGuide.pdf
[29] Metacase, MetaEdit+ Version 4.5, The Graphical Metamodelling Example, [Online] Available (2006),
http://www.metacase.com/support/45/manuals/Graphical%20Metamodelling.pdf
[30] Xactium, Applied Meta-modelling: A Foundation for Language Driven Development, Version 0.1, [Online] Available (2004), http://www.securewebonline.com/Services/AppliedMetamodellingV01.pdf

[31] Meta Object Facility (MOF) Core Specification, version 2.0, Object Management Group (OMG), [Online] Available (January 2005), http://www.omg.org/docs/formal/06-01-01.pdf
[32] Azzurri Ltd. JET Tutorial Part 1, [Online] Available: http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html
[33] Azzurri Ltd. JET Tutorial Part 2, [Online] Available: http://www.eclipse.org/articles/Article-JET2/jet_tutorial2.html