MODELING ACTIVITY DIAGRAM TO COLORED PETRI NET FOR VALIDATION AND VERIFICATION BASED ON NON FUNCTIONAL PARAMETERS

HOSSEIN NEMATZADEH BALAGATABI

A project report submitted in partial fulfillment of the requirements for the award of the degree of Master of Science (Computer Science)

Faculty of Computer Science and Information Systems
University Technology Malaysia

NOVEMBER 2008
UML is one of the modeling tools which gains wide area of usage in developing softwares. It consists of many diagrams which help developers of a software to produce better product. One of its diagrams is called Activity Diagram. It is a deliverable which is usually produced in the analysis phase of software. It consists of many important benefits, yet it has weaknesses too. One important thing which the current Activity Diagram is unable to do it is that it can not be validated and verified. The current Activity Diagram is a functional diagram and to extract non functional parameters from functional diagram is impossible but through modeling it to colored Petri net and by using the formalism of colored Petri net we may able to verify and validate the Activity Diagram. The ultimate outcome of this study would be handful information to manage the current mentioned Activity Diagram’s weakness. Moreover a computer tool is provided called ADET to validate and verify the activity diagram.
ABSTRAK

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE PAGE</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>DECLARATION</td>
<td></td>
<td>ii</td>
</tr>
<tr>
<td>DEDICATION</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td></td>
<td>iv</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>v</td>
</tr>
<tr>
<td>ABSTRAK</td>
<td></td>
<td>vi</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td></td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>xi</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td></td>
<td>xiv</td>
</tr>
</tbody>
</table>

1 INTRODUCTION

1.1 An introduction to UML 1
1.2 An introduction to Petri Net and Colored Petri Net 3
1.3 Problem Background 4
1.4 Problem Statement 5
1.5 Project Aim 5
1.6 Objectives 6
1.7 Project Scope 6
1.8 Significance of Project 7
1.9 Organization of Report 7

2 LITERATURE REVIEW 8

2.1 Introduction 8
2.2 Modeling Language 9
2.3 Unified Modeling Language 11
   2.3.1 General Description 11
   2.3.2 Modeling 13
   2.3.3 Diagrams 14
   2.3.4 Activity Diagram 16
   2.3.5 Concepts 19
   2.3.6 Criticisms 20
2.4 Petri Nets in a Holistic View 21
   2.4.1 Basic Mathematical Properties 23
   2.4.2 Reachability 25
   2.4.3 Boundedness 26
   2.4.4 Liveness 28
   2.4.5 Main Petri Net Types 29
   2.4.6 Petri Net Capability 31
2.5 Colored Petri Net 33
   2.5.1 Colored Petri Net Capability 36
2.6 Trends 37
2.7 Summary 37

3 METHODOLOGY 38

3.1 Introduction 38
3.2 Phases of Research 39
3.2.1 Investigation of Formalization of Coloured Petri Net 40
3.2.2 Investigation of Nonfunctional Quality Attributes with CPN 40
3.2.3 Developing Transformation Algorithm 41
3.2.4 Developing Validation and Verification for AD 41
3.2.5 Developing the ADET 42

3.3 Case Study 42
3.4 System Requirement for ADET 43
3.5 Summary 43

4 VERIFICATION AND VALIDATION MODEL FOR ACTIVITY DIAGRAM USING COLOURED PETRI NET

4.1 Introduction 44
4.2 Investigation in formalism of colored Petri net 45
   4.2.1 Definition of the coloured Petri net 45
   4.2.2 Characteristics of coloured Petri net 48
4.3 Investigation of Non Functional Quality Attributes with CPN 50
   4.3.1 An example for finding the quality value nondeterministically 53
   4.3.2 Reliability 54
   4.3.3 Security on Network 55
   4.3.4 Security on memories and files 56
   4.3.5 Time efficiency 57
   4.3.6 Resource efficiency 58
4.4 Developing Transformation Algorithm 59
4.5 Verification of Activity Diagram 67
4.6 Validation of Activity Diagram 68
4.7 Summary 69
5  ADET DEVELOPMENT FOR VALIDATION AND VERIFICATION OF AD

5.1 Introduction 70
5.2 ADET Architecture 71
5.3 ADET Functionalities 73
5.4 Example of Operations 75
5.5 Summary 81

6  RESULTS AND DISCUSSIONS 82

6.1 Introduction 82
6.2 Selected Case Study 83
6.3 Mapping 86
6.4 Validating and Verifying 87
6.5 Discussion on the Result 89
6.6 Summary 91

7  CONCLUSION AND FUTURE WORK 92

7.1 Conclusions 92
7.2 Future Works 93

REFERENCES 94
CHAPTER 1

INTRODUCTION

1.1 An Introduction to UML

UML is a visual modeling language. It's a tool for developers for analyzing and designing an object oriented system. UML is a specification introduced by OMG. UML contains some rules and instructions which are commonly graphical. Unified Modeling language is a visual language for specifying, constructing and documenting the artifacts of systems. The word visual in the definition is a key point. The UML is the de facto standard diagramming notation for drawing and presenting pictures (with some text) related to software – primarily OO software. UML is known as one of the most common methods in software engineering. The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process.
The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns and components.

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.
1.2 An Introduction to Petri Net and Colored Petri Net

Petri Net is a tool for studying systems. Petri Net is mathematical and it is a modeling tool. Since we can not study and analyze many fields directly, it is better we analyze them indirectly through modeling. Model is a pattern that conveys important specifications of an under studying system. In most of modeling mathematic is used. System first is modeled to Petri Net, and then the model will be analyzed. The results of analysis lead us to a useful system in other word; the model converts a complicated system to a simple system. This concept is depicted in figure 1.1.

![Figure 1.1: Modeling Schema Using Petri Net](image)
An extension of Petri net is called colored Petri net which is more powerful than Petri net since it can distinguish the tokens. It is more practical and it has more usage than PN. It is developed for systems in which communication, synchronization and resource sharing play an important role CP-nets combine the strengths of ordinary Petri nets with the strengths of a high-level programming language. Petri nets provide the primitives for process interaction, while the programming language provides the primitives for the definition of data types and the manipulations of data values. CP-nets has an intuitive, graphical representation which is appealing to human beings.

1.3 Problem Background

Nowadays, UML is a powerful and advanced methodology for analysis the complicated systems that many of software engineers and designers use to produce softwares. Despite of this, the methodology has a basic shortcoming that is lacking of a tool to verify the correctness of logic function of the diagram. In fact, this model is not still converted to a formal and verified model. Although UML system sequence diagrams are useful and used widely in analyzing, but data in these diagrams are changeless and because of lacking of dynamism, we cannot implement the diagram. Implementation of diagrams is so important that we can verify the correctness of the diagram. On the other hand, one of important problems of analyzing phase in software engineering is to verify all analyzed things before going to the design phase because starting of design phase before verifying analyze phase is a big risk in big projects. So, one of common methodologies in analyze phase is UML that have various diagrams. It does not have the ability to be tested or verified. Many researches have been done to convert UML to Petri Nets. MuDer Jeng and WeiZhao Lu have a methodology that converts UML models to Petri nets for modeling semiconductor manufacturing systems [1]. Some other tried to convert activity diagram to CPNs [2]. But still no efforts have
been done to convert UML models, especially UML diagrams, to a model that can accept fuzzy inputs in order to test and verify the correctness of the diagrams.

1.4 Problem Statement

In this project a new technique will be proposed using coloured Petri Net. Through the technique we can validate and verify the Activity Diagram based on nonfunctional parameters. First it is tried to convert the activity Diagram to coloured Petri net. And then validation and verification process started. The basic question of the research is: Given an Activity Diagram, how we can validate and verify the Activity Diagram based on non functional parameters?

1.5 Project Aim

The aim of the project is:

To provide a modeling facility for validation and verification of UML activity diagrams using colored Petri net.
1.6 Objectives

This project follows four objectives:

a. To investigate the formalism of Coloured Petri Net for determining quality values.
b. To model an Activity Diagram with CPN to find out Activity Diagrams non functional parameters.
c. To develop the verification and validation process of Activity Diagram via CPN.
d. To develop a verification and validation tool for Activity Diagram called ADET.

1.7 Project Scope

This project is focusing on investigating CPN parameters and to find a way to formalize the activity diagram in UML. The scopes of this project are as follow:

a) Initially, using Coloured Petri net to investigate the quality attributes of a software system.
b) To Verify and validate Activity Diagram via CPN.
c) To provide a tool that validates and verifies the activity diagram.
1.8 Significance of Project

Nowadays it is one of the important things in modeling to know how we can find out non-functional parameters of the system. This is not achieved if you have powerful modeling tool that can formalize the software based system. On the other hand we can use the findings from the project to formalize some diagrams in UML like the activity diagram which is straightforward to be converted to the CPN as a formalized modeling tool. On the other hand the tool is used to formalize software systems to find quality attributes.

1.9 Organization of Reports

In the following chapters literature review, methodology, validation and verification model for activity diagram using coloured Petri net, tool development, results and discussions and conclusion and future work will be discussed respectively. Chapter 2 in which literature review will be discussed deeply will be divided in to two parts, first there will be a powerful introduction of keywords and important definitions, second there are related works and backgrounds. Chapter 3 discusses how to achieve our objectives in the project. And chapter 4 concentrates on validation and verification process. Chapter 5 focuses on the architectural aspect of the tool and depicts its operation step by step. Chapter 6 calculates the results. Finally in chapter 7 concludes with the conclusion and discusses the future work.
REFERENCES

2. H. Motameni, A. Movaghar and B. Kardel “Verifying and Evaluating UML Activity by Converting to CPN” Proc of SYNASC’05, Romania, Sep 2005
4. K. Jensen “Coloured Petri Nets”, Computer science Department Aarhus University, Denmark, 1995
5. R. Braek, TIMe, TIMe Electronic Textbook v 4.0 © SINTEF – Modified:1999-07-16
22. W. Lu "Fuzzy UML", Seminararbeit, Sommersemester, 2005
23. Z. Ma "Fuzzy Information Modeling With the Uml". Idea, 2005
24. Z.M. Ma "Extending UML For Fuzzy Information Modeling In Object Oriented Database ", theories and practices, idea group publishing, 2004
26. R. M Colomb” Metamodeling and Model Driven Architecture” version 9 January 2008, Faculty of Computer Science and Information Systems, University of Technology Malaysia, 2008
32. A.Abellard, M.Moncef Ben Khelifa, and M.Bouchouicha “A human computer interface modelled with petri net for disabled transportation “, IEEE, 2005
33. J.Luo, X.Yang, Ch.Huang, “Modeling of integrated resources in network education systems based on petri net ”, IEEE, 2004
39. Baresi & Pezze “Improving UML with petri nets”, Electronic Notes in Theoretical Science 44 No. 4 Published by Elsevier Science, 2001


46. Y. Wang, Sh.Yao, Y. Zhao, M. Zhou “CPN Modeling and Analysis of L2TP”, IEEE, 2001

47. J. M. Fernandes, S. Tjel, J. B.Jørgensen, Ó. Ribeiro” Designing Tool Support for Translating Use Cases and UML 2.0 Sequence Diagrams into a Coloured Petri Net”, IEEE, Sixth International Workshop on Scenarios and State Machines (SCESM'07), 2007


55. S.J.H. Yang, Jeffrey J.P. Tsai, Ch. Chen “Fuzzy Rule Base Systems Verification Using High-Level Petri Nets”, Published by the IEEE Computer Society, 2003


