Interoperability of Software Engineering Metamodel: Lessons Learned

Muhammad Atif Qureshi

School of Software, Faculty of Engineering and IT, University of Technology, Sydney, Australia

Abstract. Use of models and modelling languages in software engineering is very common nowadays. To formalize these modelling languages, many metamodels have been proposed in the software engineering literature as well as by standard organizations. Interoperability of these metamodels has emerged as a key concern for their practical usage. We have developed a framework for facilitating metamodel interoperability based on schema matching and ontology matching techniques. In this paper we discuss not the techniques used but rather we focus on the lessons we have learned by applying the framework on several pairs of metamodels for finding similarities between them. We have highlighted some areas where these techniques can be beneficial and also pointed out some limitations of these techniques in this domain.

1 Problem Description and Motivation

Many metamodels have been proposed in different domains of software engineering such as process [1], product [2], metrics [3] and programming [4]. Most of these metamodels have been developed independently of each other with shared concepts being only accidental. These metamodels are evolving continuously and many versions of these metamodels have been introduced over the years. This evolution has extended not only the scope but their size [5] and complexity as well. The need to formulate a way in which these metamodels can be used in an interoperable fashion has emerged as a key issue in the practical usage of these metamodels. There are several benefits of such interoperability including: reduced joint complexity, ease of understanding and use for newcomers, portability of models across modelling tools and better communication between researchers [6]. This overall need is also emphasized by the software engineering community [7] and further endorsed by the rise of industry interest as well as various conferences and workshops on the topic [8]. To have interoperability between any pair of metamodels, similarities between the elements of metamodels need to be identified. This is undertaken by a matching technique as yet little utilized for metamodels although widely used in ontology engineering. Close similarity between metamodels and ontologies [7],[9],[10] suggests that it should be efficacious to adopt ontology matching techniques for facilitating meta-model interoperability with a first step of linguistic matching. Indeed, ontologies are also helpful in reducing semantic ambiguity [9], helping not only to improve the

semantics of a metamodel [10] but also providing a potential way in which these meta-models can be bridged with each other to be interoperable. A framework [11] for facilitating interoperability of metamodels has been developed based on the ontology merging and schema matching techniques. The frame-work was applied to several pairs of metamodels including OSM [12], BPMN [13], SPEM [1] and some multi agent systems (MAS) metamodels. In this paper we discuss the lessons learned by applying the framework on these metamodels. We have highlighted the areas of metamodel interoperability that can be assisted by using these techniques as well as discussing some of their limitations. In Section 2 we briefly present our framework for metamodel interoperability. Section 3 presents the lessons learned during the application of this framework to several meta-models, followed by a conclusion and summary of likely future work (Section 4).

2 Proposed Solution

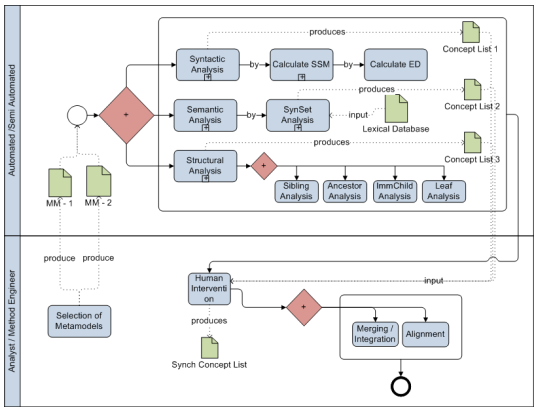

Fig. 1. Metamodel Interoperability Framework [11]

The framework for metamodel interoperability is depicted in Fig. 1 as a BPMN diagram. The framework has two major activities: Linguistic Analysis and Ontological Analysis. These are further divided into subactivities, as represented in the digram. While trying to make metamodels interoperable using this framework, we assume that there exists some commonality between a pair of metamodels. It is necessary to identify the potential common concepts (conceptual elements) that can be shared between two metamodels. The detailed discussion on this framework is not our focus in this paper but can be found in [11]. The overall similarity of any pair of elements is based on the three different types of similarities among them: syntactic, semantic and structural. In applying the framework to a variety of metamodels, several thousand different

permutations were computed for the comparison of the metamodel elements. The following sections elaborate our experience of using this framework and discuss the lessons we have learned during the experiment.

3 Lessons Learned: Limitations and Opportunities

3.1 Syntactic Matching

Opportunities: Syntactic matching between a pair of metamodels is based on a comparison between the names of the conceptual elements within those metamodels. Different techniques in the literature are available that can be used for such comparison. One such technique is known as string-edit distance of simply edit distance (ED) [14], which counts the number of token insertions, deletions and substitutions needed to transform one lexical string S1 to another S2, viewing each string as a list of tokens. For example the value of ED for two strings Brian and Brain is 2. Various other techniques for string comparison are used in different domains e.g. N-gram, Morphological Analysis (Stemming), and Stop-Word Elimination. ED can be used then to calculate the syntactic similarity (SSM) between a pair of elements [15]. Lessons Learned: These techniques can be useful in comparing the elements with-in the same domain e.g. domain ontologies; where elements with the same name have (most of the times) the same meaning. The problem with these techniques in the context of metamodels is that they are not effective when applied standalone. Our experience with metamodel matching shows that considering only syntactic similarity measures, isolated from their semantics, creates misunderstanding by expressing the same meanings in different terms. For example confirmation and notification has approximately 60

3.2 Matching the Semantics

Metamodels are generally treated as a model of a modelling language [16], [17],[18][19]. These modelling languages are designed (mostly) for specific domains. Therefore, we believe that to compare the semantic similarity of metamodel elements, it is important to consider both perspectives: linguistic and ontological. The linguistic semantics involves checking the semantics of the metamodel elements from that modelling languages perspective e.g. their properties (attributes), types of attributes and to some extent their behaviour as well. On the other hand, on-tological semantics means finding the elements that have the same meaning but may have been presented with different names. Opportunities: Techniques for comparing class diagrams e.g. [20],[21] can be utilized to find the similarities between metamodel elements, especially for the metamodels that are represented using object-oriented classes (meta-classes) e.g. OMGs family of meta-models. Different approaches in the area of computational linguistics and natural language processing can be used to find ontological semantic similarity e.g. finding the synonyms of a given conceptual element of one metamodel and

looking for those synonyms in the second metamodel. Synonyms can be found using any lexical data-base e.g. a dictionary. WordNet [21] is one lexical database that can be used for finding synonyms and word senses. WordNet is a registered trademark for Princeton University and contains more than 118,000 word forms and 90,000 different word senses. Lessons Learned: We have observed that finding ontological semantic similarity is very important as there are so many such conceptual elements in metamodels presented with different names. For example, Person in OSM [12] can be semantically matched with the Human Performer in BPMN [13]; although both have low syntactic similarity. Beside synonyms, hyponyms (sub-name, e.g. sugar-maple and maple), hypernyms (supername, e.g. step and footstep) can also be used to find semantic relevant elements, but none of these are considered so far in any technique. Similarly, meronyms (part-name, e.g. ship and fleet) and holonyms (whole-name, e.g. face and eye) can also be useful to find these similarities. Another problem is how to combine both linguistic and ontological semantic similarity for a pair of conceptual elements. Which one of them is more important and how much weight should be assigned to each of them is still unaddressed.

3.3 Comparing the Structures

Besides their level of abstraction, a metamodel is treated as a (conceptual) model of a language [22]. For a good similarity comparison between any pair of conceptual models, not only their syntax and semantics but also their structure should be compared. Opportunities: Different techniques have been proposed in the literature for structural similarity of conceptual models. Some of these [22], [14] compare the structure of business process models, whilst others [23],[24] are for matching the structure of conceptual models based on graph theory. An alternative to a graph matching technique is the schema matching techniques [24], [25][26][27][28]. In this technique, the structural similarity of two conceptual elements C1 and C2 is calculated based on their structural neighbours - ancestors, siblings, immediateChilds, and leafs. These partial similarities are then calculated by mean values of all the neighbouring concepts. Lessons Learned: The techniques used to compare the structure of business process models (e.g. [22], [14]) cannot be generalized for metamodels as business process models are behavioural models while metamodels represents the structural aspect. Converting the conceptual models to graphs [23], [24] and then applying graph matching algorithms to find the structural similarity between them is not a trivial task. To apply such a graph matching technique, we have to be very careful in the conversion of a class diagram into a graph. True replacement of relationships among classes (e.g. association, generalization, aggregation, composition) into relationships among nodes of a graph (e.g. directed/undirected, weighted/unweighted) is not straightforward. Another barrier for the application of such techniques is that most of the metamodels in the software engineering literature are specified using diagrams, tables and textual explanation. Having a single class diagram for such a huge metamodel is not easy. Techniques based on the planar graph

theory like [24] are also not feasible for meta-models because of the basic principle of planar graphs (having no cross edges). Meta-models with a rich set of constructs (classes) like UML can easily violate this rule as it is very difficult to convert class diagrams of these metamodels to graphs without any cross edges. The complexity of these graph matching techniques, as also mentioned by some authors [14], is another barrier to their application in the domain of metamodels, hence making it difficult to apply in practice. Based on the experience of applying these techniques to metamodels, we recommend that we dont need to compare the leaves of any conceptual element in a meta-model. Comparing leaf classes of a given class (conceptual element) only results in low similarity. Also, we think that rather than comparing all the ancestors of a conceptual element, it is better to compare only parent classes of that element.

3.4 Automation

Considering the size and complexity of metamodels [5], it is very convenient to have tool support for matching the similarity of metamodels. Hence, our experience with the matching of metamodels shows that, beside partial tool support, complete automated metamodel matching is not possible. Opportunities: Automation in syntactic matching of metamodels elements can be achieved by implementing ED (Edit Distance) and SSM (Syntactic Similarity Measure) algorithms using available online calculators for ED and APIs. The ontological semantics of metamodel elements can be matched automatically using lexical databases like WordNet, MS Office Thesaurus and other APIs available. Lessons Learned: Complete automation for metamodel similarity matching, especially for structural similarity, requires well formed formal definitions of metamodels that can be used as an input for any automated tool. Unfortunately, besides XML definitions for some of the metamodels (OMG metamodels with XMI definitions), metamodels lacks a formal specification and are mostly specified using a combination of textual descriptions, tables and class diagrams. Another important barrier in the complete automation is that coefficients in the equations we used do not have any fixed values and have to have value assigned by the domain expert at the time of the matching. Also, the ontological semantic similarity analysis requires the experts intellectual input to decide whether two conceptual elements are equal or not.

3.5 Refactoring

Lessons Learned: Most of the metamodels have two orthogonal forms of conceptual elements: linguistic and ontological (as also highlighted by [17]). The former represent the language definition while the latter describe what concepts exist in a certain domain and what properties they have. These two types of elements are mingled with each other in most of the metamodels and there is no explicit boundary between them. An important consideration regarding metamodel matching is to separate these two types of elements; we call it refactoring.

Metamodels need to be first refactored before matching can occur. This refactoring is required to remove the conceptual elements in metamodels that are not related to the domain of interest. Rather, most of these elements are linguistic and are present in order to maintain (glue) the structure of metamodels. For example, Resource Parameter Binding and Parallel Gateway in BPMN [13] are the concepts that are related to the language definition of BPMN and are not worth matching with any other metamodel of the same domain since every metamodel has its own language definitional elements. Rather, it is better to match the conceptual elements that are related to the domain of interest e.g. matching Activity in BPMN [13] with Activity in SPEM [1], which are more related to the common domain of interest: Workflows and Processes.

3.6 Ontology Oriented Metamodels

Lessons Learned: Our experience of matching metamodels showed that there is a high heterogenity between the ontological elements of metamodels. However, it has been observed that a major reason for that heterogeneity is the lack of a common ontology or taxonomy. Much better results in interoperability of metamodels can be achieved if metamodels share some common ontology or taxonomy of the domain of interest; as also highlighted by [8]. The use of a common ontology for designing/redesigning metamodels can result in better interoperability. For example, the use of the UFO (Unified Foundation Ontology) to redesign UML [29]. Metamodels based on a common ontology will reduce the differences of similarity matchings, especially in syntactic and ontological semantics matching.

4 Conclusion

In this paper we have discussed some of the limitations and opportunities in the field of metamodel interoperability. These recommendations are based on the application of a framework that we have developed and applied on several metamodels to find their similarities. We have come to conclude that, for better similarity findings, not only the syntax but also the semantics and structure of metamodel elements should be matched. Metamodels needs to be refactored to separate out the ontological elements before matching for more pragmatic results. To avoid the problems of syntactic and semantic ambiguities between the elements, we recommend that metamodels should be based (or at least utilize) upon some common domain ontology. Also we have shown that complete automation of matching metamodel elements is not possible and does require substantial human intervention.

References

1. OMG: Software and systems process engineering meta-model specification (2008)
2. OMG: Unified modeling language (2009)

3. OMG: Architecture-driven modernization (adm): Software metrics meta-model (smm) (2009)
4. Azaiez, S., Huget, M.P., Oquendo, F.: An approach for multi-agent metamodelling. Multiagent and Grid Systems **2**(4) (2006) 435–454
5. Henderson-Sellers, B., Qureshi, M.A., Gonzalez-Perez, C.: Towards an interoperable metamodel suite: Size assessment as one input. International Journal of Software and Informatics **6 (2)**(2) (2012)
6. Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J.J., Pavon, J., Gonzalez-Perez, C.: Faml: A generic metamodel for mas development. IEEE Trans. Softw. Eng. **35**(6) (2009) 841–863
7. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. Software and Systems **84**(2) (2011) in press
8. Bézivin, J., Soley, R.M., Vallecillo, A.: Proceedings of the first international workshop on model-driven interoperability (2010)
9. Tran, Q.N.N., Low, G.: Mobmas: A methodology for ontology-based multi-agent systems development. Inf. Software Technol **50**(7-8) (2008) 697–722
10. Devedzić, V.: Understanding ontological engineering. Communications of the ACM **45**(4) (2002) 136–144
11. Qureshi, M.A.: Interoperability of software engineering metamodels (2012)
12. OMG: Organization structure metamodel (osm) 3rd initial submission (2009)
13. OMG: Business process model and notation (bpmn) ftf beta 1 for version 2.0 (2009)
14. Dumas, M., Garca-Banuelos, L., Dijkman, R.: Similarity search of business process models. IEEE Data Eng. Bull **32**(3) (2009) 23–28
15. Maedche, S.: Comparing ontologies - similarity measures and a comparison study. Technical report, Institute AIFB, University of Karlsruhe, Internal Report (2001)
16. Henderson-Sellers, B., Gonzalez-Perez, C.: An investigation of the validity of strict metamodelling in software engineering. submitted to IEEE Trans. Software Eng. (2011)
17. Atkinson, C., Kuhne, t.: Model-driven development: A metamodeling foundation. IEEE Software **20**(5) (2003) 36–41
18. Gašević, D., Kaviani, N., Hatala, M. In: On Metamodeling in Megamodels. Volume 4735/2007. Speinger (2007) 91–105
19. Kuhne, T.: Matters of metamodelling. Software and System Modeling **5**(4) (2006) 395–401
20. Girschick, M.: Difference detection and visualization in uml class diagrams. technical report. Technical report (2006)
21. Miller, G.A.: Wordnet: A lexical database for english. Communications of the ACM **38**(11) (1995) 39–41
22. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models (2007) 1274465 71-80.
23. Voigt, K., Heinze, T.: Metamodel matching based on planar graph edit distance (2010) 1875866 245-259.
24. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching (2002 2002)
25. Bernstein, P.A.: Applying model management to classical meta data problems (2003)
26. Chukmol, U., Rifaieh, R., Benharkat, N.A.: Exsmal: Edi/xml semi-automatic schema matching algorithm (2005)

27. Filipe, J., Cordeiro, J., Sousa, J., Lopes, D., Claro, D.B., Abdelouahab, Z. In: A Step Forward in Semi-automatic Metamodel Matching: Algorithms and Tool. Volume 24 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 137–148
28. Lopes, D., Hammoudi, S., de Souza, J., Bontempo, A.: Metamodel matching: Experiments and comparison (Oct. 2006 2006)
29. Guizzardi, G., Wagner, G. In: Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. Springer-Verlag (2010)