Automating the Interoperability of Conceptual Models in Specific Development Domains

Oscar Pastor, Giovanni Giachetti, Beatriz Marín, and Francisco Valverde

Abstract An increasing number of modeling approaches for representing concepts related to a wide variety of domains can be clearly observed in software engineering. In this context, the definition of sound interoperability mechanisms to reuse knowledge and share ideas among existing conceptual models, and also apply them into concrete development processes, is an important challenge to be faced. Thus, different modeling approaches, tools, and standards can be integrated and coordinated to reduce the implementation and learning time of development processes as well as to improve the quality of the final software products. However, there is a lack of approaches to support automatic interoperability among modeling approaches. For tackling this situation, this chapter presents an interoperability approach focused on the characterization of different modeling approaches in a common software development domain. For putting in practice and automate the interoperability the approach proposed, existing modeling technologies and standards are considered. All these elements comprise a reference interoperability model, which can be used to implement specific interoperability solutions.

Keywords Conceptual modeling • Interoperability model • Interoperability process • Literature review • Model-driven interoperability

O. Pastor (✉) • F. Valverde
Centro de Investigación en Métodos de Producción de Software (PROS), Universitat Politècnica de València, Valencia, Spain
e-mail: opastor@dsic.upv.es; fvalverde@pros.upv.es

G. Giachetti
Facultad de Ingeniería, Escuela de Informática, Universidad Andres Bello, Sazié 2325, Santiago, Chile
e-mail: ggiachetti@unab.cl

B. Marín
Escuela de Ingeniería Informática, Facultad de Ingeniería, Universidad Diego Portales, Santiago, Chile
e-mail: beatriz.marin@mail.udp.cl

I. Reinhartz-Berger et al. (eds.), *Domain Engineering*,
DOI 10.1007/978-3-642-36654-3_14, © Springer-Verlag Berlin Heidelberg 2013

1 Introduction

According to the IEEE Standard Computer Dictionary [44], interoperability is defined as *the ability of two or more systems or components to exchange information and to use the information that has been exchanged*.

Interoperability is a key aspect to be faced in many different areas where interchange of information and knowledge is necessary to achieve the intended objectives. This is a well-known fact in the electronic device world, where interoperability is critical for positioning and commercializing new technologies. Nobody thinks of changing the DVD player depending on the DVD brand to play.

Other example can be found in the architecture field, where interoperability is essential for communicating building information among the different actors involved, i.e., among architects, engineers, and finally to the builders, which become the charts specified into concrete and physical structures. This last situation is very close to the software domain reality, where the correct representation of concepts that must be implemented is essential to obtain (build) software products that fit with customer's needs. In the domain engineering world, the definition of suitable conceptual elements related to a specific development domain is the Rosetta stone for depicting and obtaining appropriate software models (software charts).

The necessity of counting with interoperability mechanisms in the current software development context is a growing trend. For instance, we can observe interoperability in web applications where different web services must be coordinated to perform a specific operation. Also, interoperability becomes necessary for geographically distributed software factories that are developing different components of a same software product [16].

However, the interoperability of conceptual models presents different unsolved challenges, such as the definition of concrete mechanisms for automation and verification of interoperability operations. Moreover, models defined, even with a same modeling language, are not compatible among different modeling tools. UML is a clear example of this situation. Probably, the large variety of modeling approaches and related technologies is mainly responsible for the lack of interoperability among conceptual models. Therefore, we need to deal with this model heterogeneity since the modeling representations that can be defined in the conceptual modeling domain are practically infinite.

Thus, in this chapter, we propose a conceptual model for the definition of concrete interoperability solutions, which deals with model heterogeneity aspects and faces automation issues. This interoperability model has been instantiated with current standard and model-driven technologies to obtain a suitable solution for concrete model-based development processes.

The rest of this chapter is organized as follows: Sect. 2 presents the related work in the context of model-driven interoperability. Section 3 details the interoperability model proposed. Section 4 presents the challenges that we have faced in the definition of this interoperability model. Section 5 introduces the process to put into practice the interoperability model. Finally, Sect. 6 presents a general analysis of the interoperability proposal presented and main conclusions.

2 Review of Model-Driven Interoperability

One of the main concerns of the domain engineering field is the definition of suitable constructs and conceptual models for performing the analysis of specific application domains [26]. In this context, the reuse of domain knowledge and its application into concrete software production process is a relevant issue to be considered. The use of conceptual models, domain-specific representations, and model reuse strategies that are part of the domain engineering principles are also subject of interest in the model-based software production context. For instance, the application of domain analysis approaches, which come from the requirements engineering world, into MDD processes [5, 53, 68].

In this chapter, we face the interoperability problem from the model-driven perspective, thus proposing a *Model-Driven Interoperability* approach; i.e., the information exchange involved in interoperability tasks is performed by means of models and model management operations (such as model transformations).

In order to explore the existing evidence of model-driven interoperability, we have reviewed relevant approaches related to different domains, such as conceptual (including domain-specific) modeling, requirement engineering, and business process modeling. [35].

In the revision, the following characteristics are considered:

- The proposal provides mechanisms for *managing the heterogeneity* that may exist among the involved modeling approaches.
- The proposal considers the use of existing *standards* for the definition of models, metamodels, or model transformations.
- The proposal has some kind of *supporting technology* that automates the interoperability operations.
- The proposal provides a well-defined *application process*.
- The proposal defines *mechanisms to verify the interoperability* among the modeling approaches involved.

The studies analyzed are briefly presented as follows. We have grouped the studies according to their main features (some approaches provide more than one relevant contribution), which correspond to the following:

- *Model Weaving* (*MW*): It considers the definition of mappings among the metamodels of involved modeling approaches. These mapping are usually defined by means of a weaving model (instance of a weaving metamodel) [25].
- *Meta-Extensions* (*ME*): This corresponds to the definition of new information (extensions) in the modeling approaches involved to provide additional modeling features that are necessary to perform interoperability operations [77].
- *Interoperability Verification* (*IV*): Models and interoperability operations may have issues that produce an incomplete interchange of information. Thus, it is important to analyze verification mechanisms for assuring the correct specification of the artifacts that are involved in interoperability scenarios [34].

- *Pivot Artifact* (*PA*): This feature is related to the use of an intermediate artifact (metamodel or ontology) to manage structural differences and to identify conceptual equivalences [42].
- *Application Domain* (*AD*): Indicates the domain that is related to the approach analyzed. Despite this, the contributions of the proposal analyzed can be generalized to different domains.

2.1 Model Weaving

The model weaving approach is very popular in model transformation and model interchange contexts. Weaving models indicate semantic equivalences through the definition of links among the metamodels' constructs. These links are considered as semantic connection points because they indicate those constructs (from the involved metamodels) that have an equivalent meaning (semantics) in the application domain. The definition of these links can be extended with additional information defined to manage structural differences among the linked constructs.

Fabro and Valduriez in [25] propose the use of weaving models between two metamodels to automatically infer model-to-model (M2M) transformations based on the ATL tool [46]. These transformations automate the translation between models defined with the involved metamodels.

The proposal presented in [49] by Kappel et al. is defined to support the interoperability among modeling tools. This proposal is based on a bridge metamodel (weaving metamodel) for the definition of semantic metamodel links (bridges). The defined bridges are used to transform the involved metamodels into equivalent petri-net representations. The petri-net representation is used to operationalize M2M transformations and to perform a formal verification of the structural differences (heterogeneities) among metamodels, which may produce interoperability conflicts.

The proposal presented by Klar et al. in [50] shows how the MDD interoperability can be used to support a complete development process. In particular, this proposal is centered on the integration of requirements modeling into MDD processes. However, it does not consider how to integrate specific aspects related to a particular MDD process into the requirements approach. These MDD aspects are necessary to obtain an appropriate requirement specification in the domain of the reference MDD approach.

The proposal presented in [36] by Guerra et al. consists of a pattern-based approach for defining bidirectional relations (a weaving model) among modeling approaches. The main contributions of this proposal are the definition of specific inter-modeling patterns, which allow interoperability conflicts to be automatically identified. These patterns also facilitate the generation of model-to-model transformations, model matching, and traceability information.

2.2 Meta-Extensions

Seifert et al. in [75] indicate the advantages of using metamodels and model-to-model transformations to prevent the coupling among tools that must interoperate. This approach analyzes the pros and cons of proactive and retroactive tool integration alternatives. From this analysis, it suggests the use of a role-based metamodeling approach, which involves the extension of the metamodels of tools with specific role information (defined in a role metamodel) to improve the tool interoperability.

The *BIZYCLE* framework applied by Milanovic et al. in [58] is defined to achieve applications and data integration by means of semantic annotations. These annotations are used to identify both semantic and structural conflicts that can be solved in a semi-automatic way.

The work by Tran et al [79] suggests the extension of the modeling approaches that must interoperate to specify the information related to an MDD process.

The proposal presented by Agostinho et al. in [4] introduces an interoperability framework for business networks, which is based on UML for the definition of the involved metamodels. An interesting feature of this approach is the use of UML profiles to manage model heterogeneities and to obtain an appropriate model mapping. This work refers to those transformations that imply a structural change of the involved constructs as *model altering morphisms*.

2.3 Interoperability Verification

Radjenovic and Paige in [70] present an interoperability approach that is based on an initial identification of the issues that may prevent an appropriate model integration. This work considers both structural and behavioral interoperability conflicts. The detection of interoperability issues is performed by means of the transformation of the involved metamodels into a proprietary graph representation, which is called SMILE-X.

The proposal presented by Polgár et al. [69] indicates the need for interoperability in a common development process. In this approach, a reference ontology is used to verify whether or not the involved modeling approaches are in conformance with the target development process.

2.4 Pivot Metamodel

The differences between a pivot metamodel and a weaving metamodel are related to their definition and use. A weaving metamodel is instantiated to represent links among constructs of the involved metamodels. From these weaving models, specific M2M transformation rules can be inferred according to a specific transformation

approach, such as ATL [46], QVT [63], or ETL [51]. However, weaving models do not participate in the execution of the transformation rules. By contrast, a pivot metamodel can be a predefined representation of concepts (or constructs) for the interoperability domain or can be generated from the metamodels of the modeling approaches that must interoperate. Also, a pivot metamodel can be instantiated during the transformation process generating an intermediate pivot model.

The proposal presented in [16] by Bruneliere et al. defines a pivot metamodel to solve conflicts related to the heterogeneity among metamodels of modeling tools. This proposal is also based on current metamodeling standards and modeling tools.

The DUALLY approach presented by Crnkovic et al. [19] shows that the use of a pivot metamodel reduces the complexity of the necessary transformation rules. These rules can be automatically inferred from the pivot metamodel definition.

The proposals presented by Ziemann et al. [86] and Jankovic et al. [45] are related to enterprise modeling. They use the POP* metamodel [60] as pivot metamodel. According to these proposals, the involved modeling approaches must be mapped to the POP* metamodel to determine common interrelation points. Similar approaches are presented by Baumgart [9] in the domain of embedded systems and by Mahé [55] for visualization tools.

Berger in [10] considers the definition of a pivot metamodel that comprises all the conceptual constructs related to the modeling approaches that must interoperate. This pivot metamodel (defined as *generic metamodel* in the paper) is used as an interface among the metamodels of the involved modeling languages, which isolate the mappings from the metamodel heterogeneities. Later, by means of a set of predefined patterns, a model weaving among the involved metamodels is automatically generated to perform model-to-model transformations.

Vallecillo in [80] proposes the generation of a global model for the combination of different modeling approaches. The generation of this global metamodel is based on a viewpoint unification, which intends to comprise the benefits related to three metamodel integration techniques: metamodel extension, metamodel merge, and language embedding. The use of a common model obtained by the integration of the involved modeling approaches is also presented in Coutinho et al. [18]. This proposal is related to organizational modeling.

In [59], Moreno and Vallecillo propose a web development interoperability framework, which is centered on a generic metamodel for web development methods. Thus, by means of QVT transformations, the modeling approaches related to the different development method must be mapped to the reference metamodel. Evidently, due to the use of QVT, the proposal requires that the involved modeling approaches be defined in Meta-object-Facility (MOF)-compliant metamodels.

Diskin et al. [20] propose the generation of a pivot metamodel, which only indicates overlaps among different modeling approaches. This *overlap metamodel* reduces the complexity related to the definition of a big metamodel that covers all the modeling constructs of the involved modeling approaches. However, this proposal is still theoretical and is not supported by tools or standards.

In the work presented by Biehl et al. [12], the relevance of defining a bridge between technical spaces is clearly stated. This technical bridge is oriented to

translating the metamodels of the involved modeling tools into equivalent representations that are defined using a common metamodeling language, i.e., this solves technical interoperability conflicts. Later, structural bridges are defined in the common interoperability space to perform M2M transformations among the metamodels generated by the technical bridges. A similar approach is defined by Jouault and Guéguen in [47]. In this approach, concrete modeling tools are translated into equivalent metamodeling representations, which are defined with a common metamodeling language. The resultant metamodel is called *virtual tool*. A similar view is presented by Brambilla et al. [14], whose work proposes the translation of domain-specific languages (DSL) to equivalent MOF representations.

In addition, in a previous work presented by Lukácsy et al. in [54], the outputs generated by different information sources are transformed into equivalent models (called *interface models*) to perform interoperability among web services. These interface models are expressed in a UML-like language. An improvement to this kind of service-oriented works is presented by Tran et al. in [79]. In this paper, the authors propose a reverse-engineering mechanism to automatically infer model representation from services' views. The inference models are integrated in a view-based modeling framework to perform integration of services. These models are also used to generate code in other implementation platforms by following an MDD process.

The *ModelBus* approach presented by Hein et al. in [40] is oriented to tool interoperability among nodes that participate in a common development scenario. The interoperability is performed by means of a repository of models and modeling services (such as model transformation and model verification services). This approach is based on the original idea of model bus presented in [13], which indicates two important aspects that must be considered to achieve the MDD interoperability: the *functional connectivity* (related to metamodel heterogeneity) and the *protocol connectivity* (related to technical heterogeneity).

2.5 Pivot Ontology

The main difference between pivot ontologies and pivot metamodels is related to their application context and implementation aspects. While a pivot metamodel directly links syntactic elements (abstract syntax), a pivot ontology is based on semantic principles and requires a reference ontological standard, such as OWL and RDFS [49]. These standards only have a partial correspondence with metamodeling facilities (like MOF). Thus, a *lifting* process [48] is required to solve the gap between semantic (ontologies) and syntactic (metamodel) specifications.

Höfferer in [42] presents an interesting analysis related to the use of ontologies and metamodels to achieve the model-driven interoperability. Even though this work is framed in the context of business-process modeling, the analysis and conclusions obtained can be generalized to any model-interoperability approach.

The Sunindyo et al.'s proposal [78] uses a common bus to perform the model-driven interoperability (such as in [40]). This proposal uses ontology mappings to identify common semantic links among different modeling approaches; the definition of these links is guided by a process for automatic discovery of the involved process models.

Roser and Bauer present in [71] an approach that uses an ontology specification (based on OntoMT) as an intermediate model for managing the heterogeneities and similarities among the metamodels of the involved modeling languages. It is also used to reuse the information of already defined M2M transformations and to reduce the complexity related to changes in the versions of the involved metamodels. This approach distinguishes two kinds of model transformations: (1) mappings, which are horizontal model transformations defined at the same abstraction level and (2) refinement transformations, which imply a change from a higher (less detailed) abstraction level to a lower (more detailed) abstraction level.

Berre et al. presents an ontology-based approach related to service interoperability in [11]. Barnickel and Fluegge proposes the idea of sematic mediation at the domain level to improve the efficiency and the effectiveness of the ontology-based interoperability in [8]. The semantic mediation defines a pivot ontology for each involved domain, which groups a set of conceptual schemas. According to these authors, this approach provides a balanced interoperability solution, which is at a middle point between defining ontologies and mappings for each conceptual schema and the definition of a common pivot ontology.

Opdahl in [66] presents a modeling approach that is framed in the context of business processes. It facilitates language interoperability by applying the unified enterprise modeling language (UEML) [43]. This approach requires the translation of the involved DSMLs into the equivalent UEML representation.

Also in the context of business processes, the proposal presented by Costa et al. [17] provides a model-based platform for the enterprise interoperability. This proposal uses a reference ontology to identify semantic equivalences among the information (messages) that must interoperate. This information (defined in an XML format) is extended with annotations to manage heterogeneities in relation to the reference ontology, which are used to perform appropriate model-to-model transformations.

Table 1 summarizes the works analyzed and the main characteristics that each work presents, which correspond to the following: Management of heterogeneity (MH), Use of Standards (US), Tool Support (TS), Application Process (AP), Interoperability Verification (IV), Meta-Extensions (ME), Pivot Artifact (PA), and Application Domain (AD). In the table, letters *Y* and *N* mean *Yes* and *No*, respectively. In the *PA* column, the letter *O* means *Ontology* and the letter *M* means *Metamodel*. According to the review results, the use of a common interoperability space with a unique *Metamodeling Specification* (*MS*) is an aspect that is considered by all the approaches analyzed. Furthermore 24 approaches, which correspond to 72.7 % of the total approaches analyzed, use *Pivot Artifact* (*PA*) for managing structural heterogeneities. This pivot artifact can be defined through a metamodel or an ontology. We recommend the use of pivot metamodels since the definition

Table 1 Summary of reference model-driven interoperability works

Author	Year	MW	MH	US	TS	AP	IV	ME	PA	AD
Agostinho [4]	2011	Y	Y	Y	N	N	N	Y	N	Bussiness networks
Barnickel [8]	2010	Y	Y	N	N	N	N	N	O	Services
Baumgart [9]	2010	N	Y	N	N	N	N	N	M	Embedded systems
Berger [10]	2010	Y	Y	N	Y	N	N	N	M	Conceptual modeling
Berre [11]	2009	Y	Y	Y	N	N	N	N	N	Services
Biehl [37]	2010	Y	Y	Y	Y	N	N	N	M	Tool interoperability
Brambilla [14]	2008	Y	N	Y	Y	Y	N	N	N	Migration DSL to MOF
Brunelière [16]	2010	Y	Y	Y	Y	Y	N	N	M	Modeling tools
Costa [17]	2007	N	Y	N	Y	N	N	Y	O	Business processes
Coutinho [18]	2009	Y	Y	Y	Y	Y	N	N	M	Org. modeling
Crnkovic [19]	2009	Y	N	Y	Y	Y	N	N	M	Component models
Diskin [34]	2010	Y	Y	N	N	Y	Y	N	M	Conceptual modeling
Fabro [25]	2009	Y	N	N	Y	N	N	N	N	Conceptual modeling
Guerra [36]	2011	Y	Y	N	Y	Y	Y	N	N	Conceptual modeling
Hein [40]	2009	N	Y	Y	Y	Y	N	N	M	Tool interoperability
Höfferer [42]	2007	Y	Y	N	N	N	N	N	O	Business processes
Jankovic [45]	2007	N	N	N	N	Y	N	N	M	Enterprise modeling
Joualt [47]	2009	Y	Y	N	N	Y	N	N	M	Tool interoperability
Kappel [49]	2011	Y	Y	Y	Y	Y	Y	N	N	Conceptual modeling
Klar [50]	2008	Y	N	Y	Y	Y	N	N	N	Requirement engineering
Lukácsy [54]	2007	N	Y	N	Y	N	N	N	M	Services
Mahé [55]	2010	Y	Y	N	Y	N	N	N	M	Visualization tools
Milanovic [58]	2009	Y	Y	Y	Y	Y	Y	N	O	Conceptual modeling
Moreno [59]	2008	N	Y	Y	Y	N	N	N	M	Web development tools
Opdahl [66]	2010	Y	Y	N	Y	N	N	N	O	WebML and UML
Polgar [69]	2009	N	Y	Y	Y	Y	Y	N	O	Model-driven development
Radjenovic [70]	2010	N	Y	N	Y	Y	Y	N	N	Conceptual modeling
Roser [71]	2007	Y	Y	N	Y	N	N	N	O	Conceptual modeling
Seifert [75]	2010	Y	N	N	N	Y	N	Y	N	Tool interoperability
Sunindyo [78]	2010	N	Y	N	Y	Y	N	N	O	Signal engineering
Tran [79]	2008	N	Y	Y	Y	Y	N	Y	N	Services
Vallecillo [80]	2010	Y	Y	N	N	N	N	Y	M	Conceptual modeling
Ziemann [86]	2007	N	Y	N	Y	Y	N	N	M	Enterprise modeling

of weavings among metamodels and ontologies (also called lifting [51]) implies additional complexity.

In relation to *Interoperability Verification* (*IV*) mechanisms, only six (6) approaches (18.2 % of the approaches analyzed) consider some kind of verification mechanisms. All these works are focused on verifying interoperability at *specification level*, which corresponds to appropriate specification of interoperability artifacts (weavings, pivots, and model transformations). However, none of the analyzed approaches consider the verification of the interoperability execution,

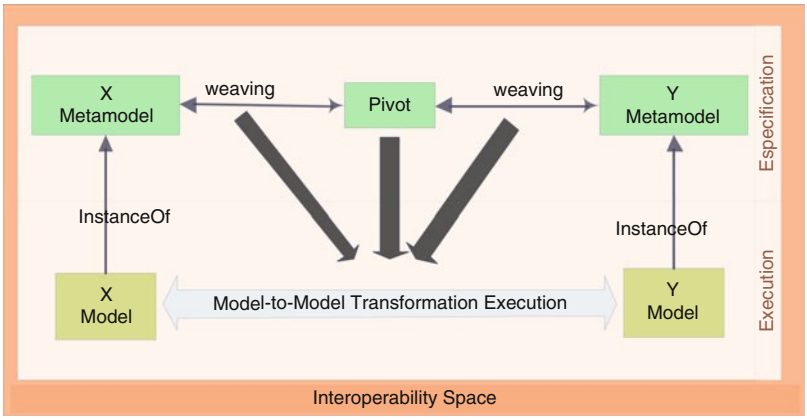

Fig. 1 General model-driven interoperability schema

i.e., the proper instantiation of the involved metamodels, and execution of model transformations.

Figure 1 provides a general model-driven interoperability schema obtained as results of this analysis.

Next section shows an interoperability model and the challenges that we have faced to achieve the automatic model-driven interoperability. This interoperability model is based on the model-driven schema obtained from the literature review results.

3 An MDD Interoperability Model

Model-based proposals related to the context of information systems and tool interoperability state different levels [39] to achieve an appropriate interoperability framework, such as [1, 67, 72, 83]. These proposals have several common aspects and present similar interoperability levels. In particular, we have centered our attention on the LISI (Levels of Information Systems Interoperability) [1] and LCIM (Levels of Conceptual Interoperability Models) [83] since the conceptual basis behind these two approaches covers the different features provided by previously analyzed studies. Moreover, these approaches can be easily generalized to different domains, have achieved a suitable maturity level, and their applicability has been empirically demonstrated.

The LISI approach was created by the U.S.A. defense department as a solution for defining, evaluating, measuring, and assessing information systems interoperability [1]. In this context, the correct and secure flow of information among the different systems is a critical concern. The LISI approach proposes an interoperability model comprised of five levels (from 0 to 4). Level 0 (isolated interoperability) corresponds to a manual interoperability, where the interoperation tasks must be

Table 2 The LISI reference model

Interoperability	Computing environment	Level	P	A	I	D
Enterprise	Universal	4	Enterprise level	Interactive	Multi-dimensional topologies	Enterprise model
Domain	Integrated	3	Domain level	Groupware	World-wide network	Domain model
Functional	Distributed	2	Program level	Desktop automation	Local networks	Program model
Connected	Peer-to-peer	1	Local/site level	Standard system drivers	Simple connection	Local
Isolated	Manual	0	Access control	N/A	Independent	Private

performed manually by the system users. Level 4 (enterprise interoperability) indicates that data and services are automatically interchanged by different applications in a transparent way for the system users.

The levels defined in the LISI reference model are transversally divided into four interoperability attributes called PAID, which correspond to Procedures, Applications, Infrastructure, and Data. Table 2 summarizes the LISI reference model.

The LCIM approach is related to modeling and simulation, and, hence, it is closer to the model-driven development domain. Modeling aspects related to LCIM have a direct correspondence to the modeling tasks involved in MDD processes. Simulation corresponds to the execution of the modeled systems; therefore, it can be considered equivalent to the model compilation tasks that are involved in MDD processes.

The LCIM proposal states seven interoperability levels (from 0 to 6). Level 0 corresponds to the non-interoperability level (the same as the LISI proposal). Level 6 corresponds to the conceptual interoperability. In this level, interoperability is achieved by means of the definition of mappings among the conceptual models that describe the involved systems. In other words, conceptual interoperability is achieved through the meta-specification of the software systems. Figure 2 shows the levels defined for the LCIM model.

If we project the LCIM ideas to the model-driven context, we can state that to achieve conceptual model-driven interoperability, it is necessary to define mappings among the involved modeling languages. To perform these mappings it is necessary to consider syntax, semantics, and technical aspects that are related to modeling languages definition, which corresponds to the levels 1, 2, and 3 of the LCIM approach.

The levels 4 and 5 (Pragmatic and Dynamic interoperability) of the LCIM approach are related to operation of information systems and management of systems' data in time. In a specific model-driven context, such as model-driven

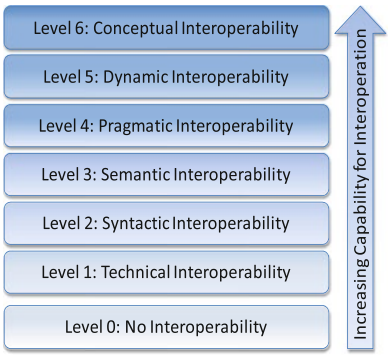

Fig. 2 LCIM model

development, these levels would be related to the evolution of the MDD models defined and their changes according to new system requirements. In the interoperability model that we propose, these interoperability levels are not considered since they are related to model synchronization and model evolution, which are topics that are not part of the model-driven interoperability vision. However, these are two interesting aspects that can be considered for future research in order to provide supporting facilities to model-driven interoperability.

In order to automate the model-driven interoperability, we have adopted the properties proposed by the LISI approach, which are the following:

- An appropriate interoperability *Procedure*, which indicates the elements that must be defined, and the steps that must be performed to interchange the modeling information.
- The *Applications* that manage the modeling information, which provide the features to automate interchange of models.
- The interoperability *Infrastructure*, which is related to the communication mechanisms among applications to assure the correct interchange of information and to prevent the loss of modeling information when the interchange process is performed.
- The *Data* (modeling information) must be specified in a standard format, which can be interpreted by different modeling tools with independency of implementation platforms and development contexts.

In summary, the interoperability model defined states model-driven interoperability in terms of technical, semantic, and syntactic interoperability. Also, model-driven interoperability can be automated by providing a concrete solution for procedure, application, infrastructure, and sata properties.

In relation to syntactic interoperability, different approaches have defined a particular syntax (abstract and concrete) to represent their modeling elements (conceptual constructs). The best example of this is UML [64]. This syntax is focused on supporting the semantics [38] of the modeling languages involved.

For the specification of the abstract syntax, it is possible to find standardized approaches, such as the MOF [62]. The MOF approach provides suitable support

for the generation of model-oriented technologies, such as model editors, and model transformation tools. This abstract syntax specification is performed by means of a metamodel definition, which represents the conceptual constructs (with their properties), the relationships that exist among the constructs, and a set of rules to manage the constructs' interaction.

From the metamodels that formalize the abstract syntax of modeling languages, the concrete syntax can be specified by using tools such as the eclipse graphical modeling framework (GMF) [23]. However, a standard for defining the concrete syntax related to a modeling language has not yet been defined.

The semantics related to modeling approaches is usually specified by means of textual representations, for instance, the UML specification [64] and the *i** framework [2]. In an MDD approach, we consider that the semantics is implicit in the mappings defined between the conceptual constructs and the corresponding software representations, which are used to perform the model-compilation process. However, there is a lack of an appropriate standard for the definition of the semantics related to modeling languages.

Thus, since only the abstract syntax of modeling languages is supported by standards that can be computationally interpreted, we propose the metamodels that formalize this abstract syntax as the starting point to support model-driven interoperability processes. From these metamodels, specific mappings can be defined (among the conceptual constructs of the involved modeling languages) to obtain semantic interoperability.

Technical interoperability can be achieved by using the interchange format defined for the open-source implementations of the metamodeling tools. For instance, the interchange mechanism implemented for the Eclipse UML2 tools is based on the XML specification [82]. This interchange mechanism is the XML Metadata Interchange (XMI) [61], which has been defined for the UML specification.

Thus, for automating model-driven interoperability it is necessary: (1) to establish an appropriate *Procedure* to generate the interoperability artifacts; (2) to indicate or implement the *Applications* that are necessary to manipulate the models and perform the model interchange; (3) to state the *Infrastructure* that will be used to communicate the *Applications*; and (4) to define the format used for the modeling *Data* representation.

We have chosen open-source applications to support automatic interoperability. In particular, we have considered the modeling tools developed in the context of the eclipse modeling project [22], such as the eclipse modeling framework (EMF) [21] and the Eclipse UML2 Project [24]. For the infrastructure, we have considered the XML implementation for the EMF tools, and the XMI specification related to the Eclipse UML2 models. The EMOF (EMF) and UML (Eclipse UML2) specifications provide the formats that are used to represent the modeling data. Thus, the applications, infrastructure, and data format are supported by current technologies, tools, and standards.

However, there is no standard procedure that can be used to perform automatic model-driven interoperability. Therefore we have defined a particular process that

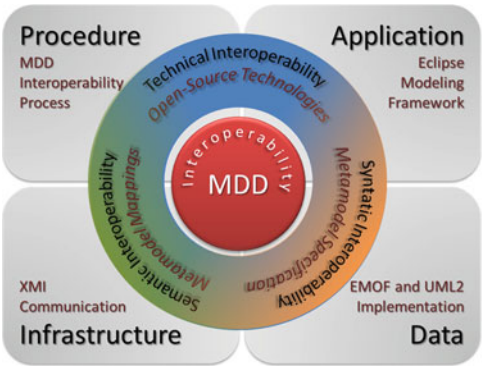

Fig. 3 MDD interoperability model instantiated

instantiates the proposed interoperability model to obtain an automatic model-driven interoperability solution. Figure 3 shows the interoperability model instantiated according to the analyzed standards, tools, and related works.

The process proposed for instantiating the interoperability model is mainly focused on the automatic interchange of modeling information in the context of an MDD process. In particular, it considers the integration of the modeling needs related to a specific MDD approach into already existing modeling approaches. For an appropriate representation of the specific MDD constructs, the constructs of the target modeling languages are customized to fix differences or to adding new properties in the context of the MDD approach. This is to define modeling language extensions. For the implementation of the required modeling language extensions, we use the UML profile extension mechanism since it is a standardized extension mechanism that has been improved according to the UML experience, and it is based on the MOF metamodeling standard. Therefore, the fundamentals related to UML profile extensions can be generalized to any modeling language that uses an MOF metamodel to formalize its abstract syntax. In addition, UML profile is a lightweight extension mechanism that does not alter the target metamodel, and, hence, the defined extensions do not affect the compatibility with the technologies that are based on the original specification of the modeling language customized.

By analyzing the previous background and related work, three main challenges must be faced to properly support model-driven interoperability. These challenges and the solutions proposed to solve them are detailed in the next section.

4 Challenges for Achieving the Model-Driven Interoperability

The first of the three challenges that must be solved for modeling language integration is to indicate the modeling artifact that will be used as starting point for this integration. The second challenge is related to define an appropriate mechanism to indicate the semantic equivalences between the involved modeling languages,

and the resolution of those interoperability issues that may prevent the correct interchange of modeling information. Finally, the third challenge is to automate the generation of the required metamodel extensions in order to reduce the potential errors and complexity that a manual modeling language customization involves.

4.1 First Challenge: Establish the Starting Point

For solving this first challenge, we have considered the definition (or selection) of the metamodels that are related to the involved modeling languages as starting point. These metamodels are the artifacts where equivalences among the modeling languages can be identified and the required extensions can be defined.

Metamodels provide good support to formalize the abstract syntax of modeling languages, which is essential to perform an appropriate integration of modeling languages. Also, in the current MDD context, metamodels are widely used for development of technologies and modeling languages. Thus, it has sense to consider an element that is commonly used by MDD-oriented approaches as starting point of an MDD interoperability process.

The paper presented by Selic in [76] indicates a set of elements that must be considered for an appropriate metamodel specification. These elements are the following:

- The set of conceptual constructs related to the modeling language, which are defined as classes (metaclasses) of the metamodel.
- The set of relationships that exist among the different conceptual constructs.
- The set constraints that manage the interaction among the different conceptual constructs, which are necessary to define valid models (instances of the metamodel).
- The notation related to each conceptual construct when corresponds.
- The meaning of the conceptual constructs defined.

For the specification of the involved metamodels we propose the use of the MOF metamodeling standard [62]. The use of MOF facilitates the definition of UML profiles for the implementation of modeling language extensions. Also, MOF is a suitable alternative for the specification of the required metamodels due to the following reasons:

- MOF is supported by a standardized interchange format (XMI [61]) .
- There exist different open-source metamodeling tools based on the MOF specification such as the Eclipse projects EMF [21] and UML2 [24].
- MOF is used by current model-to-model transformation technologies such as ATL [46] or QVT [63]
- There are many metamodel specifications based on MOF that can be used as reference modeling approaches.

- The use of MOF as common metamodeling language prevents the notation inconsistencies (at metamodel level) and facilitates the identification of equivalences between the different constructs.

However, there is an important lack in the MOF specification, which is the impossibility of indicating the notation (concrete syntax) and meaning (semantics) of the defined constructs [38]. The MOF metamodels only specify the abstract syntax of the corresponding modeling languages. Therefore, the notation and meaning of the constructs must be documented in a separated way. This information is relevant for the correct metamodel specification and it is helpful to understand the defined metamodels. Also, the notation and semantics are relevant for the appropriate implementation of MDD tools, such as modeling tools and model compilers.

The MOF specification provides two alternatives for metamodel definition, i.e., two metamodeling languages. The first of these languages is the complete set of constructs of the MOF specification, which is called CMOF (Complete-MOF). The second alternative corresponds to a subset of the MOF constructs, which provide essential metamodeling facilities. This second metamodeling language is called EMOF (Essential-MOF).

The metamodeling capabilities that are provided by EMOF are closer to the extension capabilities provided by UML profiles. By contrast, CMOF provides a set of metamodeling facilities that cannot be represented by means of UML profile extensions, for instance, n-ary associations, or property redefinition. Therefore, we consider the use of EMOF to specify the metamodels of the involved modeling languages.

Once the corresponding EMOF metamodels are specified, or selected in the case of already existing EMOF metamodels, the equivalences between metamodels must be indicated. These equivalences are used to identify the necessary metamodel extensions. At this point, the second challenge that must be faced arises.

4.2 Second Challenge: Identify Semantic Equivalences and Solve Interoperability Issues

This challenge involves the appropriate identification of semantic equivalences between the constructs related to a source and a target modeling language. In the context of our interoperability model instantiation, the source modeling language corresponds to the DSML that represent the constructs of the MDD approach involved, and the target modeling language is the preexisting modeling language that will be customized with the specific MDD syntax. This identification of semantic equivalences can be performed by means of model mappings (weavings, or semantic links) between the constructs of the source metamodel and the target metamodel. Thus, these mappings guide the identification of the necessary extensions to integrate into the target metamodel the abstract syntax of the source metamodel. However, certain structural differences (heterogeneities) between the

involved metamodels may prevent the appropriate mapping specification, and, hence, they prevent the correct identification of the required metamodel extensions. This situation is presented in works such as [76] and [85]. These works propose systematic approaches for the generation of UML profiles starting from metamodel mappings. However, due to structural differences that are present in the involved metamodels, the final UML profile generation cannot be completely automated. These structural differences also affect the completeness of the obtained UML profile, which cannot customize the target modeling language with all the modeling information required.

Therefore, to solve this challenge, we propose the definition of a pivot metamodel that allows the structural differences to be fixed, and an appropriate mapping specification to be obtained. This pivot metamodel that we have called *Integration Metamodel* [28] provides the necessary information to perform the appropriate integration of modeling languages.

4.3 Third Challenge: Automatic Generation of Metamodel Extensions

Finally, the third challenge is related to how to automate the generation of the required metamodel extensions from the defined metamodels and the metamodel mappings. This is to automate the generation of the required UML profile. The automatic generation of the required metamodel extensions prevents the potential inconsistencies between the syntax of the source and target metamodel that a manual specification may produce. In addition, the effort in the implementation of the UML profiles is considerably reduced due to the automatic generation since it is not necessary to know specific details related to the correct UML profile specification or deal with complexity of large metamodels. The benefits obtained from the automatic UML profile generation are very relevant since, according to Selic in [76], the lack of knowledge about the features of the UML profile specification has produced that many of the existing UML profiles definitions be invalid or of poor quality.

In general terms, the metamodel extensions that must be implemented in the UML profile can be automatically identified by comparing the source and target metamodels according to the semantic equivalences identified (defined in the metamodel mappings). Thus, the extensions are the additional modeling information that is necessary to fix the differences that exist between the target and source metamodel. For instance, if in the source metamodel there is a property that cannot be mapped to the target metamodel, then the UML profile extends the target metamodel with this non-mapped property.

Thus, a UML profile can be automatically generated by considering all the possible metamodel differences, and, for each one of these, to define specific rules that generate the necessary UML profile extensions.

For the application of the proposed interoperability model a specific process has been defined. The stages and artifacts involved in this interoperability process

are defined by considering the solutions proposed to solve the three challenges presented. This interoperability process is detailed in the next section.

5 The Model-Driven Interoperability Process

In this section, we introduce the process to achieve and automate interoperability in model-driven developments, which is comprised by the following four steps: (1) Definition of modeling anguage metamodels; (2) Definition of integration metamodel; (3) Automatic UML profile generation; and (4) Generation of model-interchange mechanisms. Steps 2–4 of the process are based on original contributions that were created to tackle the interoperability challenges identified.

The modeling language integration is the core of the model-driven interoperability process proposed. It automates the generation of the necessary metamodel extensions and guides the specification of appropriate mappings, which are the main artifacts to perform the automatic model interchange. Thus, the first three steps of the interoperability process are related to perform the integration of the modeling languages involved. Figure 4 shows a BPMN [84] schema of the interoperability process proposed.

In the definition of this interoperability process, different works have been considered. Some of these works are: definition of UML profiles using DSML metamodels [27, 52, 76, 85], correct use of metamodels in software engineering [41], UML profile implementations,[1] interchange between UML profiles and DSMLs [3], and new UML profile features that are introduced in UML [65]. The four steps that comprise the interoperability process are detailed below:

Step 1: Definition of Modeling Language Metamodels. The first step of the process corresponds to the starting point proposed as solution of the first integration challenge presented in the previous section, which is the specification or selection of the EMOF metamodels of the involved modeling languages. As guidance to perform this step, the paper presented in [32] shows an interesting case study related to the UML association.

Step 2: Definition of Integration Metamodel. The second step is the definition of an Integration Metamodel to identify the equivalences between the metamodels involved and to fix the mapping issues that are produced by structural differences that may exist. Detailed example of how defining an Integration Metamodel and its application is presented in [28, 29].

Step 3: Automatic UML Profile Generation. This step considers the automatic generation of the UML profile that implements the metamodel extensions that are required to customize the abstract syntax of a target modeling language with the

[1] OMG: Catalog of UML Profile Specifications, http://www.omg.org/technology/documents/profile_catalog.htm

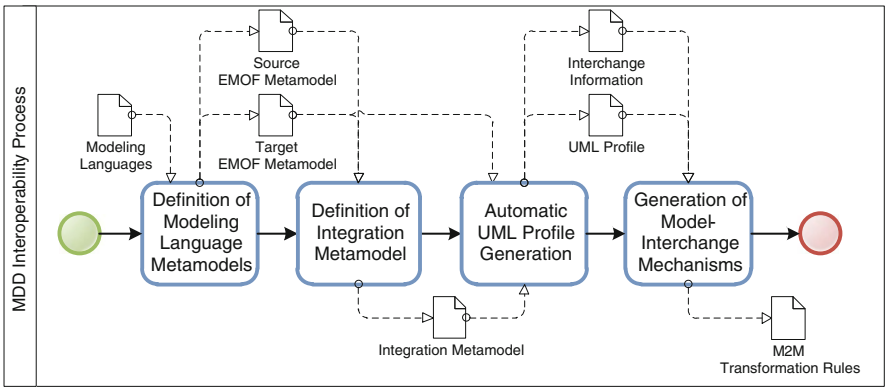

Fig. 4 Model-driven interoperability process

modeling information of the MDD approach involved. A suite of transformation rules and application example for this step has been presented in [30, 31]

Step 4: Generation of Model-Interchange Mechanisms. This step considers the generation of the necessary model transformations rules to automatically obtain from the models that are defined with the customized modeling language appropriate inputs (models) for specific modeling management tools, such as model compilers. The interchange mechanisms also transform source MDD models into equivalent representation using the customized modeling language. This is a bidirectional interchange of models. More details related to the development of this step can be found in [31].

For implementing an interoperability solution according to these four steps, existing open-source modeling tools can be used. For instance, the works presented in [32] and [34] implement a complete interoperability processes by considering the eclipse modeling tool facilities (UML profiles, eclipse metamodeling facilities (EMF), and ATL transformations). However, specific implementations can be also performed, for instance, in [29] we present a commercial tool that implements the model interchange mechanisms based on XSLT transformations.

Furthermore, the different artifacts that are involved in the application of the proposed interoperability process are defined to facilitate the validation and verification in each step. Some of the validation and verification facilities that can be obtained are the following:

- It is possible to verify the abstract syntax related to the modeling languages that must interoperate by means of the metamodels that are involved in the interoperability process. Also, the definition of metamodels by means of a standard metamodeling language (EMOF) facilitates the verification of the abstract syntax specified in relation to the supported semantics of the corresponding modeling languages.

- The construction of an Integration Metamodel facilitates the definition of specific rules for automatic UML profile generation. It allows the definition of verification mechanism that assures the correct application of these rules, and, hence, the correct generation of the resultant UML profile.
- The interchange of models is based on specific model-to-model transformation rules, which are based on the generated metamodel extensions and mappings. This allows the implementation of validation mechanisms to assure that the metamodel extensions and the defined models are defined according to the specification of the MDD approach involved.

6 Conclusions

In this chapter, we propose a conceptual model, which is used as reference to identify the necessary tools, and artifacts for the definition of concrete model-driven interoperability solutions.

The different elements considered in the proposed interoperability model have been instantiated by using the current model-based technologies. To complete the proposed interoperability model, we have defined a specific process, which indicates how the different elements of the proposed model can be coordinated to support the automatic interoperability in a model-driven context.

Thus, the interoperability process obtained is aligned with current modeling standards and technologies, such as modeling language specifications using metamodels, metamodel extension mechanisms that are implemented as UML profiles, and interchange mechanisms that are implemented through model transformations. Also, time and defects related to manual specification of metamodel extensions and transformation rules are reduced by means of the automatic generation of the interoperability artifacts involved.

The structure proposed for the interoperability process is also a suitable reference for other metamodel extension mechanisms or proposals for model interchange. This structure is easily adaptable to different model-based technologies. For instance, the UML profile generation rules can be changed to implement the required extensions with a different extension mechanism. The work presented by Bruck et al. in [15] introduces different approaches for the definition of metamodel extensions and provides a comparative summary about the approaches that are presented.

The adaptation to potential changes that the involved modeling languages may suffer is also improved by the proposed interoperability process. Changes in the modeling languages directly impact the defined metamodels. With the application of the proposed process, these changes are automatically propagated to the interoperability artifacts (metamodels extensions, and mappings). This is very important, especially when the involved modeling languages are comprised by a big number of conceptual constructs, which are permanently changing. In this context, the manual identification of the impact that a change in the modeling languages has over the

defined extensions and model transformation rules can be a titanic labor, which demands a lot of time and usually is error prone.

The report presented in [34] and the paper presented in [32] show two empirical evaluations of the proposal, which consider their industrial application in an industrial MDD scenario, and the definition of verification mechanisms to assure the completeness of the information exchanged.

Finally, the following works present the application of the interoperability model and process proposed in different contexts:

- Linking of goal-oriented modeling and MDD [5–7, 33, 68].
- Integration of domain-specific modeling languages and general-purpose modeling languages [29–32].
- Application of verification model in MDD contexts [56, 57].
- Business Modeling for Service Engineering [73, 74].
- Application of Software Maturity models in MDD processes [81].

References

1. A.W.P.: Levels of Information Systems Interoperability (LISI). Department of Defense, USA (1998)
2. Abdulhadi, S.: i* Guide Version 3.0 (August 2007). Available at http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Guide. Last access Apr 2013
3. Abouzahra, A., Bézivin, J., Fabro, M.D.D., Jouault, F.: A practical approach to bridging domain specific languages with UML profiles. Paper Presented at the Best Practices for Model Driven Software Development (OOPSLA'05), San Diego. ACM (2005)
4. Agostinho, C., Correia, F., Jardim-Goncalves, R.: Interoperability of complex business networks by language independent information models. Paper Presented at the 17th ISPE International Conference on Concurrent Engineering (CE 2010), Cracow. Springer (2011)
5. Alencar, F., Marín, B., Giachetti, G., Pastor, O., Castro, J., Pimentel, J.H.: From i* requirements models to conceptual models of a model driven development process. Paper Presented at the 2nd Working Conference on the Practice of Enterprise Modeling (PoEM 2009), Stockholm. Springer (2009)
6. Alencar, F., Marín, B., Giachetti, G., Pastor, O., Castro, J., Franch, X., Pimentel, J.: From i* to OO-method: problems and solutions. Paper Presented at the Fourth International i* Workshop (istar 2010), CAiSE Workshops, Hammamet. CEUR-WS (2010)
7. Alencar, F.M.R., Pastor, O., Marín, B., Giachetti, G., Castro, J.: Aligning goal-oriented requirements engineering and model-driven development. Paper Presented at the 11th International Conference on Enterprise Information Systems (ICEIS). Milan. Springer (2009)
8. Barnickel, N., Fluegge, M.: Towards a conceptual framework for semantic interoperability in service oriented architectures. Paper Presented at the 1st International Conference on Intelligent Semantic Web-Services and Applications, Amman. ACM (2010)
9. Baumgart, A.: A common meta-model for the interoperation of tools with heterogeneous data models. Paper Presented at the 3rd European Workshop on Model Driven Tool and Process Integration (MDTPI), Paris. Fraunhofer Institute for Open Communication Systems (2010)
10. Berger, S., Grossmann, G., Stumptner, M., Schrefl, M.: Metamodel-based information integration at industrial scale. Paper Presented at the 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2010), Oslo. ACM/IEEE (2010)
11. Berre, A.-J., Liu, F., Xu, J., Elvesaeter, B.: Model driven service interoperability through use of semantic annotations. Paper Presented at the International Conference on Interoperability for Enterprise Software and Applications China (IESA '09), Beijing. Springer (2009)

12. Biehl, M., Sjöstedt, C.-J., Törngren, M.: A modular tool integration approach—experiences from two case studies. Paper Presented at the 3rd European Workshop on Model Driven Tool and Process Integration (MDTPI), Paris. Fraunhofer Institute for Open Communication Systems (2010)
13. Blanc, X., Gervais, M.-P., Sriplakich, P.: Model bus: towards the interoperability of modelling tools. Paper Presented at the Model-Driven Architecture: Foundations and Applications (MDAFA), Linköping. Springer (2004)
14. Brambilla, M., Fraternali, P., Tisi, A.M.: A transformation framework to bridge domain specific languages to MDA. Paper Presented at the Models in Software Engineering Workshops and Symposia at MODELS 2008, Toulouse. Springer (2008)
15. Bruck, J., Hussey, K.: Customizing UML: Which Technique is Right for You? IBM, (2007). Available at http://www.eclipse.org/modeling/mdt/uml2/docs/articles/Customizing_UML2_Which_Technique_is_Right_For_You/article.html. Last access April 2013
16. Bruneliere, H., Cabot, J., Clasen, C., Jouault, F., Bezivin, J.: Towards model driven tool interoperability: bridging eclipse and microsoft modeling tools. Paper Presented at the 6th European Conference on Modelling Foundations and Applications (ECMFA 2010), Paris. Springer (2010)
17. Costa, R., Garcia, O., Nuñez, M., Maló, P., Gonçalves, R.: Integrated solution to support enterprise interoperability at the business process level on e-procurement. Paper Presented at the 3rd International Conference on Interoperability of Enterprise Software and Applications (I-ESA 2007), Funchal - Madeira Island. I-ESA (2007)
18. Coutinho, L., Brandao, A., Sichman, J., Boissier, O.: Model-driven integration of organizational models. Paper Presented at the IX Agent-Oriented Software Engineering (AOSE). Lecture Notes in Computer Science, vol. 5386, pp. 1–15 (2009)
19. Crnkovic, I., Malavolta, I., Muccini, H.: A model-driven engineering framework for component models interoperability. Paper Presented at the International Symposium on Component Based Software Engineering (CBSE), East Stroudsburg. Springer (2009)
20. Diskin, Z., Xiong, Y., Czarnecki, K.: Specifying overlaps of heterogeneous models for global consistency checking. Paper Presented at the First International Workshop on Model-Driven Interoperability (MDI), Oslo (2010)
21. Eclipse: Eclipse Modeling Framework Project. Available at http://www.eclipse.org/modeling/emf/. Last access Apr 2013
22. Eclipse: Eclipse Modeling Project. Available at http://www.eclipse.org/modeling/. Last access Apr 2013
23. Eclipse: Graphical Modeling Framework Project. Available at http://www.eclipse.org/modeling/gmp/. Last access Apr 2013
24. Eclipse: UML2 Project. Available at http://www.eclipse.org/uml2/. Last access Apr 2013
25. Fabro, M.D.D., Valduriez, P.: Towards the efficient development of model transformations using model weaving and matching transformations. Softw. Syst. Model. **8**(3), 305–324 (2009)
26. Falbo, R.A., Guizzardi, G., Duarte, K.C.: An ontological approach to domain engineering. Paper Presented at the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02), Ischia Island. ACM (2002)
27. Fuentes-Fernández, L., Vallecillo, A.: An introduction to UML profiles. Eur. J. Inform. Professional (UPGRADE) **5**(2), 5–13 (2004)
28. Giachetti, G., Valverde, F., Pastor, O.: Improving automatic UML2 profile generation for MDA industrial development. Paper Presented at the 4th International Workshop on Foundations and Practices of UML (FP-UML), ER Workshop, Barcelona. Springer (2008)
29. Giachetti, G., Marin, B., Pastor, O.: Integration of domain-specific modeling languages and UML through UML profile extension mechanism. Int. J. Comput. Sci. Appl. **6**(5), 145–174 (2009)
30. Giachetti, G., Marín, B., Pastor, O.: Using UML as a domain-specific modeling language: a proposal for automatic generation of UML profiles. Paper Presented at the 21st International Conference on Advanced Information Systems (CAiSE 2009), Amsterdam. Springer (2009)

31. Giachetti, G., Marín, B., Pastor, O.: Using UML profiles to interchange DSML and UML models. Paper Presented at the Third International Conference on Research Challenges in Information Science (RCIS), Fès. IEEE (2009)
32. Giachetti, G., Albert, M., Marín, B., Pastor, O.: Linking UML and MDD trough UML profiles: a practical approach based on the UML association. J. Universal Comput. Sci. (J.UCS) **16**(17) (2010)
33. Giachetti, G., Alencar, F., Marín, B., Pastor, O., Castro, J.: Beyond requirements: an approach to integrate i* and model-driven development. Paper Presented at the XIII Conferencia Iberoamericana en Software Engineering (CIbSE 2010), Cuenca. CIBSE (2010)
34. Giachetti, G., Alencar, F., Franch, X., Marín, B., Pastor, O.: Technical Report ProS-TR-2011-07: Automatic Verification of Requirement Models for Their Interoperability in Model-Driven Development Processes. Universidad Politécnica de Valencia, Spain (2011)
35. Giachetti, G., Marín, B., Valverde, F.: Interoperability for model-driven development—current state and future challenges. Paper Presented at the 6th International Conference on Research Challenges in Information Science (RCIS) (2012)
36. Guerra, E., Lara, J.D., Orejas, F.: Inter-modelling with patterns. Software. Syst. Model. (SoSym), **12**(1), 145–174 (2013)
37. Guha, R., Al-Dabass, D.: Impact of web 2.0 and cloud computing platform on software engineering. Paper Presented at the Electronic System Design (ISED), Bhubaneswar. IEEE (2010)
38. Harel, D., Rumpe, B.: Meaningful modeling: what's the semantics of "semantics"? IEEE. Comput. **37**(10), 64–72 (2004)
39. Haslhofer, B., Klas, W.: A survey of techniques for achieving metadata interoperability. ACM Comput. Surveys (CSUR) **42**(2) (2010)
40. Hein, C., Ritter, T., Wagner, M.: Model-driven tool integration with modelbus. Paper Presented at the 1st International Workshop on Future Trends of Model-Driven Development (FTMDD), Milan. SciTePress (2009)
41. Henderson-Sellers, B.: On the challenges of correctly using metamodels in software engineering. Paper Presented at the 6th Conference on Software Methodologies, Tools, and Techniques (SoMeT), Rome. IOS International (2007)
42. Höfferer, P.: Achieving business process model interoperability using metamodels and ontologies. Paper Presented at the 15th European Conference on Information Systems (ECIS 2007), St. Gallen. AIS (2007)
43. ICT: The future of cloud computing, opportunities for European cloud computing beyond 2010. In: Keith J., Burkhard N.-L. (eds.) EC ICT Research in FP7—Expert Group Report, ICT, European Commission. Information Society and Media (2010)
44. IEEE: IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries. IEEE, New York (1990)
45. Jankovic, M., Ivezic, N., Knothe, T., Marjanovic, Z., Snack, P.: A case study in enterprise modelling for interoperable cross-enterprise data exchange. Paper Presented at the 3rd International Conference on Interoperability of Enterprise Software and Applications (I-ESA 2007), Funchal - Madeira Island. I-ESA (2007)
46. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: a model transformation tool. Sci. Comput. Program. **72**(1–2), 31–39 (2008)
47. Jouault, F., Guéguen, T.: Integration by model-driven virtual tool. Paper Presented at the 2nd European Workshop on Model Driven Tool and Process Integration (MDTPI) (2009)
48. Kappel, G., Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W., Schwinger, W., Wimmer, M.: Lifting metamodels to ontologies: a step to the semantic integration of modeling languages. Paper Presented at the MoDELS 2006, Genova. Springer (2006)
49. Kappel, G., Wimmer, M., Retschitzegger, W., Schwinger, W.: Leveraging model-based tool integration by conceptual modeling techniques. In: The Evolution of Conceptual Modeling. LNCS, vol. 6520, pp. 254–284. Springer, Berlin (2011)

50. Klar, F., Rose, S., Schürr, A.: A meta-model-driven tool integration development process. Paper Presented at the 2nd International United Information Systems Conference (UNISCON'2008), Klagenfurt. Springer (2008)
51. Kolovos, D., Paige, R., Rose, L., Polack, F.: The Epsilon Book. Eclipse Foundation (2010). Available at http://www.eclipse.org/epsilon/doc/book/. Last access April 2013
52. Lagarde, F., Espinoza, H., Terrier, F., Gérard, S.: Improving UML profile design practices by leveraging conceptual domain models. Paper Presented at the 22th IEEE/ACM International Conference on Automated Software Engineering (ASE), Atlanta. IEEE/ACM (2007)
53. Loniewski, G., Insfran, E., Abrahao, S.: A systematic review of the use of requirement engineering techniques in model-driven development. Paper Presented at the 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2010) (2010)
54. Lukácsy, G., Szeredi, P., Benkő, T.: Towards automatic semantic integration. Paper Presented at the 3rd International Conference on Interoperability of Enterprise Software and Applications (I-ESA), Funchal - Madeira Island. I-ESA (2007)
55. Mahé, V., Brunelière, H., Jouault, F., Bézivin, J., Talpin, J.-P.: Model-driven interoperability of dependencies visualizations. Paper Presented at the 3rd European Workshop on Model Driven Tool and Process Integration (MDTPI), Paris. Fraunhofer Institute for Open Communication Systems (2010)
56. Marín, B., Giachetti, G., Pastor, O., Vos, T.E.J.: A tool for automatic defect detection in models used in model-driven engineering. Paper Presented at the 7th International Conference on the Quality of Information and Communications Technology (QUATIC), Porto. IEEE (2010)
57. Marín, B., Vos, T., Giachetti, G., Baars, A., Tonella, P.: Towards testing future web applications. Paper Presented at the 5th IEEE International Conference on Research Challenges in Information Science (RCIS 2011), Gosier. IEEE (2011)
58. Milanovic, N., Cartsburg, M., Kutsche, R., Widiker, J., Kschonsak, F.: Model-based interoperability of heterogeneous information systems: an industrial case study. Presented at the 5th European Conference on Model Driven Architecture—Foundations and Applications (ECMDA-FA). Lecture Notes in Computer Science, vol. 5562, pp. 325–336 (2009)
59. Moreno, N., Vallecillo, A.: Towards interoperable web engineering methods. J. Am. Soc. Inf. Sci. Technol. **59**, 1073–1092 (2008)
60. Ohren, O.P., Chen, D., Grangel, R., Jaekel, F.-W., Karlsen, D., Knothe, T., Rolfsen, R.K.: ATHENA-A1, Deliverable DA1.5.2: Report on Methodology description and guidelines definition. Oslo (2005)
61. OMG: XMI 2.4.1 Specification (2011)
62. OMG: MOF 2.4.1 Core Specification (2011)
63. OMG: QVT 1.1 Specification (2011)
64. OMG: UML 2.4.1 Superstructure Specification (2011)
65. OMG: UML 2.4.1 Infrastructure Specification (2011)
66. Opdahl.: Incorporating UML class and activity constructs into UEML. Paper Presented at the International Conference on Advances in Conceptual Modeling: Applications and Challenges—ER 2010 Workshops, Vancouver. Springer (2010)
67. Ouksel, A.M., Sheth, A.: Semantic interoperability in global information systems. ACM SIGMOD **28**(1), 5–12 (1999)
68. Pastor, O., Giachetti, G.: Linking goal-oriented requirements and model-driven development. In: Nurcan, S., Salinesi, C., Souveyet, C., Ralyté, J. (eds.) Intentional Perpectives on Information Systems Engineering, pp. 257–276. Springer, Heidelberg (2010)
69. Polgár, B., Ráth, I., Szatmári, Z., Horváth, Á., Majzik, I.: Model-based integration, execution and certification of development tool-chains. Paper Presented at the Second European Workshop on Model Driven Tool and Process Integration (MDTPI), Enschede. Fraunhofer Verlag (2009)
70. Radjenovic, A., Paige, R.F.: Behavioural interoperability to support model-driven systems integration. Paper Presented at the 1st Workshop on Model Driven Interoperability (MDI 2010), Oslo. ACM (2010)

71. Roser, S., Bauer, B.: Improving interoperability in collaborative modelling. Paper Presented at the 3rd International Conference on Interoperability of Enterprise Software and Applications (I-ESA 2007), Funchal - Madeira Island. I-ESA (2007)
72. Sarantis, D., Charalabidis, Y., Psarras, J.: Towards standardising interoperability levels for information systems of public administrations. In: Yannis C., Hervé P., Euripidis L., Kai M. (eds.) eJETA Special Issue on "Interoperability for Enterprises and Administrations Worldwide". eJETA J. (2008)
73. Scheithauer, G., Kett, H., Kaiser, J., Hackner, S., Hu, H., Wirtz, G.: Business modeling for service engineering: a case study in the IT outsourcing domain. Paper Presented at the Proceedings of the 2010 ACM Symposium on Applied Computing, Sierre (2010)
74. Scheithauer, G., Wirtz, G.: Business modeling for service descriptions: a meta model and a UML profile. Paper Presented at the 7th Asia-Pacific Conference on Conceptual Modelling (APCCM 2010), Brisbane, Australia. Australian Computer Society (2010)
75. Seifert, M., Wende, C., Aßmann, U.: Anticipating unanticipated tool interoperability using role models. Paper Presented at the 1st Workshop on Model Driven Interoperability, Oslo. ACM (2010)
76. Selic, B.: A systematic approach to domain-specific language design using UML. Paper Presented at the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Santorini Island. IEEE (2007)
77. Staron, M., Wohlin, C.: An industrial case study on the choice between language customization mechanisms. Paper Presented at the Product-Focused Software Process Improvement (PROFES), Amsterdam. Springer (2006)
78. Sunindyo, W.D., Moser, T., Winkler, D., Biffl, S.: A process model discovery approach for enabling model interoperability in signal engineering. Paper Presented at the 1st Workshop on Model Driven Interoperability, Oslo. ACM (2010)
79. Tran, H., Zdun, U., Dustdar, S.: View-based reverse engineering approach for enhancing model interoperability and reusability in process-driven SOAs. Paper Presented at the 10th International Conference on Software Reuse (ICSR 2008), Beijing. Springer (2008)
80. Vallecillo, A.: On the combination of domain specific modeling languages. Paper Presented at the 6th European Conference on Modelling Foundations and Applications (ECMFA 2010), Paris. Springer (2010)
81. Vasconcelos, A.M.L., Giachetti, G., Marín, B., Pastor, O.: Towards a CMMI-compliant goal-oriented software process through model-driven development. Paper Presented at the Practice of Enterprise Modeling (POEM), Oslo. Springer (2011)
82. W3C: XML Web Page. http://www.w3.org/XML/. Accessed April 2013
83. Wang, W., Tolk, A., Wang, W.: The levels of conceptual interoperability model: applying systems engineering principles to M&S. Paper Presented at the 2009 Spring Simulation Multiconference (SpringSim'09), San Diego. ACM (2009)
84. White, S.A., Miers, D.: BPMN Modeling and Reference Guide. Future Strategies Inc., Lighthouse Pt (2008)
85. Wimmer, M., Schauerhuber, A., Strommer, M., Schwinger, W., Kappel, G.: A semi-automatic approach for bridging DSLs with UML. Paper Presented at the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM), Montréal. University of Jyväskylä, Jyväskylä (2007)
86. Ziemann, J., Ohren, O., Jäkel, F.-W., Kahl, T., Knothe, T.: Achieving enterprise model interoperability applying a common enterprise metamodel. Paper Presented at the 2nd International Conference on Interoperability of Enterprise Software and Applications (I-ESA 2006), Bordeaux (2007)