INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 11(2):233-247, April 2016.

Content Based Model Transformations: Solutions to Existing Issues with Application in Information Security

J. Janulevičius, S. Ramanauskaitė, N. Goranin, A. Čenys

Justinas Janulevičius*, Simona Ramanauskaitė, Nikolaj Goranin, Antanas Čenys
Vilnius Gediminas Technical University,
Lithuania, LT-10223 Vilnius, Sauletekio al. 11
justinas.janulevicius@vgtu.lt, simona.ramanauskaite@vgtu.lt, nikolaj.goranin@vgtu.lt,
antanas.cenys@vgtu.lt
*Corresponding author: justinas.janulevicius@vgtu.lt

Abstract: Model-Driven Engineering uses models in various stages of the software engineering. To reduce the cost of modelling and production, models are reused by transforming. Therefore the accuracy of model transformations plays a key role in ensuring the quality of the process. However, problems exist when trying to transform a very abstract and content dependent model. This paper describes the issues arising from such transformations. Solutions to solve problems in content based model transformation are proposed as well. The usage of proposed solutions allowing realization of semi-automatic transformations was integrated into a tool, designed for OPC/XML drawing file transformations to CySeMoL models. The accuracy of transformations in this tool has been analyzed and presented in this paper to acquire data on the proposed solutions influence to the accuracy in content based model transformation.
Keywords: Cyber Security Modeling Language; Model Transformation; Model Driven Engineering.

1 Introduction

Model-Driven Engineering (MDE) [1] uses models as a reference in various phases of software engineering. The model is created in the early stages and reused later for a number of purposes. Since most of the processes and aspects can be formalized and represented as a model - they are commonly used for their commodity. To obtain a certain output from different type of models is vital for MDE and a variety of solutions has been proposed by the research community, spanning from experimental approaches [2] to frameworks [3]. Model transformation is a very actual problem in practice as well as research as new types of models appear and more accuracy is needed.

The aim of this paper is to simplify transformation of abstract, content based model transformations. Content based models have very abstract structure. It can be a benefit as it increases the meta-model adaptation area, but one of the main drawbacks is that model transformations have to be done in content rather than structure level. Two main problems with content based model transformations are presented in this paper along with the solutions. To analyze the effectiveness proposed solutions, they are integrated into a tool for OPC/XML drawing file to CySeMoL model transformation. The accuracy results of the transformation are presented in this paper as well.

2 Related Works

Numerous research approaches have been carried out on model transformations, as it is a very useful process that not only leads to automation of processes [1], ease of migrating data [2] and at the same time liberating the systems from legacy components [3], but also, most importantly, from the economic point of view - reducing costs by reusing the existing data [4]. Methodologies have been developed to manage the correctness of data, stored as model attributes in the process of transformation. Of which, the triple graph grammar case offers a methodology for attribute handling for bidirectional model transformations [5].

Dedicated model transformations for information security modeling is a relatively new yet very important area for research. Model-driven security is a growing trend with an expanding list of tools and methodologies for the subject [6]. Approaches, such as SecureUML model transformation semantics and analysis [7] as well as transformations between SecureUML and UMLsec [8] exist. However, new information security assessment tools require a more flexible approach with an ability to acquire data from less formalized model structures as information security modeling typically involves representing the analyzed infrastructure in a formal way. Architectural modeling languages are typically used in this case. They include SysSML [9], Business Process Modeling Notation (BPMN) [10] that enable representation of information system architecture and system environment through diagrams that can be used for various forms of analysis, one of which is security. Some of them also offer extensions for Industrial Control System Security Analysis [11]. However, the aforementioned modeling languages do not offer the reasoning process. Some solutions that offer modeling capabilities along with the reasoning based on the systemized expert knowledge base exist. One of them is OpenMADS [12], the other is Cyber Security Modeling Language (CySeMoL) [13].

2.1 Model to Model Transformations

Model transformation enables information reuse preserving consistency between the two models [14]. In this case preservation of relationship between the source and target models as well as heterogeneity of the transformed data comes as a challenge [15]. Model transformation is facing two issues: impedance mismatch and heterogeneity [16]. Heterogeneity forces to deal with different data models and encodings of values. Impedance mismatches are caused by the difference between logical schemas required by the applications and the ones exposed by data sources. These issues support the idea that data consistency between the models by adjusting the level of abstraction is the main task in order to avoid data loss along the transformation process [17].

Model transformation patterns are obtained by using the Formal Concept Analysis [18], where relations and element meta-classes of target and source models are linked together based on model classification group links that have similarities between them.

2.2 Content Dependent Model Transformations

Some languages are equipped with an abstract meta-model. The content of the model is provided in text based form as the label or property value of an element (see Fig. 1). This type of meta-model is very common in general purpose systems. The abstract meta-model allows presentation of wider, not predefined content.

According to Taxonomy of Model Transformations [19] this type of transformation is considered to be exogenous, vertical transformation. Typically it is used as synthesis of a higher-level, more abstract, specification into a lower-level, more specific one.

Transformation of such model is very content dependent. Therefore the definition of transformation rules is time consuming due to these reasons:

- Every model component and property label has to be listed in order to write a transformation rule. As labels are human generated, the list is infinite or very long as all components and properties can have multiple synonyms.

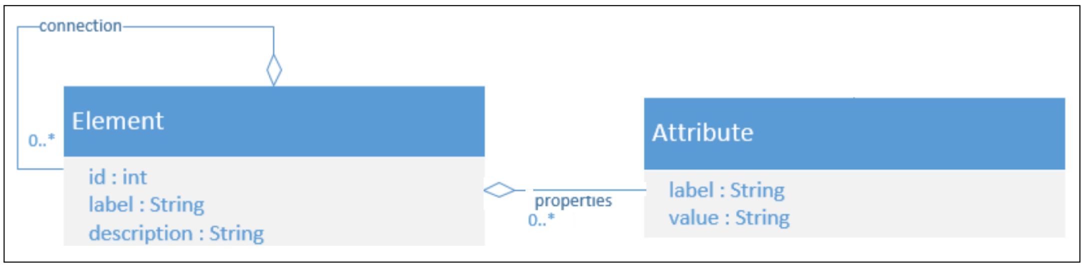

Figure 1: Example of general purpose source meta-model

- All component synonyms have to be taken into the account for the transformation rules. Therefore multiple rules are required for one target concept or rule.

These reasons cause higher resource consumption compared to discrete formal models. There is also a level of uncertainty, as some of the synonyms or concepts can be missed out of the model transformation rules and the process will not be able to transform the elements into the target model. An example of a content dependent source model definition is presented in Fig. 2.

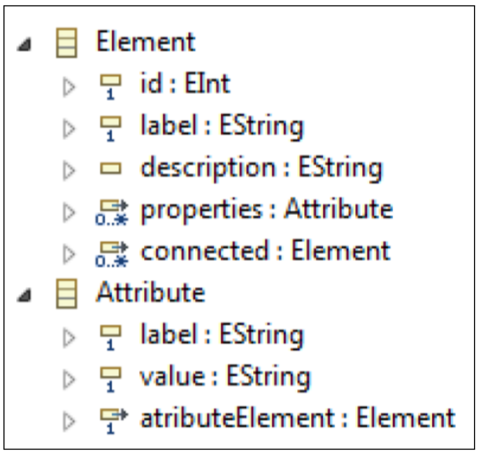

Figure 2: Example of content dependent source model definition in ECore file

An example of transformation rules of such model in ATL is presented in Fig. 3.

The provided example in figure 3 only has five synonyms, however, the list of synonyms increases by taking different languages, dialects and situations into account. Therefore, it would be difficult to modify the list of synonyms if the rule is hardcoded into the source code of software product. A solution for easy synonym integration is a valuable improvement.

The source model element type identification in content dependent models complicates when abstract element does not have a name nor a description. In such situation the information is not enough. Therefore element identification can be performed according to the structure of the element. However this task in content dependent models is complicated as well as there is no predefined specific element structure for different content source elements. Therefore, to identify the type of source model, rules can to be used to check if the containing attributes match the ones expected in the target model (see. Fig. 4).

Since the source model is abstract, the transformation is facing some complications as well:

- The attribute set for each element has to be defined individually as there is no list of attribute labels and values in the meta-model. The complications are amplified if the attribute labels are hardcoded in the software source code.

```
rule Element2Network {
  from  //define which component to take from the source
    s:Abstract!Element in IN (s.label = 'Network' or
      s.label = 'Net' or s.label = 'Internet' or
      s.label = 'LAN' or s.label = 'WAN')
    to
      //define how the source element have to be transformed
        n1:Cysemol!NetworkZone(  //creating NetworkZone element
          id <- s.id,       //with appropriate properties
          name <- s.label,
          originalConnection <- s.connected.id,
          interface <- n2
        ), //creating NetworkInterface element to connect NetworkZone
       n2:Cysemol!NetworkInterface(
          network <- n1
        )
}
```

Figure 3: Example of content dependent element transformation rule in ATL for element, associated to NetworkZone in target model

```
rule Element2Computer{
  from //searches for elements with needed attributes
    s:Abstract!Element in IN (
          s.attribute->collect(l | l.label)->
          includesAll(Set{'cpu', 'ram', 'hdd'})
        )
  to
    n1:Cysemol!OperatingSystem(
          name <- 'Computer with '+s.label
        )
}
```

Figure 4: Example of content dependent element transformation according to the obtained parameters rule in ATL

- A decision has to be made on which of the attributes provide a better definition of the element, at the same time which ones are unimportant and may be discarded. In case too many attributes are compared in the transformation rule, a missed attribute in the source model would make the rule worthless. On the contrary, if not enough attributes are be used in the transformation rule, element can be inconclusively (multiple possibilities) identified.
- Attribute labels and values are content based. Therefore multiple labels and values can be linked to the same content. Knowing all possible values is nearly impossible and it increases the complexity of transformation rules.
- Source element identification according to its structure element label, attribute labels and attribute values can be the crucial element. There is no unified methodology for measurement of the significance of the element identity from list of possible cases.

All these reasons make the source model element difficult to identify using only static rules.

3 Assumptions for Model to Model Transformation Improvement

Existing model transformation methodologies seem to have drawbacks when dealing with specific situations or have to be applied in dynamic situations [20]. Therefore new solutions are proposed to improve the process and provide an alternative method that improves the efficiency and accuracy of the transformations. In this chapter ideas on how current situation in specific situations can be improved using advanced techniques, such as grammar-based model transformations [21] and model transformation by-example [22–25] element identification are presented.

3.1 Dictionary Based Element Identification

A context analysis is a compex task as some words can have different meaning, synonyms for most of words exists etc. One of ways to implement context analysis is synonym based analysis. This aproach is used in web serach engine optimization [26] and user review analysis [27] during the last two years and shows promising results. Therefore dictionary based element identification approach on model to model transformation is proposed for simplification of the transformation of content dependent models. The main idea is to use a synonym database for each target meta-model element. This is done by providing additional dictionary meta-model (see Fig. 5) and input of synonyms for each of the target model elements.

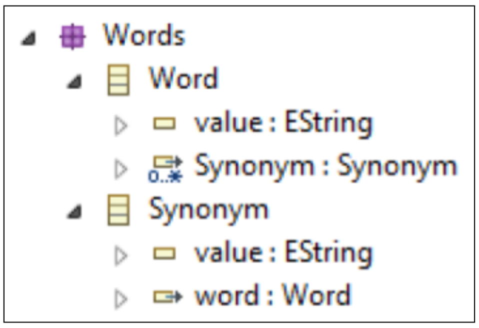
```
Words
  Word
    value : EString
    Synonym : Synonym
  Synonym
    value : EString
    word : Word
```

Figure 5: Dictionary meta-model for element identification

The condition for element identification in source model is simplified and achieved using only one condition rather than a list of conditions. An example of synonym search in dictionary model and its usage are provided in Fig. 6 and Fig. 7.

```
helper def : getBySyninym(sn : String) : String =
  if ( //looking for a synonym in the dictionary model
    Dictionary!Synonym.allInstances()->
       select(e | e.value.toLowerCase().startsWith(sn.toLowerCase()))
  ).isEmpty()
  then //if there is no synonym - we take the same value
    '<<'+sn+'>>'
  else //if there is a synonym - we take the word
    Distionary!Synonym.allInstances()->
       select(e | e.value.toLowerCase() = sn.toLowerCase())->
       collect(e | e).first().word.value
  endif;
```

Figure 6: Example code for search of an element name by comparing it to existing synonyms

```
...
  from
    s:Abstract!Element in IN (
         //helper usage to get synonyms from the dictionary
         thisModule.getBySynonym(s.label) = 'network'
       )
...
```

Figure 7: Simplified situation of Fig. 6 used to identify element type of source model

The proposed solution is more flexible as the list of synonyms for target model elements can be provided as input file and modified at any given moment. These changes do not require source code to be changed, so the dictionary file can vary depending on the target metamodel, language and other factors.

3.2 Example Based Element Identification

Example based model transformation is well known strategy for transform one model to another and other tasks. This technology is used for images and videos color transformation [28], semantic data analysis from a give string [29] etc. Therefore we propose to use example based model transformation in order to simplify the transformation of content dependent models where elements are defined by structure only. In this case a database of target meta-model element examples is used. For each target meta-model element one example is stored in the database of source meta-model. To simplify the ATL code a new meta-model was created as a copy of source meta-model (see Fig. 8).

When example target elements of source meta-model are presented, each source element is compared to the one stored in the example database to find the most similar element based on its structure. Similarity estimation q is calculated using as follows:

$$q = \frac{m}{s} + \frac{m}{e} \tag{1}$$

In (1) m is the number of matched labels between source and example elements; s is the number of attributes in the source element; e is the number of attributes in the example element.

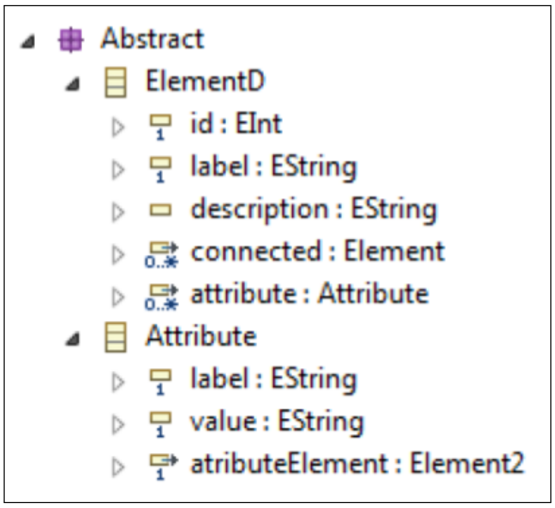
Abstract
ElementD
id : EInt
label : EString
description : EString
connected : Element
attribute : Attribute
Attribute
label : EString
value : EString
atributeElement : Element2

Figure 8: Modified source meta-model structure for target element descriptionl

Sum of these two proportions takes both attribute redundancy and shortage into account. The implementation of this method is presented in Fig. 9.

There is some space for the improvement of this example by optimizing the code, adding the comparison and attribute labels. Example based element identification should be executed after dictionary based element and attribute transformation as the labels and values of source model are content dependent and dictionary usage leads to formalization.

4 Case Analysis: OPC /XML drawing file transformation to CySeMoL

OPC is a container file standardized format [30]. An OPC format for storing graphical notation has an extension of .vsdx. The structure of the OPC/XML drawing file is presented in Fig. 10. The information about the element layout of the pages are stored in separate XML format files in sub-directory "visio/pages" (marked red in Fig. 10). In this case object and relationship information is extracted from files stored in this directory.

For this model transformation specific tags of the XML files are used. They are:

- Shapes - describes a shape array;
- Shape - describes a shape and its' identification number, name, type and master template;
- Cell - it is a versatile tag, containing information about name and value of many properties of cells under Shape and Section tags;
- Text - gives text output, most commonly an object of instance, visible graphically;
- Section - contains attribute information under it;
- Row - stores attribute information;
- Connects - describes array of connections;
- Connect - defines a connector between instances, specifying sheets, cells and parts connected.

```
helper def : calculateValue(
  a:Integer, b:Integer, c:Integer) :
  //calculates the similarity value q
  Integer = ((m/a + m/b)*100).floor();

helper def : calculateSimilarity(
  a:Abstract!Element, b:Abstract2!ElementD) :
  Integer = thisModule.calculateValue(
    //calculate a value (number of attributes in source)
    a.attribute->collect(l | l.label)->size(),
        //calculate b value (number of attributes in target)
    b.attribute->collect(l | l.label)->size(),
        //calculate m value (number of maching attributes)
    ((a.attribute->collect(l | l.label).asSet().
      intersection(b.attribute->collect(e | e.label).asSet()))->
    size())
  );

helper context Abstract!Element def : getByStructure() :
  String = let sk : String = self.getByExample2() in
    //skipping first letters, which indicates similarity
        //as the most similar element label is presented at the end
    sk.substring(5, sk->size());

helper context Abstract!Element def : getByExample2() :
  String = let elem : Sequence(Abstract2!ElementD) =
    Abstract2!ElementD.allInstances()->asSequence() in
    elem->iterate(p; label : String = '000' |
          //looking for the maximum q value
          if thisModule.calculateSimilarity(self, p) >
            label.substring(1, 3).toInteger()
          then let numb : Integer =
            //returning 3 digit value and label of the element
            thisModule.calculateSimilarity(self, p) in
        ('000'+numb).substring(('000'+numb)->size()-2,
        ('000'+numb)->size())+' '+p.label
      else label //otherwise returning the same value
      endif
        );
```

Figure 9: Example code for search of element name by comparing it to example structure

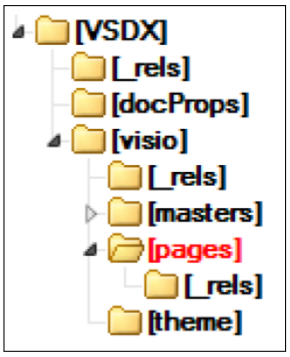

Figure 10: The structure of an OPC /XML drawing file

The proposed transformations were implemented as a tool to convert .vsdx file data into CySeMoL meta-model. The proposed transformation methods use synonym database for target meta-model. The current version is constructed for English language only. It stores over 6000 synonyms for most common CySeMoL classes and attributes. An additional integrated database for connection comparison is built as well. This database has up to 1000 possible connections between CySeMoL elements and serves as an alternative to the example based content dependent model transformation. The ideas of model transformations based on triple graph grammars are integrated [31] as well as class identification using missing elements based on the target model connection example database. This allowed a more accurate class mapping.

4.1 Transformation Accuracy Estimation Experiment

An experiment has been carried out to estimate the accuracy of the proposed model transformation methods. This experiment includes estimation of the results provided by a group of 48 Informatics Engineering senior year students. They were assigned to draw two diagrams in Microsoft Visio 2013 tool: one to present basic SMEs local network and one - basic web server diagram. The diagram type, diagram elements, description, and detailing level were entirely a matter of choice. The only constraint was to use English language exclusively. The experiment resulted in 86 different diagrams. The most common examples are presented in Fig. 11 and Fig. 12.

All provided diagrams were transformed to a CySeMoL model. The transformed models were analyzed and compared to expert prepared CySeMoL model in the EAAT tool. The EAAT tool allows graphical representation of cybersecurity area as well conforms to the model requirements for CySeMoL. Automated formal comparison as the results were not compliant to any formalization. Therefore multiple output results were generated. This fact required to analyze every situation individually by experts.

During the experiment most CySeMoL models had more elements in comparison to the source model file data. This is due to some additional elements had to be added as interfaces (see Fig. 13 and Fig. 14 as results of Fig. 11 and Fig. 12 in CySeMoL).

4.2 Results of Transformation Accuracy Estimation Experiment

The network and Web server diagrams use different Microsoft Visio diagram templates and elements, therefore they are analyzed separately. Diagram description level categorization was to the following categories: no diagram element descriptions; defined diagram element name; defined associated diagram element properties. These categories are used for assessment of usefulness of diagram name and property descriptions.

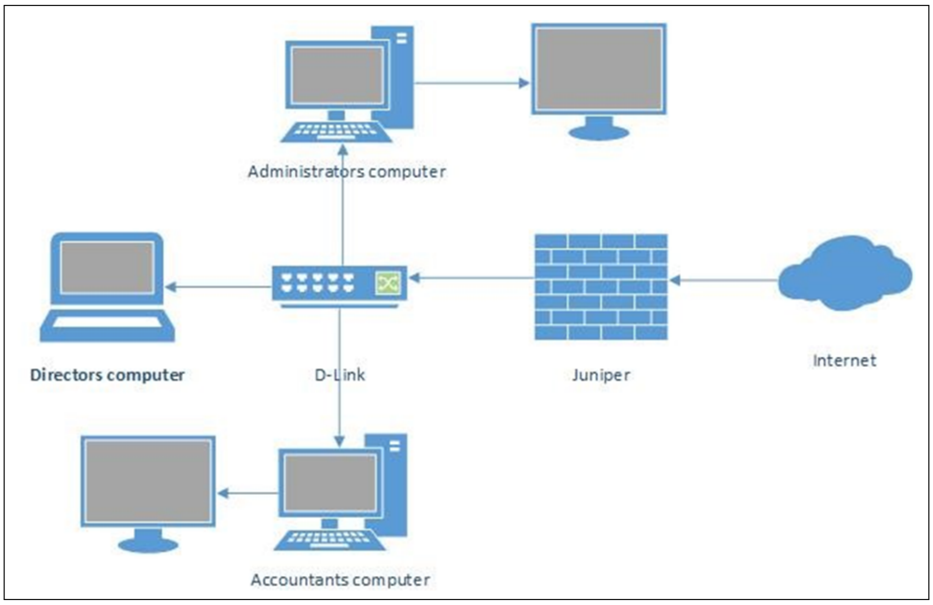

Figure 11: Typical result diagram for basic SMEs network

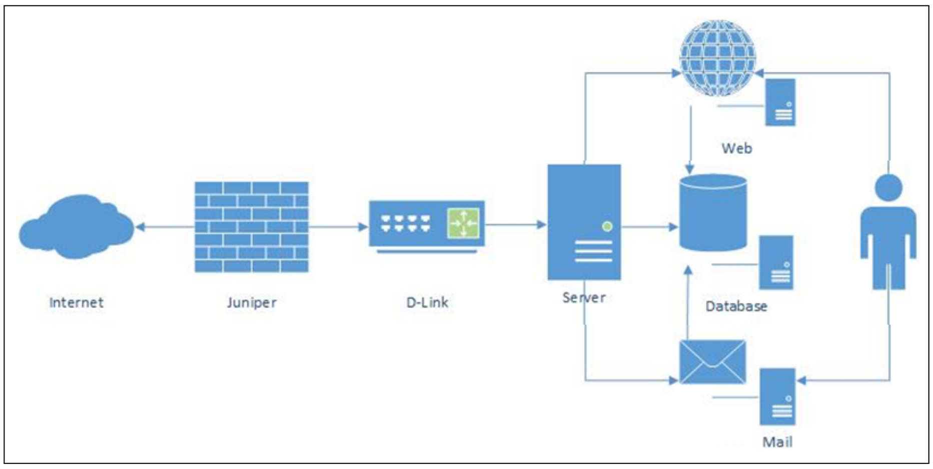

Figure 12: Typical result diagram for basic web server

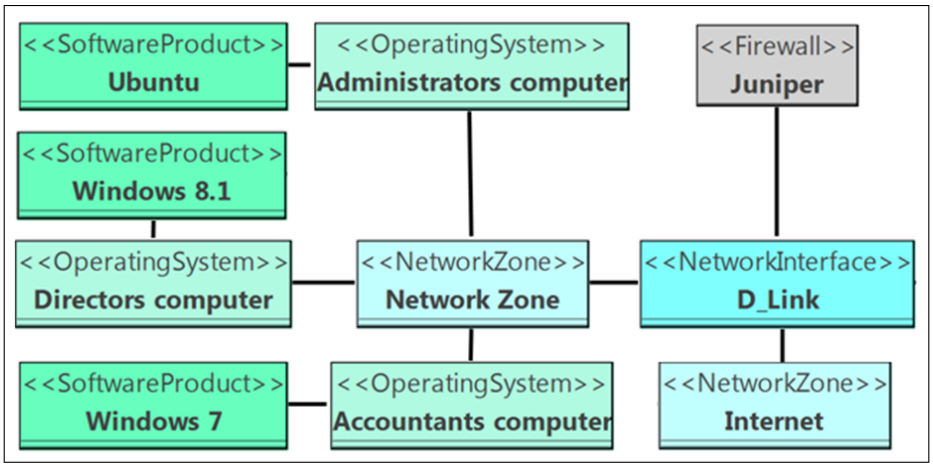
<<SoftwareProduct>> Ubuntu
<<OperatingSystem>> Administrators computer
<<Firewall>> Juniper
<<SoftwareProduct>> Windows 8.1
<<OperatingSystem>> Directors computer
<<NetworkZone>> Network Zone
<<NetworkInterface>> D_Link
<<SoftwareProduct>> Windows 7
<<OperatingSystem>> Accountants computer
<<NetworkZone>> Internet

Figure 13: Typical result diagram for basic SMEs network in CySeMoL

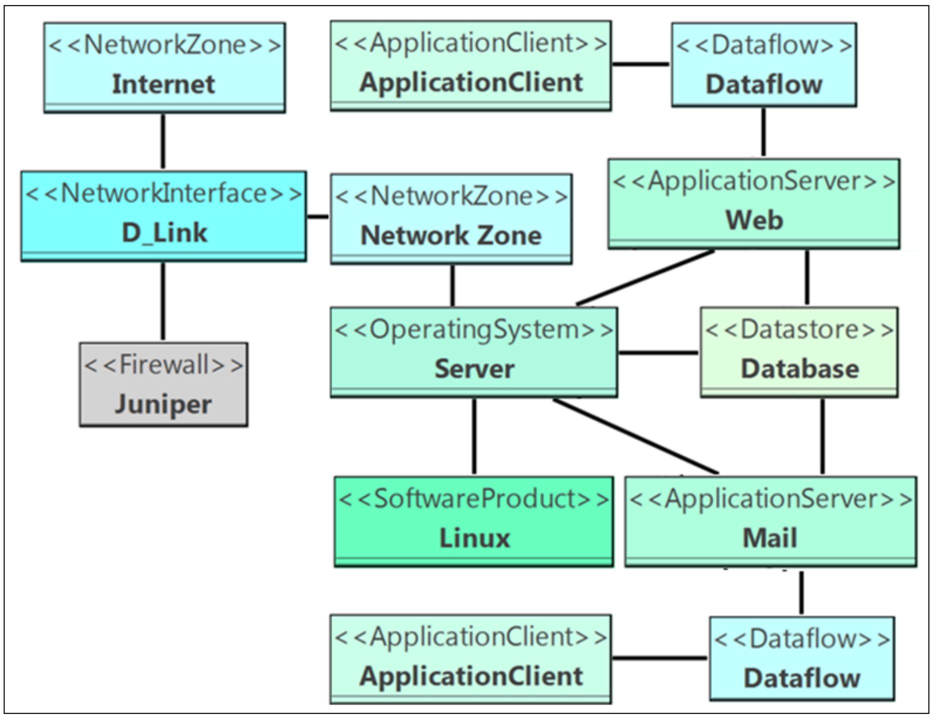

Figure 14: Typical result diagram for basic web server

The summary of analyzed file data and transformation accuracy for different diagram type and description level is presented in Tables 1. The Table 1 shows the number of property level detailed SMEs network OPC/XML files are bigger comparing to Web server diagrams as well as property level detailed diagrams usually have less components, comparing to less detailed Microsoft Visio diagrams. The most important - the generated CySeMoL model has a bigger number of elements, comparing to the source files as existing CySeMoL meta-model requires more elements to evaluate the risk.

Transformation accuracy analysis shows that the proposed transformation method is capable of generating CySeMoL models from more abstract OPC/XML drawing files - 94% of generated CySeMoL model elements are correctly identified; 88% of CySeMoL objects are transformed to template level (with defined default values); 87% of connections between CySeMoL objects are added as expected (see Table 1).

As seen in table 1 the accuracy is dependent on used diagram content and detailing level. OPC/XML drawing files have a predefined attribute list, however not all diagram elements are covered. Therefore some elements cannot be detailed by defining their attribute values. Moreover, model transformation might fail due to incorrect diagram element description, using modified terms. This requires maintenance of synonym database, keeping it up to date with human language and technology improvement changes.

Table 1: Summary of OPC/XML files and CySeMoL model data and transformation accuracy

	SME networm situation			WEB server situation			**Total**
	Components and links with no descriptions	Component name added with no properties	Component name and properties added	Components and links with no descriptions	Component name added with no properties	Component name and properties added	
Number of files	6	18	22	21	15	4	**86**
Number of .vsdx elements	42	121	145	169	127	22	**626**
Number of CySeMoL elements	130	349	435	569	412	105	**1976**
objects	62	184	231	268	193	50	**988**
connections	68	165	204	301	219	58	**1015**
Correctly identified element %	95%	100%	100%	96%	100%	100%	**98%**
Correctly identified connection %	81%	98%	100%	64%	98%	97%	**87%**
Total	**88%**	**99%**	**100%**	**79%**	**99%**	**98%**	**94%**

5 Conclusions

The proposed model transformation methods offer a semi-automatic abstract relationship-based model transformation into more detailed, domain specific template-based model. As this is a content dependent situation - detailed knowledge databases are required to extract knowledge and identify model elements according to text based names and descriptions.

Textual dictionary based analysis is used for element identification, however further reasoning is required for definition of the source model element relation to destination metamodel. Element identification in source model is one of the most important steps in model to model transformation. The combination of dictionary association, structure comparison and relationship similarities provided a 94% accuracy in this model transformation. For further improvements it requires a detailed list of attributes in order to increase the model transformation accuracy.

Bibliography

[1] L. Levi, M. Amrani, J. Dingel, L. Lambers, R. Salay, G. M. K. Selim, E. Syriani and M. Wimmer (2014), Model Transformation Intents and their Properties, *Software Systems & Modeling*, 1-38.

[2] L. M. Rose, M. Hermannsdoerfer, S. Mazanek, P. V. Gorp, S. Buchwald, T. Horn and E. Kalnina (2014), Graph and Model Transformation Tools for Model Migration, *Software & Systems Modeling*, 13(1): 323-359.

[3] G. M. K. Selim, S. Wang, J. R. Cordy and J. Dingel (2012), Model Transformations for Migrating Legacy Models: An Industrial Case Study, *ECMFA, LNCS 7349*, 90-101.

[4] S. Sen, N. Moha, V. Mahe, O. Barais, B. Baudry, J. M. Jezequel (2012), Reusable Model Transformations, *Software & Systems Modeling*, 11(1): 111-125.

[5] L. Lambers, S. Hildebrandt, H. Giese and F. Orejas (2012), Attribute Handling for Bidirectional Model Transformations: the Triple Graph Grammar Case, *Electronic Communications of the EASST*, 49: 1-16.

[6] D. Basin, M. Clavel and M. Egea (2011), A Decade of Model-driven Security, *SACMAT11 Proceedings of the 16th ACM symposium on Access control models and technologies*, 1-16.

[7] A. D. Brucker, J. Doser and B. Wolff (2006), A Model Transformation Semantics and Analysis Methodology for SecureUML, *Lecture notes in computer science*, Berlin, Springer, 306-320.

[8] R. Matulevicius and M. Dumas (2011), Towards Model Transformation Between SecureUML and UMLsec for Role-based Access Control, *Databases and Information Systems*, 6: 1-14.

[9] S. Friedenthal, A. Moore and R. Steiner (2014), *A Practical Guide to SysML*, Waltham: Elsevier, 2014.

[10] M. Chinosi and A. Trombetta (2012), BPMN: An introduction to the standard, *Computer Standards & Interfaces*, 34: 124-134.

[11] L. Lemaire and J. Lapon (2014), A SysML Extension for Security Analysis of Industrial Control Systems, *2nd International Symposium for ICS & SCADA Cyber Security Research 2014 (ICS-CSR 2014)*,1-9.

[12] E. C. Andrade, M. Alves, R. Matos, B. Silva and P. Maciel (2013), OpenMADS: An Open Source Tool for Modeling and Analysis of Distributed Systems, *Computer Safety, Reliability, and Security*, Lecture Notes in Computer Science, 8153: 277-284.

[13] T. Sommestad, M. Ekstedt and H. Holm (2013), The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures, *Systems Journal*, 7: 363-373.

[14] M. Biehl (2010), *Literature study on model transformations*, Royal Institute of Technology, Stockholm, 2010.

[15] K. Czarnecki and S. Helsen (2003), Classification of model transformation approaches, *2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, 2003.

[16] P. A. Bernstein and S. Melnik (2007), Model Management 2.0: Manipulating Richer Mappings, *SIGMOD07, Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, 1-12.

[17] T. Frisendal (2012), Business Concept Mapping, *Concept Maps: Theory, Methodology, Technology Proc. of the Fifth Int. Conference on Concept Mapping*, 1-4.

[18] B. Ganter and R. Wille (2012), *Formal Concept Analysis: Mathematical Foundations*, Berlin: Springer Science & Business Media, 2012.

[19] T. Mens and P. V. Gorp (2015), A Taxonomy of Model Transformation, *Electronic Notes in Theoretical Computer Science*, 152: 125-142.

[20] K. Czarnecki and S. Helsen (2006), Feature-Based Survey of Model Transformation Approaches, *IBM Syst. J.*, 45(3): 621-645.

[21] G. Besova, D. Steenken, and H. Wehrheim (2015), Grammar-based model transformations, *Comput. Lang. Syst. Struct.*, 43(C): 116-138.

[22] H. Saada, X. Dolques, M. Huchard, C. Nebut and H. Sahraoui (2012), Generation of operational transformation rules from examples of model transformations, *Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science*, 7590: 546-561.

[23] M. Wimmer, M. Strommer, H. Kargl and G Kramler (2007), Towards Model Transformation Generation By-Example, *HICSS 2007, 40th Annual Hawaii International Conference on System Sciences*, 285b.

[24] G. Kappel, P. Langer, W. Retschitzegger, W. Schwinge and M. Wimmer (2012), Model Transformation By-Example: A Survey of the First Wave, *Conceptual Modelling and Its Theoretical Foundations*, 197-215.

[25] D. Varro (2006), Model Transformation by Example, *Model Driven Engineering Languages and Systems*, 410-424.

[26] P. Arora and T. Bhalla (2014), A Synonym Based Approach of Data Mining in Search Engine Optimization, *International Journal of Computer Trends and Technology (IJCTT)*, 12(4): 201-205.

[27] B. Ma, Z. Dongsong, Z. Yan and T. Kim (2013), An LDA and Synonym Lexicon Based Approach to Product Feature Extraction from Online Consumer Product Review, *Journal of Electronic Commerce Research*, 14(4): 304-314

[28] Y. Chang, S. Saito and M. Nakajima (2005), Example-based color transformation for image and video, *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE05). ACM*, New York, NY, USA, 347-353.

[29] R. Singh and S. Gulwani (2012), Learning semantic string transformations from examples, *Proc. VLDB Endow.*, 5(87): 740-751.

[30] International Organization for Standartization (2012), ISO 29500-2:2012 Information technology - Document description and processing languages - Office Open XML File Formats - Part 2: Open Packaging Conventions. Third Edition, Geneva: International Organization for Standartization.

[31] F. Hermann, H. Ehrig, F. Orejas and U. Golas (2010), Formal Analysis of Functional Behaviour for Model Transformations Based on Triple Graph Grammars, *Graph Transformations*, Berlin, Springer, 155-170.