A Model Driven Method to Represent Free Choice Petri Nets as Sequence Diagram

Thong Weng Jie
Faculty of Computer Science & Software Engineering
Universiti Malaysia Pahang
Kuantan, Pahang
briantwj@gmail.com

Mohamed Ariff Ameedeen
IBM Centre of Excellence
Universiti Malaysia Pahang
Kuantan, Pahang
mohamedariff@ump.edu.my

Abstract – **The role of modelling is important in software or system development. Modelling provides a way for developers to perform model design, model analysis and model synthesis. However, each phase have different requirements which leads to the use of different modelling language for each phase. For example, in model design, usually UML is used due to its semi-formal notational nature, making it easier for developers and other stakeholders without prior knowledge on modelling or programming language to understand. In model analysis, due to its mathematical needs in carrying out mathematical analysis, a modelling language with formal semantics like Petri Net is needed. This results in a condition called heterogeneity, where two modelling language could not communicate with each other. This paper aims to bridge the gap by introducing transformation rules between two type of modelling language which is Petri Net (from model analysis) and UML Sequence Diagram (from model design). This paper also aims to introduce a method of applying Model Driven Development (MDD) model transformation from Petri Nets to UML Sequence Diagram.**

Keywords **– Petri Net; Sequence Diagram; Model Transformation; Model Driven Development**

I. INTRODUCTION

The role of modelling is vital in a software or system development. By using models in a software or system design, level of abstraction can be raised and various types of views of the system can be obtained. Also, with modelling, developers can perform model design, model analysis and model synthesis. In model design, the view of the system is represented in the form of models. Developers usually go for a semi-formal notational modelling language because it represents a good balance between the ease-of-use and precision. In model analysis, developers are able to use the models to perform preliminary analysis for the system. Due to the mathematical nature of the analysis, the modelling language used have to be made up of formal semantics and mathematical-based. Meanwhile in model synthesis, it is the process of allowing two or more models to be joined together based on a set of common elements. Due to the different requirements needed in different phase, different types of modelling languages are needed in model design, analysis and synthesis.

This paper is inspired by the limitation of SD2PN, which is a model driven approach to represent Sequence Diagrams as Free Choice Petri Nets [1]. Using SD2PN, developers are able to transform Sequence Diagram models to Petri Net models. By doing so, model analysis could be performed before the actual implementation phase. Preliminary analysis can be done on the Petri Net models to reduce critical errors. However, developers are still required to manually update the Sequence Diagram models after performing analysis on the Petri Net models. This is because SD2PN is only a one-way transformation process. This makes it a tedious repeated process.

Hence the main contribution of this paper is to design a set of transformation rules to overcome the limitation of SD2PN. The set of transformation rules aims to transform Petri Net models to Sequence Diagram models. With the new set of transformation rules, developers can first use SD2PN to transform Sequence Diagram models to Petri Net models for preliminary analysis. After making changes to the Petri Net models, developers can use the transformation rules showed in this paper to help transform Petri Net models to Sequence Diagram models.

II. FOUNDATION

This section will explain on the basic foundation of Model Driven Development, Petri Net, Sequence Diagram and SD2PN.

A. Model Driven Development

The role of modelling in software development is promoted mainly via Model Driven Development [2]. Models in the context of MDD are captured in machine-readable representation, using languages which are generally adopted by software industry [3]. So it makes them possible to communicate with such models to different parties and reuse them. This in turn results in a lower software production cost and shorter development cycles. In this paper, MDD is used to develop a method to benefit from advantages of using two representation of a system, Petri Nets and Sequence Diagrams.

Meta Object Facility (MOF) [4] is one of the standards for describing metamodels. Metamodel is defined as model of the models, from which models of the system are instantiated.