Abstract—Software development and modelling cannot be separated 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)."

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 Modelling is 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 M. As such, M 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 M.

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 infancy, 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 stylesheets.

### 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.](image)

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 implemented 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 specializaton 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 development model transformation framework.](image)

Fig. 4 depicts the framework that is used by SD2PN in its model transformation where the metamodels of Sequence Diagrams 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 written 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.](image)

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 analysis conducted 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

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.