Multi-paradigm Modelling via XSLT

Mohamed Ariff Ameedeen

Abstract—**Software development and modelling cannot be seperated in today's software life cycle. A different model is produced in every step of the way, starting from requirements all the way up to analysis. This creates a plethora of non-communicating, heterogeneous models. Multi-paradigm modelling promotes an interoperability between these models, extending the usability of the models and reducing the number of redundant models. This paper presents an alternative framework for Multi-paradigm modelling using XSLT to support various XML-based models used in software development.**

Index Terms—**Modelling, multi-paradigm, transformation, XSLT.**

I. Introduction

Modelling is becoming more and more common in today's software development, be it as a requirement specification model, a configuration model, an activity model or even a more formal model intended for analysis. These models are often created for specific stakeholders and do not communicate with one another creating an influx of heterogeneous models. This presents a hefty challenge to the software developers of today — needing to be well-versed in multiple modelling languages in order to be able to work with the various models involved in the process of developing the software.

Multi-paradigm modelling brings forward a platform of interoperability between the various models centered on model transformation. This interoperability creates a seamlessness for the developers where one type of model may be transformed into another, independent of the level of abstraction or level of formalism involved. One such example is the model transformation algorithm SD2PN [1] that creates interoperability between UML [2] Sequence Diagrams and Petri Nets [3], models with clearly differing levels of formalism.

Fig. 1 presents an example of Multi-paradigm modelling [4]-[11] where a model is designed in a semi-formal language, analysed in a formal language and the feedback from the model analysis is presented in natural language; three levels of formalism, working together seamlessly for the benefit of software developers.

One drawback of using model transformation as the basis of interoperability is the dependance on too many tools.

Manuscript received November 19, 2014; revised May 12, 2015. This work is supported in part by the Department of Higher Education, Ministry of Education, Malaysia under the Fundamental Research Grant Scheme (FRGS), RDU130116 through Universiti Malaysia Pahang.

M. A. Ameedeen is with the IBM Centre of Excellence, Universiti Malaysia Pahang, Malaysia, and the he is also with the Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Malaysia (e-mail: mohamedariff@ump.edu.my).

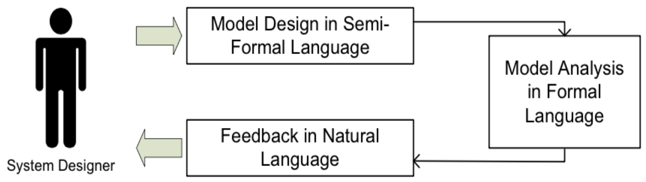
Model Design in Semi-Formal Language
Model Analysis in Formal Language
Feedback in Natural Language
System Designer

Fig. 1. Example of model design and model analysis via multi-paradigm modelling.

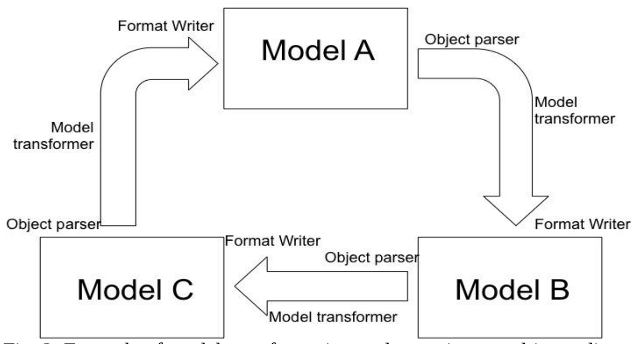

Fig. 2. Example of model transformation tools creating a multi-paradigm modelling framework.

Fig. 2 presents a Multi-paradigm modelling scenario with three types of models. Each model transformation in this scenario requires a set of three tools (an object parser to parse the source model into objects of the tools chosen programming language, a model transformer that thransforms one set of objects into another, and a format writer that writes the objects into the format specified by the destination model). In this scenario of three models, a total of nine tools are used for interoperability between there models.

This paper presents an alternative platform for Multi-paradigm modelling using a pre-existing framework; XSLT [12] or Extensible Stylesheet Language Transformations.

II. Foundation

In this section, preliminary information of the framework and technique used in this paper is provided to ease the readers' comprehension of the work.

A. *Multi-paradigm Modelling*

Multi-paradigm Modellingis a platform that promotes interoperability between heterogeneous models. Vangheluwe *et al.* [13] described Multi-paradigm modelling in modelling and simulation as a field that addresses three directions of research; multi-formalism modelling, model abstraction and metamodelling.

Multi-formalism Modelling. Multi-formalism modelling provides an interoperability platform for models with differing levels of formalisms on the basis of model transformation. Model transformation is the process of translating one model into another using a set of predetermined rules.

Currently, model transformation plays a key role in Model Driven Development (MDD) [14]. Based on a survey on model transformation [15], the intended application of model transformation include generating low-level models from higher level models, synchronizing models with different levels of formalisms and reverse engineering higher level models from low-level models. There are various frameworks available for model transformation, among others VIATRA (Visual Automated model Transformations) [16], [17], Kent Model Transformation Language [18], ATL [19], Kermeta [20] and SiTra [21], [22]. A common way to express a model transformation is using QVT relational language [23]. QVT is a standard for model transformation defined by Object Management Group (OMG).

A few key features that are common to all model transformation as described in [15] include specification, such as the pre and post conditions for a model transformation, the set of transformation rules, the directionality of the transformation as well as the source and target relationship. In an MDD model transformation, a source metamodel and a target metamodel are also required, whereby each source and target model should conform to the respective metamodels.

Model Abstraction. Model abstraction is the process of removing a certain low-level detail from the model while preserving the construct and general behaviour of the system. Similarly to multi-formalism modelling, model abstraction also uses model transformation. However, a significant difference between the two model transformations is that for model abstraction, the source and destination models are of the level of formalism.

Model abstraction is often used in removing various complicated low-level behaviours in the system, according the requirements of a specific perspective. For example, a complete model of the system filled with low-level behaviour might be too complicated for distribution to various stakeholders. However using model abstraction, the model could be simplified up to a certain level without losing its structural properties and vital behaviours. The same concept can also be used for optimization [9] of models. Using a base model that is filled with all the details, less detailed models can be automatically derived from it for various operation tasks such as control design and performance assessment.

Metamodelling. Metamodelling refers to the modelling of models. Metamodel or model of models is itself a model that defines other models. For example, suppose a modelling language L has a metamodel $\mathbb{MM}$L. As such, $\mathbb{MM}$L is a model that describes the constructs of the language L anLd every model that is written with the language L must be an instance of the metamodel $\mathbb{MM}$.

Mosterman and Vangheluwe [9] describe the advantages of metamodelling as numerous. The metamodel of a modelling language can be regarded as a specification for the language which can either be used for documentation purposes or as a basis for model analysis. Metamodelling also allows new languages to be born just by modifying or tweaking parts of existing metamodels. This allows customization of the modelling languages to serve a specific purpose.

B. SD2PN

SD2PN [1] is an MDD Model Transformation that performs transformation from Sequence Diagrams to Petri Nets.SD2PN uses a subset of the UML metamodel and a rule-based approach to transform Sequence Diagrams into a class of Petri Nets called Free Choice Petri Nets, an especially well-studied class of Petri Nets. The accuracy of the model transformation has also been established in [24] where LES was used as a common semantic domain between Sequence Diagrams and Petri Nets. Sequence Diagrams were mapped into LES using an algorithm obtained from [25] while Petri Nets were unfolded into LES using a technique from [26]. By comparing the LES, it was established that SD2PN preserves the semantics of the original Sequence Diagram throughout the transformation.

C. XSLT

XSLT or Extensible Stylesheet Language Transformations is itself a language that transforms one XML document into other XML documents, HTML documents or even plain text. In its infacy, XSLT was mostly used for interpretation of XML documents. However more recently, XSLT is used in transforming between different styles of XML documents and as a code generation language that could generate programming source code from multiple XML stylesheet documents.

III. XSLT Model Transformation

In this section, the ideology of using XSLT as the basis of Multi-paradigm modelling is presented with the aid of SD2PN as an example. With reference to Fig. 2 where interoperability between three different model types is shown to require nine separate tools, Fig. 3 presents a similar interoperability scenario with one stark difference; it does not require all those tools.

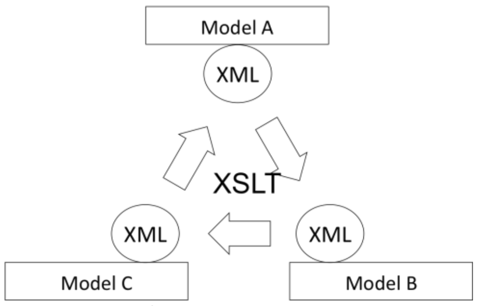

Fig. 3. Example of multi-paradigm modelling via XSLT.

Fig. 3 depicts three types of models that have XML representations as its basis. These XML documents could then be made interoperable through XSLT, where three XSLT specification could be implemeted in order to transform the stylesheet documents from one type to another.

This scenario, ideal as it may seem, depends on one fundamental requirement; the ability to represent said model in XML. Fortunately, the emergence and continuous evolvement of XML translates to more and more modelling languages adopting XML (or its equivalent) as its base language. The most widely accepted modelling language, UML uses XML Metadata Interchange (XMI) [27] as a format to represent its models. XMI is also adopted by

SysML [28] and is extensively used in various tools [29]-[31]. Other modelling languages such as Service Modelling Language (SML) [32], Business Process Modelling Language (BPML) [33], Educational Modelling Language (EML) [34], as well as formal languages such as Petri Nets (which uses a specilaization of XML called PNML) [35], Alloy [36] and also Common Logic [37] all use XML as its chosen format for representation.

To illustrate Multi-paradigm modelling via XSLT as presented in this paper, an example using SD2PN is presented.

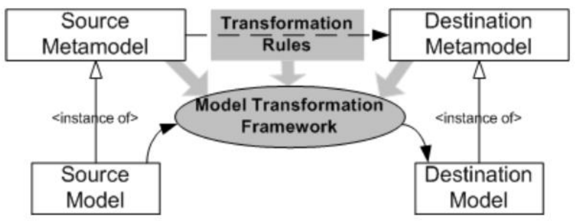

Fig. 4. Model driven development model transformation framework.

Fig. 4 depicts the framework that is used by SD2PN in its model transformation where the metamodels of Sequence Daigrams and Petri Nets are established, and a set of five transformation rules are presented together with two local functions in order transform all Sequence Diagrams into Petri Nets with the help of a Java tool.

The process of model transformation starts with parsing the XMI documents created with UML tools into Java objects. The tool then transform the Sequence Diagram Java objects into Petri Net Java objects based on the transformation rules. The Petri net Java objects are the writen into PNML documents using a specially designed format writer.

Alternatively as proposed in this paper, using XSLT minimizes the processes involved in the model transformation as presented in Fig. 5.

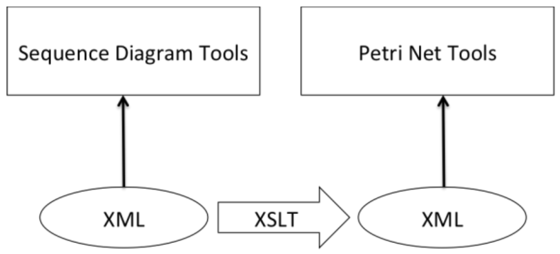

Fig. 5. SD2PN via XSLT.

Well established UML tools such as Rational Rose [29], Poseidon [30] as well as various Eclipse [31] based tools are all XMI ready. Sequence Diagrams created using such tools are saved in XMI format, and ready for transformation using XSLT. The XSLT is the defined to transform the XMI into PNML (a specialization of XML) which is a known standard for Petri Net tools such as CPNTools [38], ePNK [39] and various other tools.

This approach reduces the number of processes involved in the model transformation to a singular process as opposed to the three processes involved in the former approach. Any or all performance benefits that relates to this reduction of process is currently ignored since there are no formal performance analysisconducted as yet. Nonetheless, this approach reduces the possibility of errors in future transformations based solely on the lower number of process involved.

IV. Discussion and Conclusion

This paper has presented an alternative framework for Multi-paradigm modelling through XSLT as well as an example of XSLT based model operability using SD2PN. Neither the effectiveness of this alternative framework, nor the efficiency of it has been extensively studied in order to make a viable comparison with the existing method. This approach only provides an alternative framework for Multi-paradigm modelling between multiple models that uses XML as its base format. The effects and performance consequences of choosing this framework will have to be studied further before any recommendation could be made.

References

[1] M. A. Ameedeen and B. Bordbar, “A model driven approach to represent sequence diagrams as free choice petri nets,” in *Proc. 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC)*, München, Germany, 2008, pp. 213-221.

[2] OMG. (2007). *OMG Unified Modelling Language (UML) Superstructure 2.1*. [Online]. Available: http://www.omg.org

[3] T. Murata, “Petri nets: properties, analysis and applications,” *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580, 1989.

[4] F. Villa and R. Costanza, “Design of multi-paradigm integrating modelling tools for ecological research,” *Environmentla Modelling & Software*, 2000.

[5] The OsMoSys approach to multi-formalism modeling of systems, *Software and Systems Modeling (SoSyM)*, vol. 3, pp. 68-81, 2004.

[6] H. Vangheluwe and E. Kerckhoffs, “Computer automated modelling of complex systems,” in *Proc. 15th European Simulation Multi-Conference*, Prague, Czech Republic, 2001.

[7] J. de Lara and H. Vangheluwe, “Computer aided multi-paradigm modelling to process petri-nets and statecharts,” in *Proc. First International Conference on Graph Transformation*, 2002.

[8] J. de Lara, H. Vangheluwe, and M. Alfonseca, “Computer aided multi-paradigm modelling of hybrid systems with AToM3,” in *Proc. Summer Computer Simulation Conference: Society for Computer Simulation International (SCS)*, Montreal, Canada, 2003.

[9] P. J. Mosterman and H. Vangheluwe, “Computer automated multi paradigm modeling in control system design,” in *Proc. IEEE International Symposium on Computer-Aided Control System Design*, Alaska, 2002.

[10] P. J. Mosterman and H. Vangheluwe, “Guest editorial: Special issue on computer automated multi-paradigm modeling,” *ACM Transactions on Modeling and Computer Simulation*, vol. 12, no. 4, pp. 249-255, 2002.

[11] S. Ralf, “Multi‐paradigm modeling,” *Computer-Based Environmental Management*, pp. 97-110.

[12] W3C. (2007). *XSLT Specification 2.0*. [Online]. Availabe: http://www.w3.org

[13] H. Vangheluwe, J. D. Lara, and P. J. Mosterman, “An introduction to multi-paradigm modelling and simulation,” *AI, Simulation and Planning in High Autonomy Systems*, Lisboa, Portugal, 2002.

[14] MDA. (2005). *Model Driven Architecture*. Object Management Group. [Online]. Available: http://www.omg.org/mda/

[15] K. Czarnecki and S. Helsen, “Feature-based survey of model transformation approaches,” *IBM Systems Journal*, vol. 45, 2006.

[16] D. Varró and A. Pataricza, “Generic and meta-transformations for model transformation engineering,” in *Proc. 7th International Conference on the Unified Modeling Language*, Lisbon, Portugal, 2004.

[17] D. Varró, G. Varró, and A. Pataricza, “Designing the automatic transformation of visual languages,” *Science of Computer Programming*, vol. 44, 2002.

[18] D. H. Akehurst, W. G. Howells, and K. D. McDonald-Maier, “Kent model transformation language,” in *Proc. Model Transformations in Practice Workshop*, Montego Bay, Jamaica, 2005.

[19] F. Jouault and I. Kurtev, “Transforming models with ATL,” in *Proc. Model Transformations in Practice Workshop*, Montego Bay, Jamaica, 2005.

[20] Kermeta. (2005). *Triskell Metamodelling Kernel*. [Online]. Available: http://www.kermeta.org
[21] D. H. Akehurst *et al.*, "SiTra: Simple transformations in java," in *Proc. ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems*, 2006, Genova, Italy.
[22] SiTra. (2006). *Simple Transformer (SiTra): An MDE Tool*. [Online]. Available: http://www.cs.bham.ac.uk/~bxb/SiTra.html
[23] OMG. (2008). *MOF 2.0 Query/View/Transformation (QVT) Specification*. [Online]. Available: www.omg.org
[24] M. A. Ameedeen, "A model driven approach to analysis and synthesis of sequence diagrams," Diss., University of Birmingham, 2012.
[25] J. Küster-Filipe, "Modelling concurrent interactions," *Theoretical Computer Science*, vol. 351, no. 2, pp. 203-220, 2006.
[26] K. L. McMillan, "A technique of state space search based on unfolding," *Methods Syst. Des.*, vol. 6, no. 1, pp. 45-65, 1995.
[27] OMG. (2005). XML Metadata Interchange (XMI). v2.0. [Online]. Available: http://www.omg.org
[28] S. Friedenthal, A. Moore, and R. Steiner, "A practical guide to SysML: The systems modeling language," *Elsevier*, 2011.
[29] Process, Rational Unified, *Rational Software Corporation*, Cupertino, 1999.
[30] Poseidon. (2006). Poseidon for UML. Gentleware. [Online]. Available: http://www.gentleware.com/
[31] F. Budinsky *et al.*, *Eclipse Modeling Framework: A Developer's Guide*, Addison Wesley, 2003.
[32] De Bruijn *et al.*, "The web service modeling language WSML: An overview," Springer Berlin Heidelberg, 2006.
[33] R. K. Thiagarajan *et al.* "BPML: A process modeling language for dynamic business models," in *Proc. Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, 2002.
[34] H. Hermans, J. Manderveld, and H. Vogten, *Educational Modelling Language*, 2003.
[35] L. M. Hillah *et al.*, "PNML framework: An extendable reference implementation of the Petri Net Markup Language," *Applications and Theory of Petri Nets*, Springer Berlin Heidelberg, pp. 318-327, 2010.
[36] Alloy Analyzer. (2005). *Alloy Analyzer Website*. [Online]. Available: http://www.alloy.mit.edu/beta/
[37] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2004.
[38] CPNTools. *Computer Tool for Coloured Petri Nets*. [Online]. Available: http://www.wiki.daimi.au.dk/cpntools/
[39] E. Kindler, "The ePNK: An extensible petri net tool for PNML," *Applications and Theory of Petri Nets*, Springer Berlin Heidelberg, pp. 318-327, 2011.

Mohamed Ariff Ameedeen was born in Kuala Lumpur, Malaysia on July 30, 1983 and obtained his first degree with honors in computer systems and networking from Universiti Malaysia Pahang, Malaysia in 2006. He subsequently obtained his doctorate in computer science from University of Birmingham, United Kingdom.

He has been a faculty member in the Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang since 2006, and is currently a senior lecturer in the faculty. He is also the director of IBM Centre of Excellence in the same university.