

Study of Interoperability between Meta-Modeling Tools

Heiko Kern

University of Leipzig

Augustusplatz 10

04109 Leipzig, Germany

Email: kern@informatik.uni-leipzig.de

Abstract—Modeling is a fundamental concept in software engineering and other system development disciplines. Nowadays the modeling process is supported by powerful modeling tools. Generally speaking, tools which support the definition and usage of self-defined languages are called meta-modeling tools. An important requirement for meta-modeling tools is the interoperability among each other. For instance, interoperability helps to build complex tool chains covering the whole development process. Furthermore, interoperability can also avoid the vendor lock-in effect. Thus, interoperability facilitates the replacement of a tool by a new tool better fitting the customer needs. The objective of this paper is to investigate the current status of interoperability between meta-modeling tools. In more detail, we study the degree of model exchange between meta-modeling tools and look for typical exchange approaches. The study focuses on meta-modeling tools and approaches which are being used in practice or the real world, respectively.

I. INTRODUCTION

MODELING is a fundamental concept in software engineering and other disciplines. A model represents a system in an abstract way. The abstraction helps to improve the understanding of a system and can facilitate the communication between different stakeholders. Beyond that, in modern development approaches (e.g. Model-Driven Software Development (MDS) [22] or Domain-Specific Modeling [11]) models are increasingly used for automating development tasks such as code generation, model transformation or model-based testing.

Beside a theoretical foundation of modeling, a suitable tool infrastructure is necessary to enable the practical usage of MDS approaches. Current modeling tools offer a variety of features which support the user during the modeling process. Modeling tools supporting the definition as well as the usage of self-defined languages are called meta-modeling tools. The modeling languages in these tools are generally defined by meta-models. Examples of such meta-modeling tools are MetaEdit+ [11], Generic Modeling Environment [15] or Microsoft Visio [4].

An important requirement for meta-modeling tools is the interoperability among each other. Interoperability is the ability of two or more tools to work together. For instance, often a tool is dedicated to a specific task. Tools have to work together or inter-operate to build complex tool chains covering the whole development process. Another issue is the evolution of a tool landscape. Interoperability can avoid the vendor lock-in effect.

Thus, interoperability facilitates the replacement of a tool by a new tool better fitting the customer needs.

The objective of this paper is to investigate the current status of interoperability between meta-modeling tools. Although there are a variety of approaches, the current state of practice in the area of modeling is unclear. In more detail, we want to study the degree of model exchange between meta-modeling tools and look for typical exchange approaches. The study focus on meta-modeling tools and approaches which are used in practice or the real world, respectively. We mainly consider the import and export features of meta-modeling tools in order to exchange models and meta-models. The objective can be founded with the following two research questions:

– *Question 1: What is the degree of interoperability?*

The first research question investigates the degree of interoperability. We want to analyze between how many of the involved tools an exchange of models is possible? Based on our experience, we assume that the model exchange between different meta-modeling tools is insufficient. This study will prove this assumption.

– *Question 2: What are the approaches to realize interoperability?* In order to give a satisfying answer to this question, research of approaches is necessary. There are already a variety of approaches in theory and literature. However, in this study we want to identify approaches used in practice.

The paper is structured as follows. In the subsequent section, we give a foundation of the interoperability concept. In section III we present a set of aspects which helps to scope the investigation. Afterward in section IV, we describe the methodology for the tool selection and analysis of these tools. In section V we present the results of the study and discuss the validity of these results. Finally, we conclude in section VI.

II. INTEROPERABILITY

Interoperability is in research and in practice a subject of discussion since there are software systems. The word consists of two parts: “inter-operate” and “ability”. Inter-operate means that two systems can work together [6] and the suffix ability expresses “the ability of a system [...] to work with or use the parts or equipment of another system” [1]. A basis for interoperability is the capability to exchange information between two or more systems and to use the information that has been exchanged [2]. Furthermore, interoperability is

the basis to integrate systems. Integration is also an widely-used term in software and system development and can be defined as the combination and coordination of separate things, elements or units into a whole, so that they work together effectively [1], [3]. Regarding the concept of interoperability, integrated systems must be interoperable in any form, but interoperable systems do not need to be integrated. Interoperability extends the borders of already existing systems and enables the connection to other systems. Here, interoperability is often associated with loosely-coupled systems, where systems keep their autonomy [17]. In contrast to this, integration is characterized by a closely-coupled systems, where system are interdependent and difficult to separate from each other.

Another term in this context is migration. Generally, migration denotes processes of spatial movement. In information technology there are different application areas for migration, such as software systems, databases, application systems or hardware. A migration in the area of software systems is, for instance, updating from one major software release to the next highest version of the same software vendor. Already existing data, settings or specific extensions have to be transferred to the new software system.

The focus of this article is the interoperability of different meta-modeling tools. In this context, interoperability deals with the exchange of models and meta-models between meta-modeling tools. The exchange is realized as a migration of models and meta-models from one tool to another. The migration from one to another tool should be an isomorphic relation in order to preserve the structure and semantics of models. The terms integration, interoperability and migration can be used as synonyms in this paper.

III. SCOPE OF STUDY

There are a variety of problems and solutions concerning the interoperability issue. This shows, for instance, the annotated bibliography from Wicks [26] which contains a huge amount of papers about the interoperability issue. In this section, we define a set of aspects or dimensions which helps to scope the research (questions) of this study. The finding and selection of these aspects are based on a theoretical study of different approaches and problems through literature analysis and the authors's knowledge. The dimensions and their properties are not eligible for completeness but fit the objective of this paper.

A. Unification Mechanism

One of the main reason for missing interoperability is heterogeneity between artifacts that have to be exchanged (e.g. models and meta-models). This heterogeneity between the models can be, for instance, of syntactic or semantic nature. To achieve interoperability between the participating models it is necessary to overcome this heterogeneity and to find a unification between different structures. We can distinguish between the following two fundamental unification mechanisms.

1) *Common Structure*: A mechanism to realize interoperability is to avoid heterogeneity a prior by defining a common structure. The definition can be regarded as a development process for a standard. Such a standard defines, for instance, a common structure of models and meta-models, their semantics and a specification for the exchange of models. If all systems conform to a selected standard, interoperability is guaranteed by this standard. Standards can address different aspects of the exchange of models and languages. In the domain of modeling, there are standards which define a whole language (syntax, semantic and pragmatics). One example is the Unified Modeling Language (UML) [7]. Additional to this, there are standards which define a whole meta-modeling environment (e.g. Meta Object Facility (MOF) [20] or Eclipse Modeling Framework (EMF) [23]) and a corresponding exchange format (e.g. XML Metadata Interchange (XMI) [21]). Meta-modeling and modeling tools which are implementing MOF, UML and XMI as serialization syntax can exchange models and meta-models without problems (theoretically).

2) *Transformation*: Another mechanism is the transformation of different models and meta-models. A transformation defines a mapping between different structures in order to overcome heterogeneity. Similar to standards, transformation can address semantic or syntactic issues. If there is no standard in order to exchange data between tools, transformations are a powerful approach to exchange models or meta-models. The mechanism of a common structure and transformations are not mutually exclusive. A proprietary meta-modeling environment can implement a standard by using transformations in order to create a model and meta-models conforming to this standard. But this is only possible up to a certain degree.

B. Modeling Level

A meta-modeling tool consists of a modeling and a language level. On the language level (also called as meta-modeling level) a language engineer can define different modeling languages by the creation of meta-models. These modeling languages can be used by a modeler at the modeling level to create models. Based on these two levels, we differentiate between the following two cases of model exchange.

1) *Model Level*: The exchange on the model level includes only the models themselves. The exchange of languages is excluded on this level.

2) *Language Level*: Additionally to the exchange on the model level, it is possible and necessary to exchange languages between meta-modeling tools. The exchange of artifacts on language level includes generally the exchange on model level. A sub-aspect on this level is the language preservation. Generally the source and target models and language should be isomorphic. Language preservation can relate to the meta-model and the concrete syntax of language.

C. Topology

Meta-modeling tools exchange their data by using different topologies. The concept of topology originally stems from the field of computer networks and is also used in software

and system integration (e.g. Enterprise Application Integration [16]). The topology concept can be transferred to the area of meta-modeling tool integration. We can distinguish the following topologies.

1) *Point-to-Point*: A simple topology is a point-to-point connection which connects two tools directly to each other. In this case a tool exports their models and meta-models via a file. The target tool imports this file and reads the models and meta-models. If an external transformation is part of this point-to-point connection, than we have an indirect point-to-point connection. Otherwise, we talk about a direct point-to-point connection.

2) *Complex Topologies*: If there are more than two meta-modeling tools involved in the integration, a point-to-point integration can be insufficient. For that reason, there are more complex integration topologies such as star or bus. A star topology is characterized by the fact that there is one common exchange format or interface to exchange models and meta-models between all participating tools. The realization of a star solution often requires an additional integration component, which is in the center of the integration architecture. This component controls the integration process and serves as a common structure for models and meta-models. Realizations of more complex structure are, for instance, Model Bus [9] or BPM-X-Converter¹.

D. Integration Layer

The integration of software requires access to artifacts that should be exchanged between the software systems. Generally there are different layers to access these artifacts. Integration layers describe on which layer the exchange of artifacts is realized. Based on the integration of software development tools [25], we can differentiate between the following typical integration layers.

1) *Data*: Models and meta-models can be represented as data in files or databases. Hence, the integration can be realized on the data layer. Many tools enable the export and import of models and/or meta-models as files. In this case, no complex infrastructure is necessary for building an integration solution. The disadvantage of this approach is that the serialization of models and meta-models as data are often complex. The processing of these complex data structures implies a complex solution (transformations and/or exchange formats).

2) *Function*: Above the data layer, many tools provide an API with different functions. The usage of functions are easier than operating directly on the data layer because complex operation are encapsulated. Typical functions are the selection and creation of model elements. There are integration approaches which uses the function layer instead of the data layer.

3) *Presentation*: The third layer is the presentation layer. The exchange on presentation layer often only considers the graphical representation of models. For instance, the export as image considers only the graphical representation. In doing

TABLE I
SCOPE OF THE STUDY (GRAY = FOCUS)

Unification Mechanism	Common Structure	Transformation	
Modeling Level	Model Level	Language Level	
Topology	Point-to-Point	Complex Topology	
Integration Layer	Data	Function	Presentation

so, the import is unsatisfying because the access of single model elements is impossible. A further example of exchange at presentation layer is the approach of Object Linking and Embedding (OLE).

E. Research Scope

Based on the dimensions described in this section, we setup the scope of the study as follows. Table I shows an overview of the selected investigation dimensions.

- *Unification Mechanism*: The study includes the investigation of common structures (standards) and transformation approaches.
- *Modeling Level*: We investigate approaches on model and language level but the focus lies on approaches on the languages level while preserving the language structure.
- *Topology*: We study interoperability approaches that realize a direct point-to-point connection between tools.
- *Integration Layer*: The study considers all layers but we focus on data layer.

IV. SELECTION AND ANALYSIS OF TOOLS

A. Tool Selection

The selection of tools is an important aspect of this study because the tools form the basis for later analysis. For the tool search we mainly use the World Wide Web. Finding meta-modeling tools is a difficult task because the term meta-modeling tool is a theoretical term that is often used in the context of the Model-Driven Engineering community. Tool vendors uses different names for their meta-modeling tools. Beside the term meta-modeling tool, we use different synonyms such *meta-case tool* [24] or *modeling tool*. Most meta-modeling tools are denoted as modeling tools with the ability to define their own language. Hence, we also include in our search the general word *modeling tool*. Based on the initial set of tools, we filter the tools by the following criteria.

- *Maturity level*: The tools must fulfill a certain maturity level. A tool must be installable and usable for later analysis. Many tool vendors provide a trial version of their tool. The most tools are available as desktop applications but there are also web-based tools.
- *Concrete syntax*: A further requirement concerns the concrete syntax of models. We only select meta-modeling tools that enable the definition of graphical modeling languages.
- *Modeling domain*: The third criterion concerns the modeling domain of tools. Generally, meta-modeling tools are tools which allow the definition of domain-specific

¹<http://www.bpm-x.com/>

languages in different domains. This implies that many tools have a universal/generic character. However, many modeling tools relate to a certain modeling discipline. In this study we focus on the following domains: software development, business process modeling, and data modeling.

- *Meta-modeling capability*: The last criterion is the meta-modeling capability. We can distinguish between the heavyweight and lightweight approach [8]. The heavyweight approach enables the creation of a language through the definition of a complete new meta-model (e.g. MetaEdit+). The lightweight approach adapts an already existing meta-model (e.g. UML profile mechanism). A tool must support the heavyweight meta-modeling approach by using a three-level model hierarchy consisting of model, meta-model and meta-model.

Table III in the appendix shows the modeling tools we found during our search. Each tool in this list fulfill the first three selection criteria. The third column in Table III shows the meta-modeling capability of each tool. We only include tools that support the heavyweight meta-modeling approach. The last column indicates that a tool is included in this study. Overall the study includes 20 tools which fulfill the defined criteria.

B. Tool Analysis

The analysis starts with the installation of each tool. Afterwards, we investigate the import and export functionality. Typically, there are different interface layers: user, function and data interfaces. We concentrate on the user interface and especially on the tool menus. Often tools provide a menu entry for import and export of modeling artifacts. Some tools have no extra import and export menu because they offer this functionality in the load and save menu.

In addition to the user interface analysis, available documentation is used to find out exchange possibilities. For instance, we use product information and manuals because a lot of tool vendors emphasize and describe their import and export capabilities.

Some tools provide import and export capabilities in their programming interface or provide a generator component that can be used to implement export and import scripts. These functions are excluded in the study. We only include export and import interfaces that already exist. The exchange possibilities have to be ready to use without any programming of generators or functions.

Furthermore, we restrict the analysis of the exchange possibilities of a tool. We test the exchange mechanism in order to understand the approach used, but we do not investigate the quality of the model exchange. Table IV in the appendix shows the result of this analysis. The table contains the import and export capabilities of each tool. Based on this raw data, we derive the results in the next section.

V. RESULTS

A. Unification Mechanism

1) *Common Structure*: Many tools use the approach of a common structure in order to exchange their meta-models and models. But the used formats are often a proprietary definition which realize the saving and loading of models instead of the model exchange between different tools. However, some tools use the Visio format in order to exchange models and language elements. In addition to the proprietary formats, there are standards which allow only the exchange of models. These models must conform to a certain (standard) language. For instance, some tools enable the export and import of BPMN-XML [19]. But these standards do not allow the exchange of languages or meta-models. Finally, there is no common format – with the exception of the Visio format – that is used to exchange meta-models and models between different tools.

2) *Transformation*: The exploration of transformation approaches are difficult because most tools implement their import and export process as a black box. Hence, we can only investigate transformations which are explicit or visible during the import or export. We found some tools which support transformations during the exchange. The first approach is used in Agilian, Visual Paradigm for UML and Business Process Visual Architect. The transformation is realized by a wizard which allows to configure a mapping between Visio models and models of this tool. The mapping can only be applied to certain modeling languages. A further tool which supports a mapping during the exchange process is ARIS. This tool allows a configurable import of Visio models. The description of the configuration is realized in XML on language level. Generally the transformations are not comparable to powerful transformation approaches such as Eclipse Epsilon Transformation Language (ETL) [14] or Atlas Transformation Language (ATL) [10].

B. Modeling Level

1) *Model Level*: Some meta-modeling tools contain pre-defined modeling languages. Based on these pre-defined languages, some tools offer a language-specific exchange. An example is Agilian that allows the export of BPMN-XML and Business Process Visual Architect that enables the import of BPMN-XML. These specific languages are often standards in a certain domain. Regarding the unification mechanism, this approach follows the strategy of a common structure to exchange models. The limitation to a certain language is unsatisfying in the context of meta-modeling tools.

Additional to this, there is an approach allowing the generic exchange of models between tools. The approach can exchange each model as a generic graph or tree. But the interpretation of models which conform to this generic format is unclear because of the missing language definition.

For instance, the following tools support exchange on the model level:

- Agilian, Visual Paradigm for UML, Business Process Visual Architect: These tools allow the import of Visio

stencils, but the imported masters of a stencil are not part of a target language. Masters are transformed into a separated icon library. Thus, these tools allow only the import of models conforming to a certain language. Please note, a Visio stencil can be regarded as a meta-model [13].

- ARIS: This tool allows the import of Visio models. It is not possible to import stencils. ARIS offers a mapping function which enables a mapping between Visio stencil elements and certain ARIS language elements. Based on this mapping, ARIS imports Visio models.
- Edraw Max: This tool enables the import of Visio models. It is impossible to import stencil elements.
- Lucidchart: This tool allows the import and export of Visio models. The export was not testable in the free version. The import transforms only graphical elements and no stencil elements.
- *Dia*: *Dia* allows the import and export of Visio models without stencils.

C. Language Level

In contrast to the model level, there are tools which allow the exchange of modeling languages and models conforming to these languages. In this case, the exchange approach transforms the source language into the target language. Based on this transformation, all models conforming to the source language are transformed into models, which are conformed to the target language. Some tools provide the reverse order of these transformations, that is, the tool imports model elements and after that the tool creates the corresponding language elements. But this reverse order leads to the problem that the import only considers a certain set of language elements.

- *ConceptDraw*: *ConceptDraw* enables the import of Visio models. Additionally to the import of models, *ConceptDraw* can import language elements (*Visio* masters). *ConceptDraw* indirectly imports the master elements via the model import. *ConceptDraw* also allows the export of models to *Visio* but there is no stencil support.
- *iGrafix*: This tool allows the import of *Visio* models by using the clipboard and the tool also imports stencil elements which are used by the imported model elements.

D. Degree of Interoperability

Table II shows the export and import connections of the investigated tools. The vertical axis is the source tool and the horizontal axis is the target tool. The source tool exports models and the target tool imports models. For instance, there is a directed connection from *Visio* to *Concept Draw*. That is, *Visio* can export models and *ConceptDraw* can import these *Visio* models. We differentiate the connections depending on their modeling level. The rectangle (\square) stands for an approach that supports the exchange of languages (meta-models) and models. The plus sign (+) represents an approach which supports only the exchange on model level. This includes the approach for the exchange of language specific-models and the approach for the generic exchange of models. We combine

both signs (\boxplus), if a tool supports exchange of languages and models as well as language-specific exchange or generic exchange of models.

The diagonal in the matrix shows that each tool allows the exchange of their own languages and models because each tool can save and load their own language definitions and models. Additionally, many tools allow the exchange on the model level (language-specific or generic model exchange). There are only three connections which allow the transformation of languages and models. The matrix also shows that *Visio* plays a key role because a lot of tools enable the import of *Visio* models.

All in all, there are $20 \times 20 = 400$ directed connections in this matrix. We assume that the export and import between the same tool is a basic feature in order to save and load models and languages. For this reason, we exclude the diagonal in our calculation. Thus, we have $20 \times 20 - 20 = 380$ possible connections. Out of these connections, there are 30 connections between different tools. This leads to a ratio of 7.9%. There are 27 connections that allow the exchange on model level, a ratio of 7.1%. Regarding the exchange on the language level, there are only three connections. This is a ratio of 0.8%. Hence, the degree of interoperability can be considered low.

E. Further Observations

Generally we identified different data formats for realizing the exchange of model data. One known exchange format for models is XML Metadata Interchange (XMI) [21]. We consider XMI in our study but we are not focus on XMI because their close relationship to MOF, EMF and UML. Other meta-modeling tools do not use XMI for realizing the exchange of their models and meta-models. Another mechanism to exchange models is the usage of graph formats such as Graph Exchange Language (GXL) [27] or GraphML [5]. Graph formats are suitable for exchange models because models and meta-models can be regarded as graphs. For instance, *MetaEdit+* uses an adapted version of GXL for serializing their models and meta-models, but no other tool in the study can import this graph format. *yEd* can import the GraphML format. A further observation is that some tools provide Excel exports. This is not for exchange reasons, but rather than a format to make reports. Beside the possibilities to exchange meta-models and models, there are a lot of language-specific formats, depending on the tool's domain. For instance, in this study many tools support typical formats from the business modeling domain such as BPMN-XML, BPEL [18] or XPD [28].

Another observation is that there are more tools allowing import than tools allowing the export of models. This could be a strategic reason. Most tools support the import because tool vendors want to increase their usage and often it is necessary to import data in order to replace other tools. The export is undesirable because the tool vendors try to bind their customers to a certain tool.

The last observation concerns the transformation capabilities. Some tools allow the definition of mappings between

TABLE II
MODEL AND LANGUAGE EXCHANGE (□=LANGUAGE LEVEL, +=MODEL LEVEL)

Tools	Agilian	ARIS BA	Atom ³	Business Process VA	ConceptDraw	Cubetto Toolset	Dia	Edraw Max	Enterprise Architect	GME	iGrafix Process	Lucidchart	Maram Meta-Tools	MetaEdit+	Microsoft Visio	PowerDesigner	ViFlow	VP for UML	VMSDK	yED
Agilian	⊕	+	+						+							+				
ARIS BA	⊕	+							+											
AToM ³			□																	
Business Process VA				⊕																
ConceptDraw					□										□					
Cubetto Toolset						□														
Dia							□								+					
Edraw Max								□												
Enterprise Architect	+	+		+					⊕									+		
GME										□										
iGrafix Process				+							□									
Lucidchart												□								
Maram Meta-Tools													□							
MetaEdit+														□						
Microsoft Visio	+			+	□		+	+	+		□	+			□	+		+		
PowerDesigner	+								+							⊕				
ViFlow																	□			
VP for UML	+	+		+					+									⊕		
VMSDK																			□	
yED																				□

models or languages in a simple and limited way. The tools do not provide powerful transformation languages such as ATL or ETL.

F. Threats to Validity

The interoperability between meta-modeling tools is between 0.8 and 7.9%. If we take a look at our study restrictions, the interoperability could be higher than these measured values. We focus on a limited set of tools in a selected domain. Maybe other tools in other domains have a higher interoperability degree. Furthermore we only looked for interoperability mechanisms that are provided by the tool itself. Some meta-modeling tools provide a generator which allows to generate every exchange format. Furthermore, we exclude external tools, such as BPM-X-Converter, which allows the migration of models between meta-modeling tools.

In contrast to this, we can argue for a lower value of interoperability. We only investigated the opportunity to import and export models. We cannot say anything about the quality of these exchange mechanisms. Furthermore, some tools relate very close to Visio. Hence, the import and export is easy for these tools. This is maybe similar to MOF-implemented meta-modeling tool. If we would not consider the Visio imports on language level, the interoperability would go against zero.

VI. CONCLUSION

In this paper we presented a study about interoperability between meta-modeling tools. The study included 20 tools in

the area of software development, business process modeling and data modeling. In the first part of this paper we defined the investigation scope. In the second part, we analyzed the tools and presented the results.

Regarding the first research question about the degree of interoperability, we can give the following answer. Depending of the approach considered, the degree of interoperability is low with values between 0.8% and 7.9%. The answer to the second question is more complicated because of the different approaches. Regarding the modeling level (section V-B), we identified the following three approaches: (1) the exchange of models which conform to a specific modeling language, (2) the generic exchange of each model without their modeling language, and (3) the exchange of models with their language. Besides the aspect of the modeling level, there are many other dimensions including many other approaches for interoperability.

In our future work we want to increase the degree of interoperability between meta-modeling tools. We assume a main reason for the missing interoperability is the heterogeneity between the different meta-modeling languages of the tools. Despite the heterogeneity, there are common concepts which can be mapped to each other. These similarities and differences are described in a comparative analysis between different meta-modeling languages [12]. Based on this, we can develop a transformation-based adapter approach in order to enhance the model and meta-model exchange.

APPENDIX

TABLE III
MODELING AND META-MODELING TOOLS (●=YES, --=NO)

Name	Vendor	Version	Meta-modeling approach (light/heavy)	Included in study
Agilian	Visual Paradigm	4	●/●	●
Altova UModel	Altova	2012	●/--	--
ArgoUML		0.34	●/--	--
Archi	University of Bolton	2.3	--/--	--
ARIS Business Architect	Software AG	7.1	●/●	●
ARIS Express	Software AG	2.3	--/--	--
Artisan Studio	Atego	7.4	●/--	--
Astah	Astah	6.6.3	●/--	--
AToM3	McGill University	2008	--/●	●
bflow* Toolbox		1.2.5a	--/--	--
Bizagi Process Modeler	Bizagi	2.3	--/--	--
BOUML	Bruno Pagès		--/--	--
Business Process Visual Architect	Visual Paradigm	5	--/●	●
Cadifra UML Editor	A. & F. Buehlmann	1.3.3	--/--	--
CaseComplete	Serlio Software	7.0 (2012)	--/--	--
ConceptDraw	CS Odessa	9	--/●	●
Cubetto Toolset	Semture	1.7.1	--/●	●
Database Design Tool		1.5	--/--	--
DB Wrench	Nizana Systems	2.3.0	--/--	--
dbConstructor	DBDeveloper Solutions		--/--	--
DbSchema	Wise Coders Solutions		--/--	--
Dia		0.97.2	--/●	●
Edraw Max	EdrawSoft	6.3	--/●	●
Enterprise Architect	Sparx Systems	9.3	●/●	●
ER Creator	modelCreator Software	3.0	--/--	--
ER/Studio Software Architect	Embarcadero Technologies	1.1.0	●/--	--
ER/Studio Business Architect	Embarcadero Technologies	1.7.0	--/--	--
Generic Modeling Environment	Vanderbilt University	10.8	--/●	●
Gliffy	Gliffy		--/--	--
Grapholite	Perpetuum Software	1.6.0.7	--/--	--
iGrafix Process	iGrafix	2011	--/●	●
Intalio BPMS Designer	Intalio	6.1.12	--/--	--
Lucidchart	Lucid Software		--/●	●
MagicDraw	NoMagic	17.0.2	●/--	--
Maram Meta-Tools	University of Auckland	--	--/●	●
MetaEdit+	MetaCase	5.0	--/●	●
Microsoft Visio	Microsoft	2010 (14)	--/●	●
Modelio	Modeliosoft	2.1.1	●/--	--
NClass	Balazs Tihanyi	2.04	--/--	--
Objectteering	Objectteering Software	6.1	●/--	--
objectiF	microTOOL	7.1	●/--	--
Open ModelSphere	Grandite	3.2	●/--	--
ORM Designer	Inventic		--/--	--
Poseidon for UML	Gentleware	8	--/--	--
Papyrus	Eclipse	1.12	●/--	--
PowerDesigner	Sybase	16.1	●/●	●
Process Modeler	itp commerce	5	--/--	--
RISE	RISE to Bloome Software	4.5	--/--	--
Select Architect	Select Business Solutions		●/--	--
SemTalk	Semtation	4	--/--	--
Signavio Process Editor	Signavio	6.0	--/--	--
SmartDraw	SmartDraw Software		--/--	--
Topcased		5.2	●/--	--
UML Lab	Yatta Solutions	1.4.3	●/--	--
UMLet		11.5.1	--/--	--
ViFlow	ViCon		--/●	●
Violet UML Editor		0.21.1	--/--	--
Visual Paradigm for UML	Visual Paradigm	9	●/●	●
Visual Use Case	TechnoSolutions	4.069 (2009)	--/--	--
Visualization and Modeling SDK	Microsoft	VS2012	--/●	●
WinA&D	Excel Software		--/--	--
Xcase	Resolution Software	9.1	--/--	--
yED	yWorks	3.9.2	--/●	●

TABLE IV
IMPORT AND EXPORT FORMAT OF META-MODELING TOOLS

Agilian	
Import	Rational Rose (mdl) files, Rational DNX files, BizAgi project file, specific XML, XMI (1.2, 2.1), Eclipse UML2 (XMI 2.1), Visual Paradigm project file, MS Excel file with specific schema, Visio drawings, Visio ERD, Visio drawing/stencils into Agilian Stencil, NetBeans 6.x UML diagrams, Telelogic System Architect, Telelogic Rhapsody, PowerDesigner project file
Export	BPMN2.0-XML, specific XML, XMI (1.2, 2.1), Eclipse UML2 (XMI 2.1), Visual Paradigm project file, MS Excel file with specific schema, VPP (ZIP project archiv)
ARIS Business Architect	
Import	XML with specific schema, UML (XMI1.1)
Export	ADF (ARIS filter), XMI, XML, Visio (VDX), BPEL, ADB (ARIS database)
Business Process Visual Architect	
Import	BizAgi project file, XML, BPMN2.0-XML, XPDL2.1, Telelogic System Architect, Excel, Visio
Export	BPMN 2.0 XML, XML, BPMN2.0-XML, XPDL2.1, Excel
ConceptDraw	
Import	Visio (VDX), MS PowerPoint
Export	CDX file (XML), Visio (VDX), MS PowerPoint
Cubetto Toolset	
Import	-
Export	ETZ format
Dia	
Import	Visio models, Dia, Dxf (specific XML file), SVG, Xfig
Export	Visio models, Dia, Dxf (specific XML file), SVG, Xfig
Edraw Max	
Import	Visio
Export	
Enterprise Architect	
Import	Database Schema, specific Visio models (Communication, Activity, Class, Object, Component, Deployment, Custom), Doors, XMI (UML 1.1, 1.3 or 2.x), ARCGIS, ODM (OWL/RDF), Rhapsody, Rational Software Architect (EMX/UML2)
Export	XMI 1.0 (UML1.3), XMI 1.1 (UML1.3), XMI 1.2 (UML1.4), XML 2.1 (UML2.0), MOF1.4 (XMI1.2), MOF1.3 (XMI1.1), specific XML, Ecore, OWL/RDF, BPMN2-XML
iGrafix Process	
Import	Visio models and metamodels
Export	BPEL XML, XPDL, XML
Lucidchart	
Import	Visio models (vdx, vsd, vsdx)
Export	Visio models (vdx)
MetaEdit+	
Import	GXL-adapted (models and meta-models)
Export	GXL-adapted (models and meta-models)
Microsoft Visio	
Import	-
Export	-
PowerDesigner	
Import	Excel, ERwin, XMI, Rational Rose (MDL), SIMUL8 file, specific Visio models
Export	UML2, XMI2.1 XML schema files
Visual Paradigm for UML	
Import	ERWin Data Modeler project files, BizAgi project file, System Architect business process diagram, XMI (1.2, 2.1), Excel, Visio, Visio ERD, Visio diagram to Stencil, Rational Rose (MDL) files, Rational DNX files, Rational Software Architect files, PowerDesigner project file, Telelogic Modeler
Export	BPEL, XPDL, JPDL, BPMN2.0-XML, XMI (1.2, 2.1), Excel, SCXML
yEd	
Import	Graph Markup Language (GRAPHML), yWorks Binary Graph Format, Graph Modeling Language (GML, XGML), Trivial Graph Format (TGF), Gedcom Data (GED)
Export	-

REFERENCES

- [1] *Webster's Third New International Dictionary*. Merriam Webster, 1986.
- [2] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. IEEE, 1991.
- [3] *Longman Dictionary of Contemporary English*, 5th ed. Langenscheidt ELT, February 2009.
- [4] B. Biafore, *Visio 2007 Bible*. Wiley Publishing, April 2007.
- [5] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. Marshall, "GraphML Progress Report Structural Layer Proposal," in *Graph Drawing*, ser. Lecture Notes in Computer Science, P. Mutzel, M. Jünger, and S. Leipert, Eds. Springer Berlin Heidelberg, 2002, vol. 2265, pp. 501–512. [Online]. Available: http://dx.doi.org/10.1007/3-540-45848-4_59
- [6] David Chen (ed.), "Practices, principles and patterns for interoperability (Deliverable D6.1)," Network of Excellence - Contract no.: IST-508 011, Tech. Rep., May 2005.
- [7] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd ed. Addison-Wesley, September 2003.
- [8] D. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley Publishing, January 2003.
- [9] C. Hein, T. Ritter, and M. Wagner, "Model-Driven Tool Integration with ModelBus," in *First International Workshop on Future Trends of Model-Driven Development, FT added*, 2009, pp. 35–39.
- [10] F. Jouault and I. Kurtev, "Transforming Models with ATL," in *Proceedings of the 2005 International Conference on Satellite Events at the MoDELS*. Berlin, Heidelberg: Springer, 2006, pp. 128–138. [Online]. Available: http://dx.doi.org/10.1007/11663430_14
- [11] S. Kelly and J.-P. Tolvanen, *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley-IEEE Computer Society, March 2008.
- [12] H. Kern, A. Hummel, and S. Kühne, "Towards a Comparative Analysis of Meta-metamodels," in *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPEs'11, NEAT'11, VMIL'11*, ser. SPLASH '11 Workshops. New York, NY, USA: ACM, 2011, pp. 7–12. [Online]. Available: <http://doi.acm.org/10.1145/2095050.2095053>
- [13] H. Kern and S. Kühne, "Integration of Microsoft Visio and Eclipse Modeling Framework Using M3-Level-Based Bridges," in *2nd ECMDA Workshop on Model-Driven Tool and Process Integration at Fifth European Conference on Model-Driven Architecture Foundations and Applications 2009*, Enschede, Netherlands, 2009.
- [14] D. Kolovos, R. Paige, L. Rose, and F. Polack. (2014, April) The Epsilon Book. [Online]. Available: <http://www.eclipse.org/epsilon/doc/book/>
- [15] A. Ledeczki, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi, "The Generic Modeling Environment," in *Workshop on Intelligent Signal Processing*, 2001. [Online]. Available: <http://www.cs.virginia.edu/~rp2h/home/research/ReadingList/gmepaper.pdf>
- [16] D. S. Linthicum, *Enterprise Application Integration*. Addison-Wesley, 1999.
- [17] A. Molina, H. Panetto, D. Chen, L. Whitman, V. Chapurlat, and F. Vernadat, "Enterprise Integration and Networking: Challenges and Trends," *Studies in Informatics and Control*, vol. 16, no. 4, pp. 353–368, 2007.
- [18] *Web Services Business Process Execution Language Version 2.0*, OASIS Std., April 2007. [Online]. Available: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- [19] *Business Process Model And Notation (BPMN), Version 2.0*, Object Management Group Std., January 2011. [Online]. Available: <http://www.omg.org/spec/BPMN/2.0/PDF>
- [20] *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, V1.1*, Object Management Group Std., January 2011. [Online]. Available: <http://www.omg.org/spec/QVT/1.1/>
- [21] *XML Metadata Interchange (XMI) Specification, Version 2.4.2*, Object Management Group Std., April 2014. [Online]. Available: <http://www.omg.org/spec/XMI/2.4.2>
- [22] T. Stahl and M. Völter, *Model-Driven Software Development*. Wiley, May 2006.
- [23] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed., ser. The Eclipse Series. Addison-Wesley, December 2008.
- [24] J.-P. Tolvanen, "Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence," Ph.D. dissertation, University of Jyväskylä, 1998.
- [25] A. I. Wasserman, "Tool integration in software engineering environments," in *Software Engineering Environments*, ser. Lecture Notes in Computer Science, F. Long, Ed. Springer, 1990, vol. 467, pp. 137–149. [Online]. Available: http://dx.doi.org/10.1007/3-540-53452-0_38
- [26] M. N. Wicks, "Tool Integration within Software Engineering Environments: An Annotated Bibliography," Heriot-Watt University, Tech. Rep., August 2006. [Online]. Available: <http://www.macs.hw.ac.uk/cs/techreps/docs/files/HW-MACS-TR-0041.pdf>
- [27] A. Winter, B. Kullbach, and V. Riediger, "An Overview of the GXL Graph Exchange Language," in *Software Visualization: International Seminar Dagstuhl Castle, Germany, May 20–25, 2001 Revised Papers*, ser. Lecture Notes in Computer Science. Springer, 2002, pp. 324–336.
- [28] *Process Definition Interface – XML Process Definition Language, Version 2.2*, Workflow Management Coalition Std., August 2012. [Online]. Available: [http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20\(2012-08-30\).pdf](http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf)