Several Issues on the Model Interchange Between Model-Driven Software Development Tools

Una Ieva Zusane, Oksana Nikiforova, Konstantins Gusarovs
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
{una.zusane, oksana.nikiforova, konstantins.gusarovs}@rtu.lv

***Abstract* — Models are widely used and are one of the advanced tools of software engineering. There is a necessity to export software models from one software development tool or environment and to import them into another tool or environment, especially actual this task is within the Model-Driven Software Development. Despite of the popular model description standard XML Metadata Interchange (XMI), which can be used to perform the task of the model interchange, several problems according to information loss still are appearing. The research is devoted to the comparison of the tools' abilities to exchange the software model presented in the form of Unified Modeling Language (UML) diagrams with other tools, based on a set of test cases suitable to check the completeness of the model description according to XMI standard. Authors open the discussion about the dependency between tools correspondence to the XMI standard and tool's ability of model interchange.**

Keywords – model interchange; UML diagrams; model-driven software development tool.

I. Introduction

In software development projects, models are wildly used because they are not only visually easier comprehensible in requirement gathering and design phase, but also model transformations turn them into useable artefacts in implementation phase. Models are usually portrayed as diagrams [1], however they can also be written in a textual modeling language.

Model Driven Software Development (MDSD) uses abstractions provided by models to develop software systems [2]. The development process begins with higher level of abstraction, which is continuously transformed to more detailed levels of abstraction until final system is developed.

During the process there may be a need for model interchange between modeling tools. One scenario is that computation independent model may be designed in one modeling tool, and further work on platform specific model should continue in another tool. Another scenario may be a change of modeling tools in the software development lifecycle due to tools' pricing or available options.

XMI [3] is a popular model interchange standard, which is implemented by many modeling tools. XMI was developed by Object Management Group to improve model interchange abilities between different modeling tools. UML [4] models that are portrayed visually in MDSD support tools, can be converted to text conforming to XMI standard.

In an ideal world model, interchange process should be straight forward, if two tools use the same standard for model interchange. The standard could define precise requirements for interchangeable metadata structures, so that there would be little or no variation for possible. This would provide foundation for errorless model interchange process. However, often there is still a loss of data during the model interchange process. This is caused by different interpretations of the XMI standard, as well as tools' wish to extend XMI with model layout information and different other extensions.

The goal of the research is to evaluate whether there is a dependency between the amount of warnings and errors discovered in MDSD support tools exported XMI file and a modeling tools' XMI model interchange ability. Research consists of two parts. Firstly, tools' ability to export files conforming to XMI standard is evaluated. Secondly, models for three test cases are practically exchanged between tools.

The rest of the paper is structured as follows. Section II describes the importance of model interchange in the MDSD process. In Section III, the National Institute of Standards and Technology (NIST) XMI validation tool is considered as a way to determine the quality of MDSD tools' exported XMI model. The next Section analyses the results of practical model interchange between the modeling tools. Related work is considered in Section V. In the conclusion Section, the results of the research are summarized and the directions of future research are suggested.

II. The Task of the Model Interchange within the Model-Driven Software Development

Model is an integral part of MDSD [2], because it can portray different levels of abstraction [1]. MDSD support tools need to have a reliable model interchange ability in order to preserve their users. If a tool cannot cooperate with other development tools, users will have to do a lot of unnecessary manual work, which may lead them to choose another tool for modeling.

Models usually are portrayed visually and saved in MDSD support tools in different formats, which depend on the technology used in the building process of a tool. Some modeling tools can generate XMI files for models, which mean transforming visual models to text. Other tools have an ability to import XMI files, transforming a model from a text file to tools native form of a model. This process is restricted by modeling standards (e.g., UML) and XMI standard rules.

XMI standard provides a set of rules how to write a models' metadata information in Extensible Markup Language (XML) [3]. XML was introduced in 1996 by World Wide Web Consortium [5] with a goal of simplifying data exchange process between different software tools.

In research, three MDSD support tools are reviewed in order to evaluate their ability of model interchange – Enterprise Architect [6], Magic Draw [7] and Modelio [8]. These tools were selected by Model Interchange Working Group in 2011 as ones to be evaluated by Model Interchange Tests [10].

Enterprise Architect is developed by Sparx Systems and has more than 350 000 users in 160 countries [6]. It is a visual modeling tool, which is based on Model Driven Architecture approach developed by Object Management Group. There are numerous modeling standards available in this tool, for example, UML, Business Process Model and Notation (BPMN) and System Modeling Language (SysML). The user interface in Enterprise Architect is user friendly and intuitive, which boosts the usage productivity. The current edition of Enterprise Architect is 12.

Magic Draw is a modeling tool developed by No Magic [7] in order to support object-oriented systems analysis and design. The tool incorporates such industry standards as UML, SysML and BPMN. Magic Draw supports code generation from models to different programming languages (Java, C++, C# and CORBA IDL). The current edition of Magic Draw is 18.1.

Modelio is an open source modeling tool developed by Modeliosoft [8]. The tool is designed for business and system analysts, as well as software developers. Tool consists of modules that can be added to default version of the tool according to user needs. The tool supports code generation from models to Java language. The current edition of Modelio is 3.3.1.

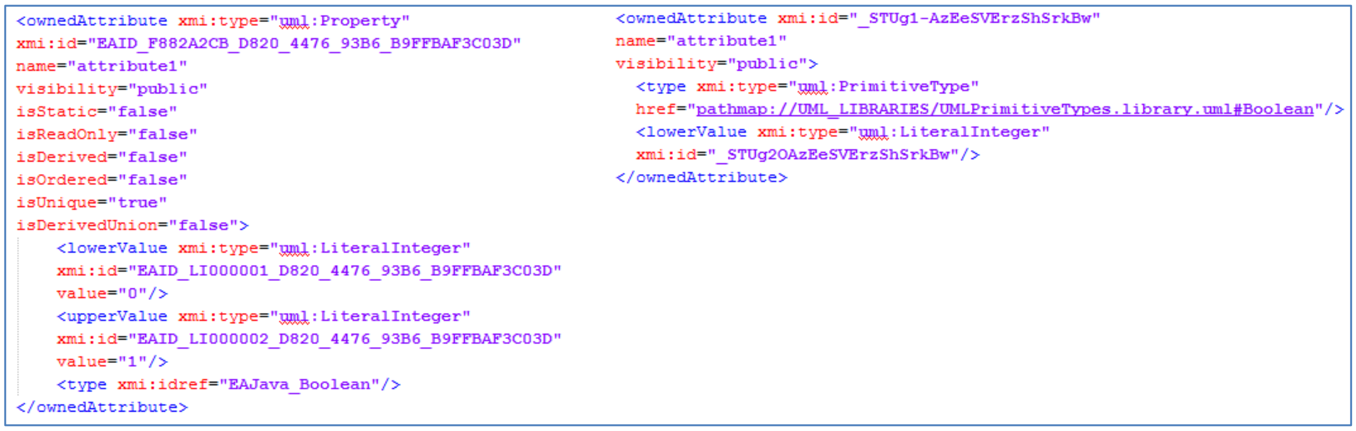
It is possible to export models in XMI format files from Enterprise Architect and Modelio. On the other hand, Magic Draw can export Models only in XML file. This means that all three exported models are written in XML language and more or less conforms to XMI standard. The model interchange problems arise when tool A does not have appropriate transformation rules for XML structures, which are used in tool B.

As an example of differences in tools' interpretation of XMI standard and the way of generating models for interchange purposes, we will look at a small example. In an UML class there is an attribute with name "attribute 1". It is type "Boolean" and can take values from 0 to 1.

In the right part of Figure 1, a fragment of class diagram export from Enterprise Architect can be seen. In addition to previously mentioned characteristics attribute 1 has some more metadata, which are included in Enterprise Architect models.

In the left part of Figure 1 the same fragment of attribute1 exported from Modelio can be seen. This fragment is considerably shorter, as Modelio by default doesn't create as many additional characteristics for an attribute. It is also unclear what the value restriction for this attribute is – this will be considered as a difference between the uploaded XMI file and the "valid XMI" for the test case by NIST validation tool.

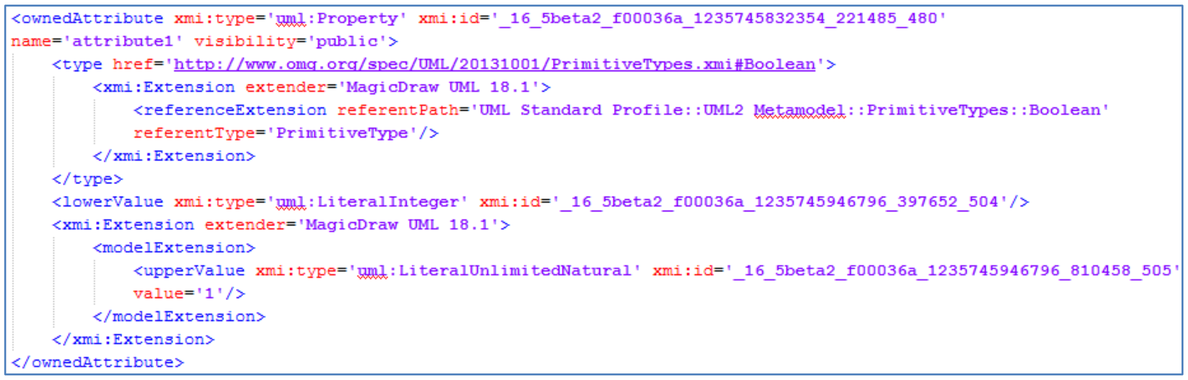
Magic Draw exported a file with an extension .xml for class diagram. This tool wildly uses extensions, which make the text form of attribute1 in Figure 2 differ even more from other discussed tools. However, when NIST validation tool compares Magic Draw XML's canonical XMI form to "valid XMI" in Section III the results are good and the amount of discovered validation errors is low in comparison with other tools.



```
<ownedAttribute xmi:type="uml:Property"
xmi:id="EAID_F882A2CB_D820_4476_93B6_B9FFBAF3C03D"
name="attribute1"
visibility="public"
isStatic="false"
isReadOnly="false"
isDerived="false"
isOrdered="false"
isUnique="true"
isDerivedUnion="false">
    <lowerValue xmi:type="uml:LiteralInteger"
    xmi:id="EAID_LI000001_D820_4476_93B6_B9FFBAF3C03D"
    value="0"/>
    <upperValue xmi:type="uml:LiteralInteger"
    xmi:id="EAID_LI000002_D820_4476_93B6_B9FFBAF3C03D"
    value="1"/>
    <type xmi:idref="EAJava_Boolean"/>
</ownedAttribute>
```

```
<ownedAttribute xmi:id="_STUg1-AzEeSVErzShSrkBw"
name="attribute1"
visibility="public">
  <type xmi:type="uml:PrimitiveType"
  href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#Boolean"/>
  <lowerValue xmi:type="uml:LiteralInteger"
  xmi:id="_STUg2OAzEeSVErzShSrkBw"/>
</ownedAttribute>
```

Figure 1. Comparison of XMI atribute1 in Enterprise Architect and Modelio



```
<ownedAttribute xmi:type='uml:Property' xmi:id='_16_5beta2_f00036a_1235745832354_221485_480'
name='attribute1' visibility='public'>
    <type href='http://www.omg.org/spec/UML/20131001/PrimitiveTypes.xmi#Boolean'>
        <xmi:Extension extender='MagicDraw UML 18.1'>
            <referenceExtension referentPath='UML Standard Profile::UML2 Metamodel::PrimitiveTypes::Boolean'
            referentType='PrimitiveType'/>
        </xmi:Extension>
    </type>
    <lowerValue xmi:type='uml:LiteralInteger' xmi:id='_16_5beta2_f00036a_1235745946796_397652_504'/>
    <xmi:Extension extender='MagicDraw UML 18.1'>
        <modelExtension>
            <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='_16_5beta2_f00036a_1235745946796_810458_505'
            value='1'/>
        </modelExtension>
    </xmi:Extension>
</ownedAttribute>
```

Figure 2. XMI atribute1 in Magic Draw

All three MDSD support tools discussed in this paper have different ways of writing an attribute from a class diagram in XML language. When it comes to more complicated parts of models, these differences become increasingly important in the context of model interchange.

III. NIST VALIDATION TOOL

In 2009, Object Management group announced the creation of Model Interchange Working Group (MIWG) [11]. MIWG has created a test suit, which consists of 40 test cases [9]. The test suit allows demonstrating model interchange abilities of several modeling tools. 25 of the tests are defined for UML 2.3 standard. Each test case consists of one or more diagrams and according XMI file, which conforms to XMI standard and is considered as a "valid XMI" for the model. This XMI file is used in the validation process, when the exported XMI files from modeling tools are compared to it.

USA National Institute of Standards and Technology (NIST) [12] has developed a validation tool that can validate an XMI file exported from a modeling tool against the "valid XMI" for a chosen test case. XMI is compared in its canonical form. There are various ways how model can be represented in XMI, which all conform to XMI standard [9]. Canonical XMI has additional points in its specification that eliminates variation. There is only one way in which a model can be correctly represented in the canonical form. Usage of canonical XMI makes it possible to compare two XMI models expressed in it to find the differences. No tools used in the research exports canonical XMI form directly. NIST validation tool converts uploaded XMI files to their canonical form before comparing them to the "valid XMI".

After the validation of an XMI file the summary or results is displayed to the user. In the heading there is information about XMI file: XMI version, object count in the XMI file, used meta-model. It is followed by a list of warnings, which arises when XMI does not fully conform to the Object management groups' developed standard. Warnings may cause problems with model interchange, because the importing modeling tool may not interpret these parts of XMI correctly.

There are two parts of validation errors discovered by the NIST validation tool [12]:

- General errors;
- Differences between the uploaded XMI file and the "valid XMI" for the test case.

If an XMI that is independent from all test cases provided by MIWG is validated by NIST validation tool, only general errors will be displayed.

In order to see differences in the tools' ability to export models, we compared tools in two dimensions.

Firstly, there are several XMI files available from MIWG interchange tests in 2011 [10]. They were exported from MDSD support tools described in the previous Section. All the tools have developed new versions since then, so it is possible to compare the older version of a tool with the new one. Version numbers for tools described in this paper are shown in Table 1.

TABLE I. MODELING TOOL VERSIONS

	Enterprise Architect	MagicDraw	Modelio
Year 2011	9.1	17.0	2.4.19
Year 2015	12.0	18.1	3.3.1

Secondly, all the tools have exported models from the same test cases. This gives the grounds to compare the number of validation errors discovered by the NIST validation tool between the different MDSD support tools.

The comparison of tools in this Section uses three test cases for UML diagrams: class diagram [13], activity diagram [14] and use case diagram [15]. The choice of test cases covers both behavioral and structural UML diagrams.

Class diagrams describe structure of a system showing objects used in a system as classes. Each class can have attributes (characteristics of an object) and methods (actions that an object can do). Classes are linked with each other with different relationships, e.g., association and generalization.

Activity diagrams are used for business process modeling. They display the sequence of activities in a workflow and decisions resulting from activities.

Use case diagrams show user interaction with the system. Users are portrayed as actors and they are connected to use cases by using different links to specify the relation.

Class diagram is the first test case to be analyzed. The comparison of MDSD support tools and data from the tools' versions from years 2011 and 2015 is shown in Figure 3. The amount of validation errors discovered by the NIST validation tool for the class diagram is displayed there.

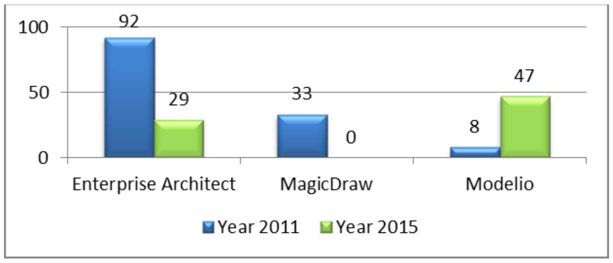


Figure 3. Comparison of validation errors in the class diagram

In the 12.0 version of Enterprise Architect the exported XMI conforms better with the XMI standard than the version 9.1. The amount of validation errors has decreased three times. In the Magic Draw version 17.0 there are 33 validation errors, but in the version 18.1 NIST validation tool did not discover any errors. In Modelio the trend is reversed. In the version 2.4.19 there were 8 validation errors, but in the version 3.3.1 the NIST validation tool discovered 47 validation errors in the exported XMI of the class diagram test case. When making a comparison between different tools, the best in class diagram test case is Magic Draw, which is followed by Enterprise Architect and Modelio. The amounts of validation errors in these tools are increasingly higher.

Activity diagram is the second test case to be analyzed. The comparison of MDSD support tools and data from the tools' versions from years 2011 and 2015 can be seen in Figure 4. The amount of validation errors discovered by the NIST validation tool for the activity diagram is displayed in Figure 4.

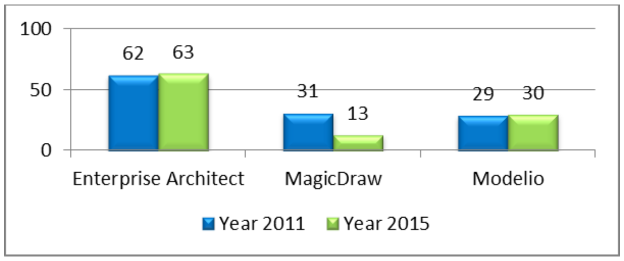


Figure 4. Comparison of validation errors in the activity diagram

When comparing older and newer versions of Enterprise Architect an increase by one validation error can be seen in the version 12.0. Magic draw in newer version 18.1 has 18 validation errors less than version 17.0. Modelio, similarly as Enterprise Architect, in the version 2.4.19 has one validation error more than there was in the version 3.3.1. In the export of activity diagrams Magic Draw has the best results with 13 validation errors discovered in the current tool's version. Modelio has approximately three times more validation errors than Magic Draw, but the highest amount of validation errors belongs to Enterprise Architect.

Use case diagram is the third test case to be analyzed. The comparison of MDSD support tools and data from the tools' versions from years 2011 and 2015 can be seen in Figure 5. The amount of validation errors discovered by the NIST validation tool for the use case diagram is displayed in Figure 5.

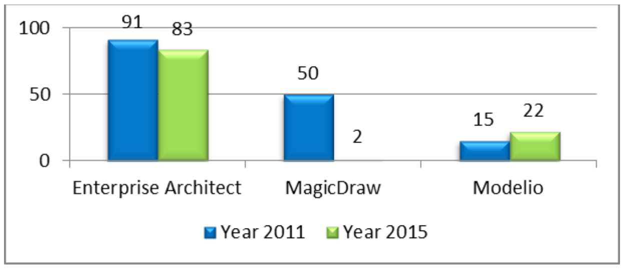


Figure 5. Comparison of validation errors in the use case diagram

The use case diagram XMI file, which is exported from Enterprise Architect version 12.0, has 83 validation errors. That is by 8 validation errors less than Enterprise Architect version 9.1. There is even better improvement in Magic Draw – the amount of validation errors from 50 in version 17.0 has decreased to only 2 in version 18.1. For Modelio, similarly as in the case of class and activity diagrams, an increase in the amount of validation errors for the newer version 3.3.1. can be seen. Magic Draw with its 2 validation errors has the lowest amount of errors for the use case diagram. It is followed by Modelio (22 validation errors) and Enterprise Architect (83 validation errors).

There were various validation errors discovered by NIST validation tool. In the error messages user uploaded XMI file is referenced as "User.xmi" and preloaded XMI for the test case is referenced as "Valid.xmi". The most frequent validation errors were:

- User.xmi is missing an element present in Valid.xmi;
- User.xmi contains an element not present in Valid.xmi;
- An object property value in User.xmi differs from that of Valid.xmi, for example in User.xmi class visibility is defined as "Public", but in Valid.xmi the value is null;
- User object missing a value specified in Valid.xmi.

The highest amount of general errors was about the serialization of a default value.

The summary of all the amounts of validation errors from three test cases for each tool is shown in Figure 6. Summary is made for the tool versions in year 2015.

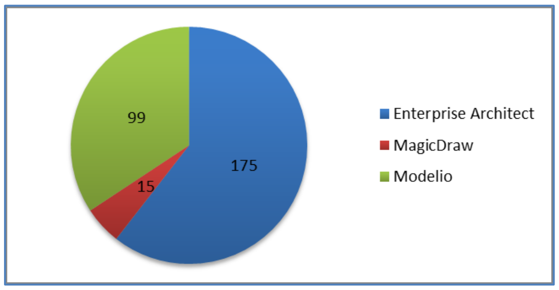


Figure 6. Summary of model validation errors

Enterprise Architect has 175 validation errors, which adds up to 61% from the total amount of validation errors. The majority of validation errors were about the serialization of a default value. Enterprise Architect specifies the visibility of a public class, where in XMI it is considered a default value for class visibility and should not be specified.

Modelio has half the amount of validation errors. Modelio has a mentionable trend to become less conformant with the XMI standard in the newer version. In all test cases, the amount of validation errors for Modelio version 3.3.1 was higher than for version 2.4.19.

In each test case Magic Draw had the lowest amount of validation errors. Only 5% of the total amount of validation errors was created by Magic Draw.

IV. RESULTS OF THE MODEL INTERCHANGE BETWEEN THE TOOLS

In order to evaluate, whether a model exported form one of described tools can be used in other tools, we tested model interchange practically. In a perfect scenario model interchange should provide a possibility to export a model from one tool and import model in another tool without losing any elements, links and layout.

For each test case analyzed in Section III we practiced model interchange between the described tools and evaluated it according to these criteria:

0 points – model from one tool cannot be imported in another tool;

1 point – model can be imported from tool A into tool B, but it is missing some elements or links;

2 points - model can be imported from tool A into tool B and it has all elements and links;

3 points - model can be imported from tool A into tool B and it has all elements, links and layout.

Results for each test case are displayed in the tables below. Tools named in listed exported diagrams that were imported in tools listed in rows.

TABLE II. CLASS DIAGRAM MODEL INTERCHANGE

From / To	Enterprise Architect	Magic-Draw	Modelio
Enterprise Architect	X	3	2
MagicDraw	2	X	2
Modelio	1	0	X

Model interchange results for class diagram are shown in Table 2. Modelio was missing some elements in the diagram exported from Enterprise Architect and could not import model from Magic Draw at all. Enterprise Architect could retrieve model layout exported by Magic Draw.

TABLE III. ACTIVITY DIAGRAM MODEL INTERCHANGE

From / To	Enterprise Architect	Magic-Draw	Modelio
Enterprise Architect	X	3	2
MagicDraw	1	X	2
Modelio	2	0	X

Model interchange results for activity diagram are shown in Table 3. Model did not have all the elements and links in interchange between Enterprise Architect and Magic Draw. Other tools received complete model from Modelio, but did not get the layout.

TABLE IV. USE CASE DIAGRAM MODEL INTERCHANGE

From / To	Enterprise Architect	Magic-Draw	Modelio
Enterprise Architect	X	3	2
MagicDraw	2	X	2
Modelio	2	0	X

Model interchange results for use case diagram are shown in Table 4. All model interchanges that were functional transported complete models from one tool to another. The only interchange that did not work was from Magic Draw to Modelio.

All the obtained points for both import and export of three test case models are summarized in Figure 7.

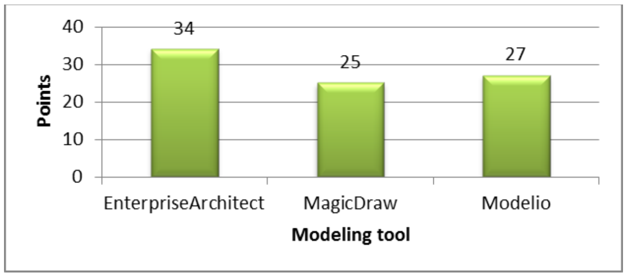


Figure 7. Comparison of practical model interchange

According to previously raised criteria Enterprise Architect has the best ability of model interchange with other modeling tools used in this research. Tools option to import files in XMI, as well as XML formats and ability to take model layout is an advantage.

Modelio has 7 points less than Enterprise Architect. Modelio is best evaluated for the ability to export a model, which can be imported in all other tools with high accuracy.

Modelio is followed by Magic Draw. Tools biggest flaw was its inability to import models layout. Magic Draw offers its users a variety of automatic layout options for models, during practical tests it was recognized that it was not enough. The automatic layout option did not work for use case diagram.

V. CONCLUSION

In this paper the UML model interchange capabilities of three modeling tools were analyzed. The tools are: Enterprise Architect, Magic Draw and Modelio. Three test cases designed by MIWG were used: class diagram, activity diagram and use case diagram.

With the NIST validation tool the largest amount of validation errors were discovered in the XMI files of Enterprise Architect, but the smallest amount of validation errors were in Magic Draw exported models. When analyzing the trends of development from older to newer versions of tools it can be seen that there is improvement in the amount of validation errors in Enterprise Architect and Modelio. Both tools in year 2015 have less validation errors than they had in year 2011.

In practical model interchange Enterprise Architect is recognized as the most precise of the analyzed tools. It is followed by Modelio and Magic Draw. This result seems to be counterintuitive: Enterprise Architect has the highest amount of validation errors discovered by NIST validation tool, yet it has the best model interchange ability. This could be explained by the tools import abilities, which cannot be evaluated by NIST validation tool. It is also evident that not all validation errors have negative impact on tools ability of model interchange.

In conclusion, the dependency between the amount of XMI validation errors and tools' practical model interchange ability is not evident. The amount of XMI validation errors discovered by NIST validation tool is not enough to determine, whether a MDSD support tool will have good model interchange ability.

One explanation for this result is that only the quality of exported XMI files can be tested by NIST validation tool. Unfortunately, a good conformance to XMI standard does not insure that other modeling tools will import the file successfully. When testing a tools ability of model interchange, both the quality of the export XMI and import to other tools should be examined.

The research can be continued in two directions. Firstly, more MDSD support tools can be compared using the same test cases for UML diagrams. Secondly, the validation errors discovered by NIST validation tool can be analyzed in order to determine, which have significant impact on model interchange.

ACKNOWLEDGMENT

The research presented in the paper is supported by Latvian Council of Science, No. 342/2012 "Development of Models and Methods Based on Distributed Artificial Intelligence, Knowledge Management and Advanced Web Technologies".

REFERENCES

[1] T. Kuhne, What is a Model? Internat. Begegnungs-und Forschungszentrum für Informatik. 2005

[2] S. Beydeda, M. Book, and V. Gruhn, Model-Driven Software Development, Springer, 2005.

[3] XML Metadata Interchange (XMI) Specification. Available: http://www.omg.org/spec/XMI/ [retrieved: July, 2015].

[4] Unified Modeling Language (UML) Resource Page. Available: http://www.uml.org/ [retrieved: July, 2015].

[5] Extensible Markup Language (XML) 1.0 (Fifth Edition). Available: http://www.w3.org/TR/2008/REC-xml-20081126/, [retrieved: July, 2015].

[6] Enterprise Architect. Available: http://www.sparxsystems.com/products/ea/ retrieved: July, 2015].

[7] MagicDraw. Available: http://www.nomagic.com/products/magicdraw.html retrieved: July, 2015].

[8] Modelio. Available: https://www.modelio.org/about-modelio/features.html retrieved: July, 2015].

[9] Model Interchange Wiki. Available: http://www.omgwiki.org/model-interchange/doku.php?id= [retrieved: July, 2015].

[10] UML/SysML Tool Vendor Model Interchange Test Case Results Now Available. Available: http://www.omg.org/news/releases/pr2011/12-01-11.htm, retrieved: July, 2015].

[11] S. Covert, OMG Announces Model Interchange Working Group. Available: http://www.omg.org/news/releases/pr2009/07-08-09.htm, retrieved: July, 2015].

[12] NIST XMI validator. Available: http://validator.omg.org/se-interop/tools/validator, retrieved: July, 2015].

[13] Test Case 1 - Simple Class Model. Available: http://www.omgwiki.org/model-interchange/doku.php?id=test_case_1_uml_2.3, retrieved: July, 2015].

[14] Test Case 4 - Simple (fUML) Activity Model. Available: http://www.omgwiki.org/model-interchange/doku.php?id=test_case_4_uml_2.3, retrieved: July, 2015].

[15] Test Case 8 - Use Cases. Available: http://www.omgwiki.org/model-interchange/doku.php?id=test_case_8_uml_2.3, retrieved: July, 2015].