

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

To my beloved parents and family

Thanks for your immense love, your precious prayers, supports and all that you have done to me. May the blessing of God, shower upon you.

## **ACKNOWLEDGEMENT**

I would like to take this opportunity to express my gratitude to everyone who has assisted me to make this project a success.

First of all I would like to thank my supervisor, Prof. Dr. Safaai Bin Deris for his guidance and assistance and Dr. Siti Zaiton for her discussion and guidance throughout the project. Without them, I would not be able to complete my project in time.

And would like to express my appreciation to those who were involved directly or indirectly supporting me in the completion of this project. I really owe them where my words alone are not worth what they have done for me.

## ABSTRACT

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

UML ialah salah satu alat permodelan yang mendapat penggunaan meluas dalam pembangunan perisian-perisian. Ia mengandungi pelbagai rajah yang membantu pembangun perisian untuk menghasilkan produk yang lebih baik. Satu dari rajah-rajah berkenaan dikenali sebagai Rajah Aktiviti. Ianya bersifat penyampai yang biasanya dihasilkan di dalam fasa analisis bagi sesuatu perisian. Ia mengandungi banyak kelebihan disamping terdapat juga kelemahan. Satu perkara penting yang mana Rajah Aktiviti yang ada sekarang tidak mampu lakukan adalah ia tidak boleh disah dan ditentusahkan. Rajah Aktiviti yang ada kini adalah rajah fungsian dan adalah mustahil mengekstrak parameter-parameter bukan fungsian daripada rajah fungsian, akan tetapi dengan memodelkannya kepada *colored Petri net* serta menggunakan perumusan *colored Petri net* kita mampu untuk mengesah dan menentusahkan Rajah Aktiviti. Dapatan daripada kajian ini berupa maklumat berguna untuk mengurus kelemahan Rajah Aktiviti yang ada sebagaimana disebutkan tadi. Tambahan juga, peralatan computer yang dipanggil *ADET* disediakan untuk mengesah dan menentusahkan rajah aktiviti tersebut.

**TABLE OF CONTENTS**

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE</b>
	<b>TITLE PAGE</b>	<b>i</b>
	<b>DECLARATION</b>	<b>ii</b>
	<b>DEDICATION</b>	<b>iii</b>
	<b>ACKNOWLEDGEMENTS</b>	<b>iv</b>
	<b>ABSTRACT</b>	<b>v</b>
	<b>ABSTRAK</b>	<b>vi</b>
	<b>TABLE OF CONTENTS</b>	<b>vii</b>
	<b>LIST OF TABLES</b>	<b>x</b>
	<b>LIST OF FIGURES</b>	<b>xi</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>xiv</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
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
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>8</b>
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
<b>3</b>	<b>METHODOLOGY</b>	<b>38</b>
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
<b>4</b>	<b>VERIFICATION AND VALIDATION MODEL FOR ACTIVITY DIAGRAM USING COLOURED PETRI NET</b>	<b>44</b>
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

<b>5 ADET DEVELOPMENT FOR VALIDATION AND VERIFICATION OF AD</b>	<b>70</b>
5.1 Introduction	70
5.2 ADET Architecture	71
5.3 ADET Functionalities	73
5.4 Example of Operations	75
5.5 Summary	81
<b>6 RESULTS AND DISCUSSIONS</b>	<b>82</b>
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
<b>7 CONCLUSION AND FUTURE WORK</b>	<b>92</b>
7.1 Conclusions	92
7.2 Future Works	93
<b>REFERENCES</b>	<b>94</b>

**LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE</b>
2.1	Components of Petri Nets	22
2.2	Summary of Early Researches over Different Concepts Using Petri Nets	31
2.3	Summary of Early Researches over Modeling UML Using Petri Nets	32
2.4	Summary of Early Researches over Different Concepts Using Colored Petri Nets	36
4.1	Mapping Table from Activity Diagram to CPN	65
6.1	Results of Activity Diagram Verification in its Third Iteration Using the Tool	89

## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
1.1	Modeling Schema Using Petri Net	3
2.1	Hierarchically View of UML Diagrams	14
2.2	Sample Activity Diagram	17
2.3	Sample Swimlane	18
2.4	Sample Petri Net	24
2.5	Corresponding Matrixes for Petri Net in Figure2.4	24
2.6	Reachability Graph of a Sample Petri Net	27
2.7	Example of Place-Transformation	28
2.8	Graphically Shown Six Main Petri Nets	30
3.1	The main phases of research	39
4.1	An Example of CPN	47
4.2	Determining Quality Values	52
4.3	An Example of Calculating Quality Values	53
4.4	Calculation of reliability	54
4.5	Security on Network	55
4.6	Security on Memories and Files	56
4.7	Time Efficiency	57
4.8	Resource Efficiency	58
4.9	Additional Shapes	60
4.10	CPN for Activity Diagram Figure 2.2	64
4.11	Flowcharts for Mapping from Activity Diagram to CPN	66

5.1	Architecture of the ADET	71
5.2	Logical architecture of ADET	72
5.3	The ADET interface	74
5.4	Sample coloured Petri net	75
5.5	First part of compiling	76
5.6	Second part of compiling	77
5.7	Third part of compiling	78
5.8	Validation	79
5.9	Verification	80
6.1	Relation of Process Sale with its Two Actors	84
6.2	Process Sale Activity Diagram	85
6.3	The mapped activity diagram	86
6.4	The mapped activity diagram	90

## LIST OF ABBREVIATIONS

AD	-	Activity Diagram
ADET	-	Activity Diagram Evaluation Tool
BPMN	-	Business Process Modeling Language
CPN	-	Coloured Petri Net
ESL	-	Energy System Languages
FMC	-	Fundamental Modeling Concept
IDEF	-	ICAM Definition Language
ISO	-	International Organization for Standardization
MDA	-	Model Driven Architecture
MDD	-	Model Driven Development
MDE	-	Model Driven Architecture
MOF	-	Meta Object Facility
OMG	-	Object Management Group
OO	-	Object Oriented
PDA	-	Personal Digital Assistant
PN	-	Petri Net
POS	-	Point Of Sale
RUP	-	Rational Unified Process
SysML	-	System Modeling Languages
UML	-	Unified Modeling Languages
XMI	-	XML Metadata Interchange
XML	-	Extensible Modeling Language

# **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.

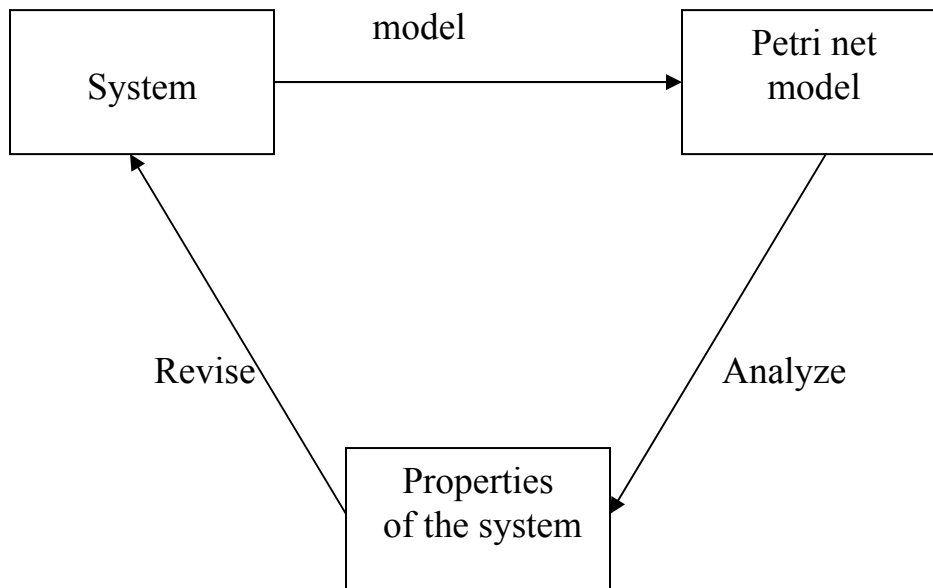
The primary goals in the design of the UML were:

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.
7. Integrate best practices.

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

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 2 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.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Introduction**

Sometimes if you want to check the correctness of the system, you need to model a system first through a tool among several modeling tools. Why do we model? Because it is difficult to assess a system and give an idea on it when we don't have any schema over it. So modeling languages (sometimes called modeling tools) produced to model such complicated systems. One of the most discussed modeling tools is Petri net and its extended versions. Petri net as a modeling language has a broad usage in the world of modeling. Petri net also has an important role in industrial applications [3]. One of the extensions of Petri net is colored Petri net [4]. Due to some extensions colored Petri net has more functionalities than the simple Petri net. Colored Petri net is able to differentiate between input data (token), where as the simple was not able to do this. UML is another famous modeling language, it is used to model systems too, it has several diagrams and it is momentous to software engineering [5]. But as we compare UML to Petri net and its extensions we understand that UML is less powerful than Petri net and its extensions in some areas. This is because UML is less formal than Petri net.

Due to lack of formalism it is difficult to assess UML diagrams perfectly and correctly. So maybe we need to convert UML to a formal language. Here we can use Petri net and its extensions. Actually we try to model a specific modeling language with another modeling language that is UML by Petri net. In the following we have holistic view on modeling languages, UML, Petri net, Colored Petri Net and related background on them respectively.

## 2.2 Modeling Language

A **Modeling Language** is any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure. A modeling language can be graphical or textual. [6] Graphical modeling languages use diagram techniques with named symbols that represent concepts and lines that connect the symbols and that represent relationships and various other graphical annotation to represent constraints. Textual modeling languages typically use standardised keywords accompanied by parameters to make computer-interpretable expressions.

An example of a graphical modeling language and a corresponding textual modeling language is EXPRESS-G and EXPRESS (ISO 10303-11). Not all modeling languages are executable, and for those that are, the use of them doesn't necessarily mean that programmers are no longer required. On the contrary, executable modeling languages are intended to amplify the productivity of skilled programmers, so that they can address more challenging problems, such as parallel computing and distributed systems. Examples of modeling languages are:

- EXPRESS and EXPRESS-G (ISO 10303-11) is an international standard general-purpose data modeling language. It is used among others to specify various ISO standard data models, such as the application protocols of ISO 10303 (STEP), ISO 13584, ISO 15926 and others.
- Unified Modeling Language (UML) is a general-purpose modeling language that is an industry standard for specifying software-intensive systems. UML 2.0, the current version, supports thirteen different diagram techniques, and has widespread tool support.
- Petri nets use variations on exactly one diagramming technique and topology, namely the bipartite graph. The simplicity of its basic user interface easily enabled extensive tool support over the years, particularly in the areas of model checking, graphically-oriented simulation, and software verification.
- IDEF is a family of modeling languages, the most notable of which include IDEF0 [6], for functional modeling, and IDEF1 for information modeling.
- SysML [7] is a Domain-Specific Modeling language for systems engineering that is defined as a UML profile (customization).
- Energy Systems Language [8] (ESL), a language that aims to model ecological energetics & global economics.
- Business Process Modeling Notation [9] (BPMN, and the XML form BPML) is an example of a Process Modeling language.
- Fundamental Modeling Concepts [10] (FMC) modeling language for software-intensive systems.
- Gellish English, a structured subset of natural English, which includes a Gellish English dictionary and in which facts are expressed in a standardised Gellish Database Table [11]. Gellish English is not a meta language, but integrates and upper ontology with domain ontologies in one language.

Various kinds of modeling languages are applied in different disciplines, including computer science, information management, business process modeling, software engineering, and systems engineering. Modeling languages can be used to specify

system requirements, structures and behaviors. Modeling languages are intended to be used to precisely specify systems so that stakeholders (e.g., customers, operators, analysts, designers) can better understand the system being modeled. The more mature modeling languages are precise, consistent and executable. Informal diagramming techniques applied with drawing tools are expected to produce useful pictorial representations of system requirements, structures and behaviors, but not much else. Executable modeling languages applied with proper tool support, however, are expected to automate system verification, validation, simulation and code generation from the same representations.

## **2.3 Unified Modeling Language**

In the field of software engineering, the Unified Modeling Language (UML) is a standardized specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model. We have extensions for UML, one of them is fuzzy UML. Recently, a model named fuzzy UML has been introduced [22, 23, 24] which has the UML characteristics, is also able to model uncertain concepts.

### **2.3.1 General Description**

UML is officially defined at the Object Management Group (OMG) by the UML metamodel, a Meta-Object Facility metamodel (MOF) [26]. Like other MOF-based specifications, the UML metamodel and UML models may be serialized in XML

Metadata Interchange (XMI). UML was designed to specify, visualize, construct, and document software-intensive systems. UML is not restricted to modeling software. UML is also used for business process modeling, systems engineering modeling and representing organizational structures. The Systems Modeling Language (SysML) is a Domain-Specific Modeling language for systems engineering that is defined as a UML 2.0 profile.

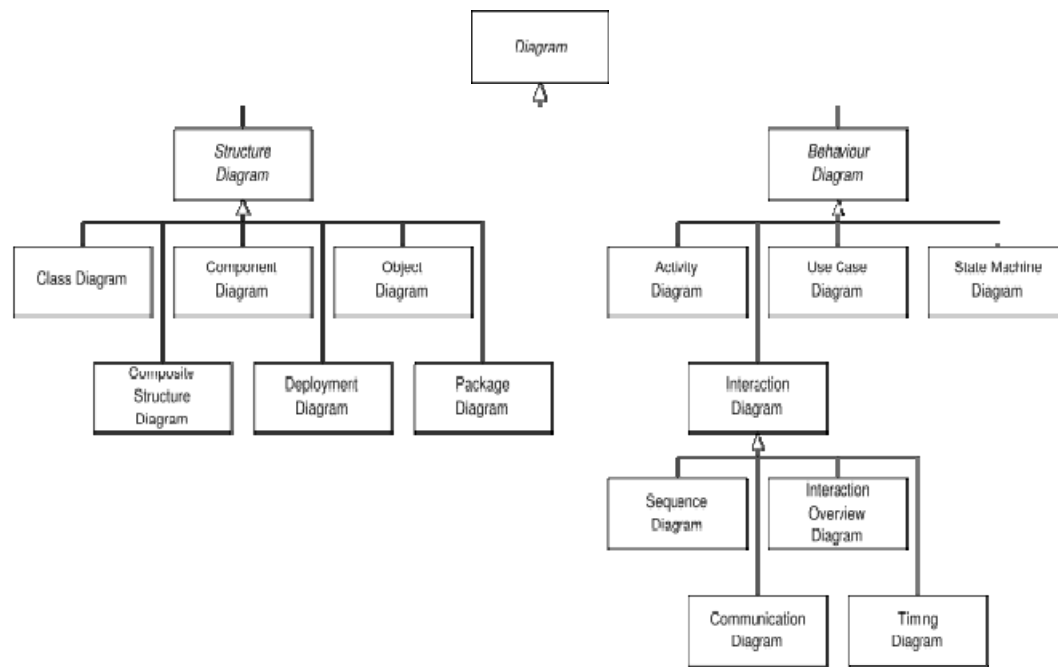
UML has been a catalyst for the evolution of model-driven technologies, which include model-driven development (MDD) [12], model-driven engineering (MDE) [13, 14], and model-driven architecture (MDA) [15]. By establishing an industry consensus on a graphic notation to represent common concepts like classes, components, generalization, aggregation, and behaviors, UML has allowed software developers to concentrate more on design and architecture. UML models may be automatically transformed to other representations (e.g. Java) by means of QVT-like transformation languages, supported by the OMG. UML is extensible, offering the following mechanisms for customization: profiles and stereotype. The semantics of extension by profiles have been improved with the UML 2.0 major revision. UML is not a method by itself; however, it was designed to be compatible with the leading object-oriented software development methods of its time (for example OMT, Booch, Objectory). Since UML has evolved, some of these methods have been recast to take advantage of the new notation (for example OMT), and new methods have been created based on UML. The best known is Rational Unified Process (RUP). There are many other UML-based methods like Abstraction Method, Dynamic Systems Development Method, and others, designed to provide more specific solutions, or achieve different objectives.

### 2.3.2 Modeling

It is very important to distinguish between the UML model and the set of diagrams of a system. A diagram is a partial graphical representation of a system's model. The model also contains a "semantic backplane" — documentation such as written use cases that drive the model elements and diagrams. UML diagrams represent three different views of a system model. Functional requirements view that emphasizes the functional requirements of the system from the user's point of view which includes use case diagrams. Static structural view that emphasizes the static structure of the system using objects, attributes, operations, and relationships which includes class diagrams and composite structure diagrams. Dynamic behavior view that emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects which includes sequence diagrams, activity diagrams and state machine diagrams. UML models can be exchanged among UML tools by using the XMI interchange format.

### 2.3.3 Diagrams

In UML 2.0 there are 13 types of diagrams. To understand them, it can be useful to categorize them hierarchically, as shown in the hierarchy chart below in figure 2.1.



**Figure 2.1:** Hierarchically View of UML Diagrams

Structure diagrams [27, 28] emphasize what things must be in the system being modeled:

- Class diagram
- Component diagram
- Composite structure diagram
- Deployment diagram
- Object diagram
- Package diagram

Behavior diagrams [28, 30] emphasize what must happen in the system being modeled:

- Activity diagram
- State Machine diagram
- Use case diagram

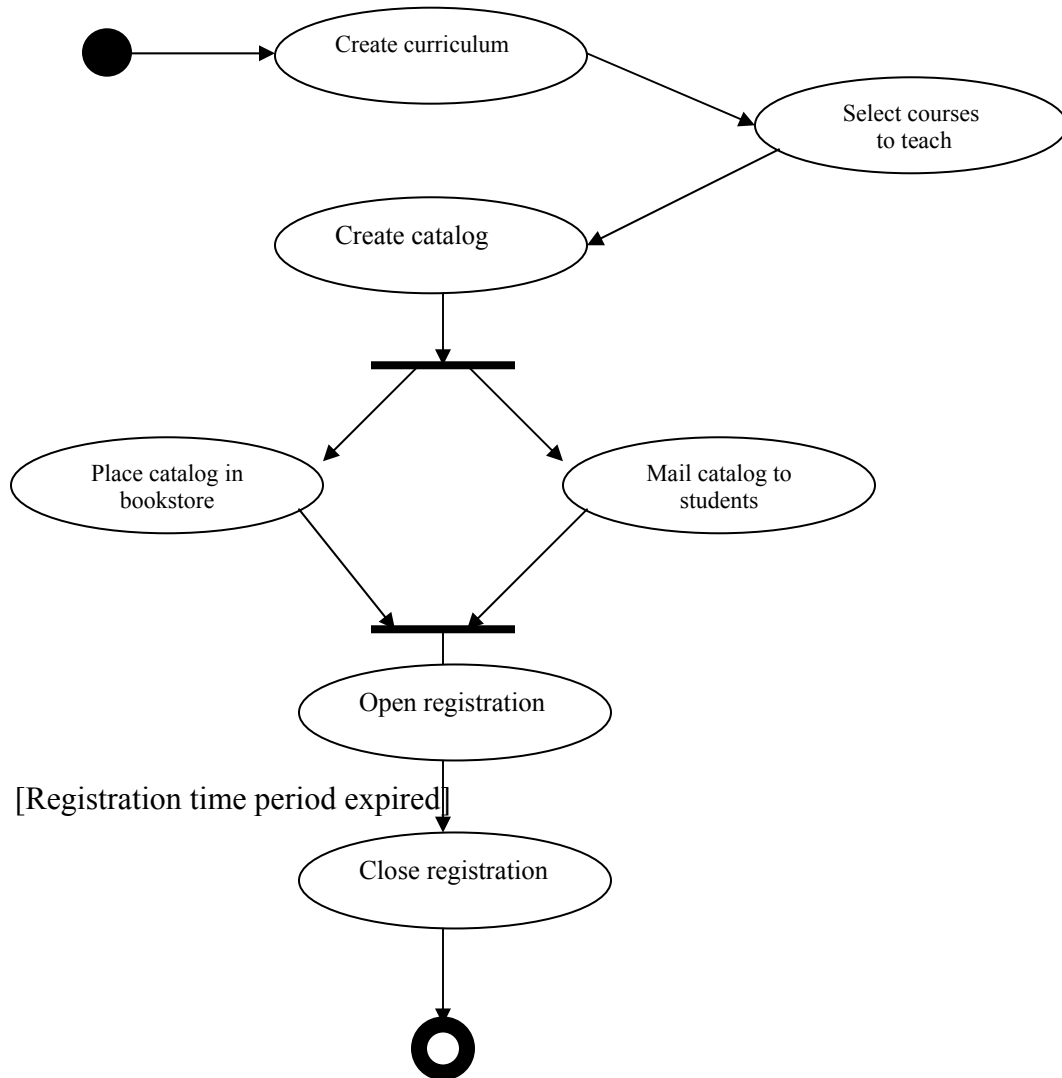
Interaction diagrams, [28] a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled:

- Communication diagram
- Interaction overview diagram (UML 2.0)
- Sequence diagram
- UML Timing Diagram (UML 2.0)

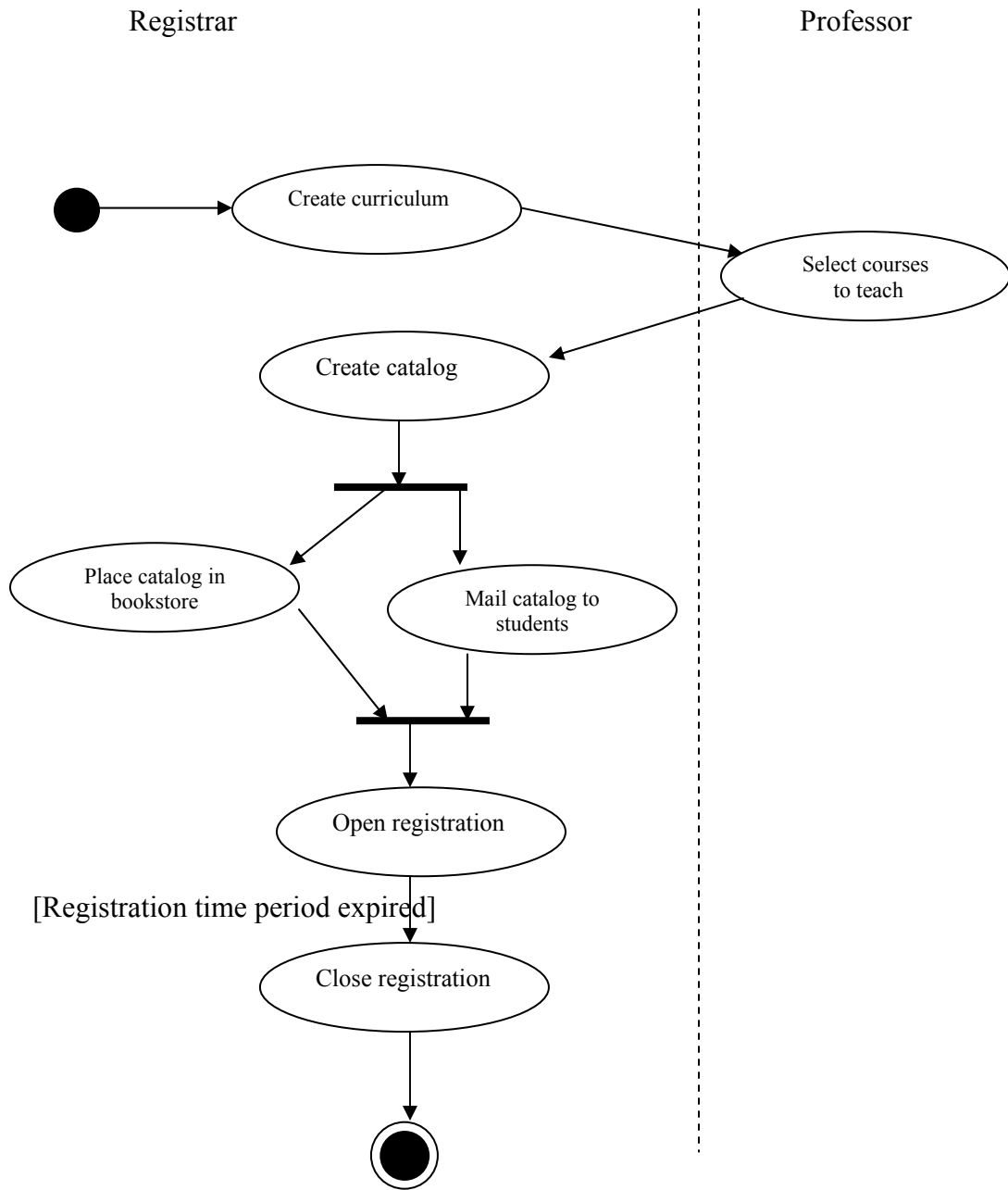
The Protocol State Machine is a sub-variant of the State Machine. It may be used to model network communication protocols. UML does not restrict UML element types to a certain diagram type. In general, every UML element may appear on almost all types of diagrams. This flexibility has been partially restricted in UML 2.0. In keeping with the tradition of engineering drawings, a comment or note explaining usage, constraint, or intent is always allowed in a UML diagram.

### 2.3.4 Activity Diagram

Among these diagrams activity diagram [28] is the most prominent one since it shows the flow of control. This concept is shown in figure 2.2. The other reason for choosing activity diagram to talk about here is that activity diagram is the diagram which is chosen to be enhanced later in my project. Activity diagrams show the flow of control. As illustrated in Figure 2, you can see activities represented as rounded rectangles. Activities are typically action states – states that transition automatically to the next state after the action is complete. The filled in circle represents the start of the activity diagram – where the flow of control starts. Transitions are shown as arrows show how you move from activity to activity. Synchronization bars show how activities happen in parallel. As a first look one may say that it is the same flow chart and exactly it is, except it is not at the programming level. It illustrates what use case has to happen. Activity diagram is also used to show how things flow between use cases. Activity diagram may be used to show the flow between the methods of a class. One of the things that can be showed with activity diagrams is swimlanes. One of the best ways to use swimlanes and activity diagrams is to show ownership. It clearly illustrates to all parties what individual or group is responsible for what activity. The concept of swimlane is depicted in figure 2.3.



**Figure 2.2:** Sample Activity Diagram



**Figure 2.3:** Sample Swimlane

### 2.3.5 Concepts

UML uses many concepts from many sources. Notable concepts are listed here.

*A) For structure*

Actor, attribute, class, component, interface, object, package.

*B) For behavior*

Activity, event, message, method, operation, state, use case.

*C) For relationships*

Aggregation, association, composition, dependency, generalization (or inheritance).

*Other concepts*

- Stereotype. It qualifies the symbol it is attached to.
- Multiplicity notation which corresponds to database modeling cardinality, e.g., 1,
- 0..1, 1..\*
- Role

### 2.3.6 Criticisms

Although UML is a widely recognized and used modeling standard, it is frequently criticized for the following deficiencies:

- **Language bloat.** UML is often criticized [16] as being gratuitously large and complex. It contains many diagrams and constructs that are redundant or infrequently used. This criticism is more frequently directed at UML 2.0 than UML 1.0, since newer revisions include more design-by-committee compromises.
- **Problems in learning and adopting.** The problems cited above make learning and adopting UML problematic, especially when required of engineers lacking the prerequisite skills.
- **Only the code is in sync with the code.** Another perspective holds that it is working systems that are important, not beautiful models. As Jack Reeves succinctly put it, "The code is the design. [17] Pursuing this notion leads to the need for better ways of writing software; UML has value in approaches that *compile the models* to generate source or executable code. This however, may still not be sufficient since it is not clear that UML 2.0's Action Semantics exhibit Turing completeness.
- **Cumulative Impedance/Impedance mismatch.** As with any notational system, UML is able to represent some systems more concisely or efficiently than others. Thus a developer gravitates toward solutions that reside at the intersection of the capabilities of UML and the implementation language. This problem is particularly pronounced if the implementation language does not adhere to orthodox object-oriented doctrine, as the intersection set between UML and implementation language may be that much smaller.
- **Aesthetically inconsistent.** This argument states that the adhoc mixing of abstract notation (2-D ovals, boxes, etc) make UML appear jarring and that more effort could have been made to construct uniform and aesthetically pleasing representations.

- **Tries to be all things to all people.** UML is a general purpose modeling language, which tries to achieve compatibility with every possible implementation language. In the context of a specific project, the most applicable features of UML must be delimited for use by the design team to accomplish the specific goal. Additionally, the means of restricting the scope of UML to a particular domain is through a formalism that is not completely formed, and is itself the subject of criticism.

## 2.4 Petri Nets in a Holistic View

A Petri net (also known as a place/transition net or P/T net) is one of several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations. As such, a Petri net has place nodes, transition nodes, and directed arcs connecting places with transitions. Petri nets were invented in 1962 by Carl Adam Petri.




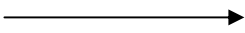
A Petri net is a 5-tuple  $(S, T, F, M_0, W)$ , where [18]

$S$  is a set of places.

- $T$  is a set of transitions.
- $F$  is a set of arcs known as a flow relation. The set  $F$  is subject to the constraint that no arc may connect two places or two transitions, or more formally:  $F \subseteq (S \times T) \cup (T \times S)$ .
- $M_0 : S \rightarrow \mathbb{N}$  is an initial marking, where for each place  $s \in S$ , there are  $n_s \in \mathbb{N}$  tokens.
- $W : F \rightarrow \mathbb{N}^+$  is a set of arc weights, which assigns to each arc  $f \in F$  some  $n \in \mathbb{N}^+$  denoting how many tokens are consumed from a place by a transition, or alternatively, how many tokens are produced by a transition and put into each place.

A variety of other formal definitions exist. Some definitions do not have arc weights, but they allow multiple arcs between the same place and transition, which is conceptually equal to one arc with a weight of more than one. Basically the components of petri net are shown graphically as it is shown in table 2.1

**Table 2.1:** Components of Petri Nets

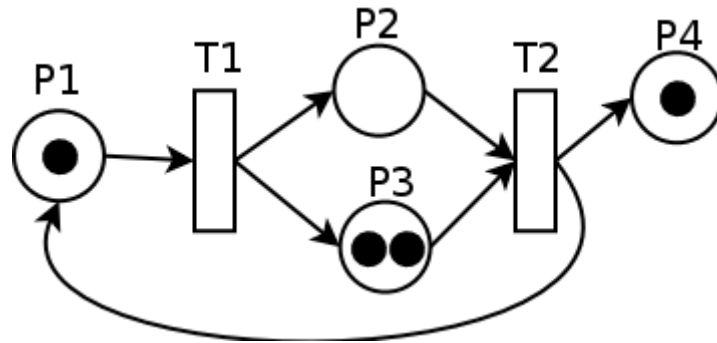
<b>Components of a simple petri net</b>	<b>Graphically representation</b>
<b>Token</b>	
<b>Transition</b>	
<b>Place</b>	
<b>Arc</b>	

### 2.4.1 Basic Mathematical Properties

The state of a Petri net can be represented as an M vector, where the 1st value of the vector is the number of tokens in the 1st place of the net, the 2nd is the number of tokens in the 2nd place, and so on. Such a representation fully describes the state of a Petri net.

A state-transition list,  $\vec{\sigma} = \langle M_{i_0} t_{i_1} M_{i_1} \dots t_{i_n} M_{i_n} \rangle$ , which can be shortened to simply  $\vec{\sigma} = \langle t_{i_1} \dots t_{i_n} \rangle$  is called a firing sequence if each and every transition satisfies the firing criteria (i.e. there are enough tokens in the input for every transition). In this case, the state-transition list of  $\langle M_{i_0} M_{i_1} \dots M_{i_n} \rangle$  is called a trajectory, and  $M_{i_n}$  is called reachable from  $M_{i_0}$  through the firing sequence of  $\vec{\sigma}$ . Mathematically written:  $M_{i_0}[\vec{\sigma} > M_{i_n}$ . The set of all firing sequences that can be reached from a net N and an initial marking  $M_0$  are noted as L (N,  $M_0$ ).

The state-transition matrix W – is | S | by | T | large, and represents the number of tokens taken by each transition from each place. Similarly, W + represents the number of tokens given by each transition to each place. The sum of the two,  $W = W^+ - W^-$  can be used for calculating the above mentioned equation of  $M_{i_0}[\vec{\sigma} > M_{i_n}$  which can now be simply written as  $M_0 - M_n = W^T \cdot \sigma$ , where  $\sigma$  is a vector of how many times each transition fired in the sequence. Note that just because the equation can be satisfied, does not mean that it can actually be carried out - for that there should be enough tokens for each transition to fire, i.e. the satisfiability of the equation is required but not sufficient to say that state  $M_n$  can be reached from state  $M_0$ . A sample Petri net is provided in figure 2.4. Likewise corresponding matrixes are in figure 2.5.



**Figure 2.4:** A Sample Petri Net

$$W^+ = \begin{bmatrix} * & t1 & t2 \\ p1 & 0 & 1 \\ p2 & 1 & 0 \\ p3 & 1 & 0 \\ p4 & 0 & 1 \end{bmatrix} \quad W^- = \begin{bmatrix} * & t1 & t2 \\ p1 & 1 & 0 \\ p2 & 0 & 1 \\ p3 & 0 & 1 \\ p4 & 0 & 0 \end{bmatrix} \quad W = \begin{bmatrix} * & t1 & t2 \\ p1 & -1 & 1 \\ p2 & 1 & -1 \\ p3 & 1 & -1 \\ p4 & 0 & 1 \end{bmatrix}$$

$$M_0 = [1 \ 0 \ 2 \ 1]$$

**Figure 2.5:** Corresponding Matrixes for Petri Net in Figure 2.4

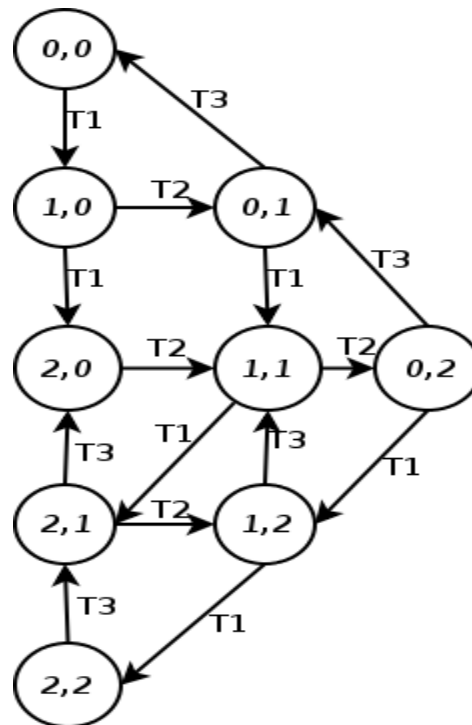
### 2.4.2 Reachability

All states that can be reached from a net  $N$  with an initial marking  $M_0$  are denoted as  $R(N, M_0)$ . The reachability problem is then the following: is it true that  $M_w \in R(N, M_0)$  where  $M_w$  is, for example, an erroneous state.

The reachability of a petri net's states can be represented with a reachability graph, a directed graph whose points represent states (i.e.  $M$ ) and arcs represent transitions between two such states. The graph is constructed through breadth-first search as follows: the starting state ( $M$ ) is taken, and all possible transitions are explored from this state, then the transitions from these states, and so on. As the reachability graph may be infinitely large, breadth-first search is preferred because depth-first search would not find all possible states even if given infinite time. While reachability seems to be a good tool to find erroneous states, for practical problems the constructed graph usually has far too many states to calculate. To alleviate this problem, linear temporal logic is usually used in conjunction with the tableau method to prove that such states cannot be reached. Reachability is known to be decidable, in at least exponential time. All known general algorithms so far, however, employ non-primitive recursive space [17].

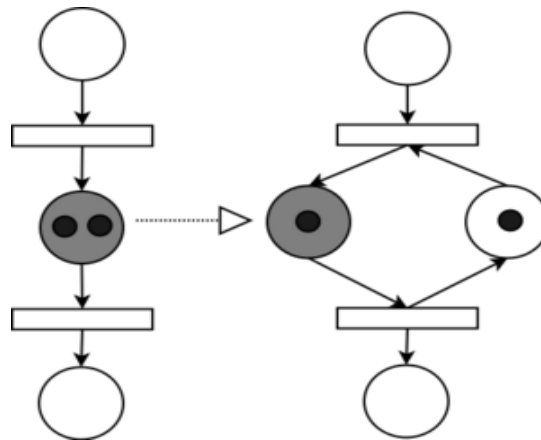
### 2.4.3 Boundedness

A Petri net is inherently  $k$ -bounded if in no reachable state can at any place contain more than  $k$  tokens. A Petri net is safe if it is 1-bounded. Naturally, the initial  $M_0$  marking is also restricted by the boundedness. Note that a Petri net is inherently bounded if and only if all its reachability graphs (i.e reachability graphs with all possible starting states) all have a finite number of states. Formally,  $K : S \rightarrow \mathbb{N}^+$  is a set of capacity restrictions, which assigns to each place  $s \in S$  some positive number  $n \in \mathbb{N}^+$  denoting the maximum number of tokens that can occupy that place. A net in which each of its places has some capacity  $k$ , is known as an 'inherently  $k$ -bounded' Petri net. Boundedness is decidable by looking at covering, by constructing the Karp-Miller Tree. In computer programming and other applications, the property of boundedness of a Petri net is used to model that system resources such as CPUs are bounded. Boundedness of a certain place in an inherently bounded net can be mimicked in a non-inherently bounded net by doing a place-transformation, where a new place (called counter-place) is created, and all transitions that put  $x$  tokens to the original place take  $x$  tokens from the counter-place, and all transitions that take away  $x$  tokens from the original place put  $x$  tokens to the counter-place. The number of tokens in  $M_0$  must now satisfy the equation  $\text{place} + \text{counter-place} = \text{boundedness}$ . Thus, doing a place-transformation for all places in a bounded net, and restricting the starting state  $M_0$  to conform to the above noted equality, a bounded net can easily be transformed to a non-bounded net. Therefore any analysis that is used on inherently non-bounded nets can be used on bounded nets (but not the other way around). this concept is shown in figure 2.6.



**Figure 2.6:** Reachability Graph of a Sample Petri Net, if the net is 2-bounded. In this case, it can only have a maximum of 9 (or  $3^2$ ) states

In figure 2.7 the other concept of petri net which is place transformation is depicted. The grey place that was originally inherently 2-bounded has been transformed into two places: a grey original, and a counter place.



**Figure 2.7:** Example Place Transformation.

#### 2.4.4 Liveness

Petri nets can be described as having different levels of liveness  $L_0 - L_4$ . While the levels of liveness are defined on transitions within the Petri net, the Petri net itself is considered  $L_K$  live if and only if every transition within it is  $L_K$  live.

A Petri net  $(N, M_0)$ 's transition  $t$  is

- $L_0$  live, or dead, if and only if it can not be fired, i.e. it is not in any firing sequence  $\vec{\sigma}$  where  $\vec{\sigma} \in L(N, M_0)$

- L1 live if and only if it can possibly be fired, i.e. it is in a firing sequence  $\vec{\sigma}$  where  $\vec{\sigma} \in L(N, M_0)$
- L2 live if and only if for any  $k$  positive whole number,  $t$  can be fired at least  $k$  times in a firing sequence  $\vec{\sigma}$  where  $\vec{\sigma} \in L(N, M_0)$
- L3 live if and only if there exists a firing sequence  $\vec{\sigma} \in L(N, M_0)$  where  $t$  is fired infinitely
- L4 live or simply live if and only if in any reachable state  $M$  (i.e.  $\forall M \in R(N, M_0)$ ),  $t$  is L1 live

Note that these are increasingly stringent requirements such that if a transition is L3 live for example, it is automatically L1 and L2 live as well. As an example, (b) Example Petri net is live with the given initial state, but with a different (e.g. totally empty) initial state, all of its transitions are dead. In computer programming and other applications, the property of liveness of a Petri net is used to model that the system can never lock up.

### 2.4.5 Main Petri Net Types

There are six main types of petri net:

State Machine (SM) - here, every transition has one incoming arc, and one outgoing arc. This means, that there can not be concurrency, but there can be conflict (i.e. Where should the token from the place go? To one transition or the other?).

Mathematically:  $\forall t \in T : |t \bullet| = |\bullet t| = 1$

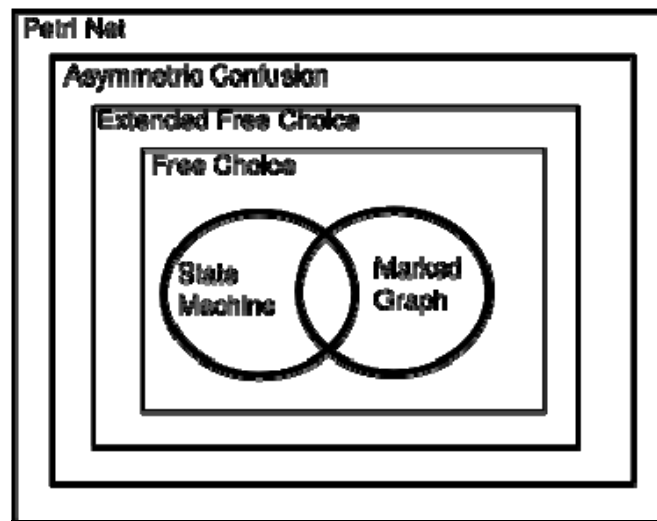
1. Marked Graph (MG) - here, every place has one incoming arc, and one outgoing arc. This means, that there can not be conflict, but there can be concurrency.

Mathematically:  $\forall s \in S : |s \bullet| = |\bullet s| = 1$

2. Free choice (FC) - here, an arc is either the only arc going from the place, or it is the only arc going to a transition. I.e. there can be both concurrency and conflict, but not at the same time. Mathematically:  $\forall s \in S : (|s \bullet| \leq 1) \vee (\bullet(s) = \{s\})$
3. Extended free choice (EFC) - a Petri net that can be transformed into an FC.
4. Asymmetric choice (AC) - concurrency and conflict (in sum, confusion), but not asymmetrically. Mathematically:  

$$\forall s_1, s_2 \in S : (s_1 \bullet \cap s_2 \bullet \neq \emptyset) \rightarrow [(s_1 \bullet \subseteq s_2 \bullet) \vee (s_2 \bullet \subseteq s_1 \bullet)]$$
5. Multiple Asymmetric choice (MAC) - multiple concurrency and conflict (in sum, multiple confusion). Mathematically: for a set  $|P|=k$ ,  $\forall t \in T$  exist a subset  $\bullet t$ , and  $\bullet T$  contains all subset and is the Power Set  $2^k$  without the empty subset
6. Petri Net (PN) - confusion is allowed (i.e. everything is allowed).

Six main types of Petri Net is depicted in figure 2.8



**Figure 2.8:** Graphically Shown Six Main Petri Nets

### 2.4.6 Petri Net Capability

As it is mentioned before Petri net has a broad usage in different areas of knowledge. One of the fields that Petri net is quite capable in it which is our subject of discussion is modeling. Of course in the other areas of knowledge the trace of Petri Net is obvious. Lots of works have been done in these areas. Table 2.2 lists some of the early contributions.

**Table 2.2:** Summary of Early Researches Over different concepts Using Petri Nets

Concept	Author, Year
Modeling	Drathen Arne von, 2007 [31] Abellard, khelifa, bouchounicha, 2005 [32] Junzhou Luo, Xiaoyan Yang, Chen Huang, 2004 [33] Ferrarini & luigi, 2004 [34] Tricas & Ezpeleta, 2003 [35] Hanzalek, 1998 [36] Juanole & faure, 1989 [37]
Design	Cbang-Pin Lin, Yi-Pin Lin, Mu Der Jeng, 2004 [52] Zhou, M.C.; DiCesare, F, 1990 [53]
Verification	S. Castano , P. Samarati , C.Villa, 1993 [54] Stephen J.H. Yang, Jeffrey J.P. Tsai, Chyun-Chyi Chen, 2003 [55] Jonathan Billington, Geoffrey R. Wheeler , Michael C. Wilbur-ham, 1988 [58]
Tool	Itamar S. Lima, Angelo Perkusicht ,Jorge C. A. de Figueiredol, 1996 [56] S. Distefano, A. Puliafito, M. Scarpa, 2005 [57]

Precisely speaking, in field of UML modeling petri net was used too. On the other hand some usages of petri net as an evaluation tool is mentioned. These two may support the idea of fuzzy petri net application in the current project. Table 2.3 contains those researches that have been done based on modeling UML with petri net.

**Table 2.3:** Summary of Early Researches over Modeling UML Using Petri Nets

Concept	Author, Year
UML Modeling	Bernardi & donatelli & merseguer, 2002 [38] Baresi & Pezze, 2001 [39] Saldhana & shatz, 2000[40]
Evaluation	Gerrit K. Janssens, An Caris, Katrien Ramaekers,2008[60] Osama S. Youness, Wail S. El-Kilani, Waiel F. Abd El-Wahed,2008[61] Haoxun Chen; Amodeo, L.; Feng Chu; Labadi, K.2005[59]

## 2.5 Colored Petri Net

Petri net has many extensions, the most important ones are *stochastic petri nets*[18,20], *timed Petri nets*[20,21], *colored petri net* [29] and etc. we focus on the third one to model UML diagrams because it is quite useful in modeling and so many works have been done to model UML diagrams through CPN. The other reason that CPN is chosen to talk about here is that the mapping which is going to be produced later (mapping from activity diagram to CPN) utilizes CPN as its basement. So in this section we try to take a very brief look at CPN.

Coloured Petri Nets (CP-nets or CPNs) is a modelling language developed for systems in which communication, synchronisation 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 have an intuitive, graphical representation which is appealing to human beings. A CPN model consists of a set of modules (pages) which each contains a network of places, transitions and arcs. The modules interact with each other through a set of well-defined interfaces, in a similar way as known from many modern programming languages. The graphical representation makes it easy to see the basic structure of a complex CPN model, i.e., understand how the individual processes interact with each other. CP-nets also have a formal, mathematical representation with a well-defined syntax and semantics. This representation is the foundation for the definition of the different behavioural properties and the analysis methods. Without the mathematical representation it would have been totally impossible to develop a sound and powerful CPN language. However, for the practical use of CP-nets and their tools,

it suffices to have an intuitive understanding of the syntax and semantic. This is analogous to programming languages which are successfully applied by users who are not familiar with the formal, mathematical definitions of the languages.

CPN models can be made with or without explicit reference to time. Untimed CPN models are usually used to validate the functional/logical correctness of a system, while timed CPN models are used to evaluate the performance of the system. There are many other languages which can be used to investigate the functional/logical correctness of a system or the performance of it. However, it is rather seldom to find modelling languages that are well-suited for both kinds of analysis. CP-nets can be simulated interactively or automatically. In an interactive simulation the user is in control. It is possible to see the effects of the individual steps directly on the graphical representation of the CP-net. This means that the user can investigate the different states and choose between the enabled transitions. An interactive simulation is similar to single-step debugging. It provides a way to "walk through" a CPN model, investigating different scenarios and checking whether the model works as expected. This is in contrast to many off-the-shelf simulation packages which often act as black boxes, where the user can define inputs and inspect the results, but otherwise have very little possibility to understand and validate the models on which the simulations build. It is our experience that the insight and detailed knowledge of a system, which the users gain during the development and validation of a simulation model, is often as important as the results that the users get from the actual simulation runs. Automatic simulations are similar to program executions. Now the purpose is to be able to execute the CPN models as fast and efficient as possible, without detailed human interaction and inspection. However, the user still needs to interpret the simulation results. For this purpose it is often suitable to use animated, graphical representations providing an abstract, application-specific view of the current state and activities in the system. CP-nets also offer more formal verification methods, known as state space analysis and invariant analysis. In this way it is possible to prove, in the mathematical sense of the word, that a system has a certain set of behavioral properties. However, industrial

systems are often so complex that it is impossible or at least very expensive to make a full proof of system correctness. Hence, the formal verification methods should be seen as a complement to the more informal validation by means of simulation. The use of formal verification is often restricted to the most important subsystems or the most important aspects of a complex system.

### 2.5.1 Coloured Petri Net Capability

Like PN, One of the fields that colored Petri net is quite capable in it which is our subject of discussion is modeling. It has capability in other areas too. Lots of works have been done in this area. Table 2.4 lists some of the early contributions.

**Table 2.4:** Summary of Early Researches Over different concepts Using Colored Petri nets

Concepts	Author, year
Modeling	Hui Kang & Xiuli Yang & Sinmiao Yuan, 2007 [41] C.H Yang & J.X Liu & H.H. Chen & X.S. Luo, 2007 [42] Peng, Lei & Wu, Lei & Ye, Yalan & Yu, Fengqi & Yuan, Hai, 2007 [43] A. Kovacs & E. Nemeth & K. M. Hantos, 2005 [44] T.E. Lee & J.S. Wu & C.H. Lin ,2003[45] Yongwei Wang & Shaowen Yao & Ying Zhao & Mingtian Zhou, 2001[46]
Tools	João M. Fernandes & Simon Tjel & Jens Bæk Jørgensen & Óscar Ribeiro, 2007[47]
Methodology	Amorim, L.& Maciel, P.& Nogueira, M.& Barreto, R.& Tavares, E, 2005 [48]
Algorithm	C.H. Lin, 2003[49]
Analysis	Shaowen Yao & Mingtian Zhou & Jiazhi Zen, 2000[50] Shaowen Yao & Jonathan Billington & Mingtiaai Zhou, 2000[51]

## **2.6 Trends**

The researches that have been studied here mostly focused on modeling. And some other concentrated in some other concepts like methodology, analysis, techniques and tools. It can be predicted that modeling in different area of knowledge is going to be more important in future research as it is now. More researches are going to carry out by researchers in modeling.

## **2.7 Summary**

This chapter has explained the related literature review and also the related research work on modeling with PN and CPN. And all this partial results have contributed to carry out this project in giving a better and concrete conclusion.

## **CHAPTER 3**

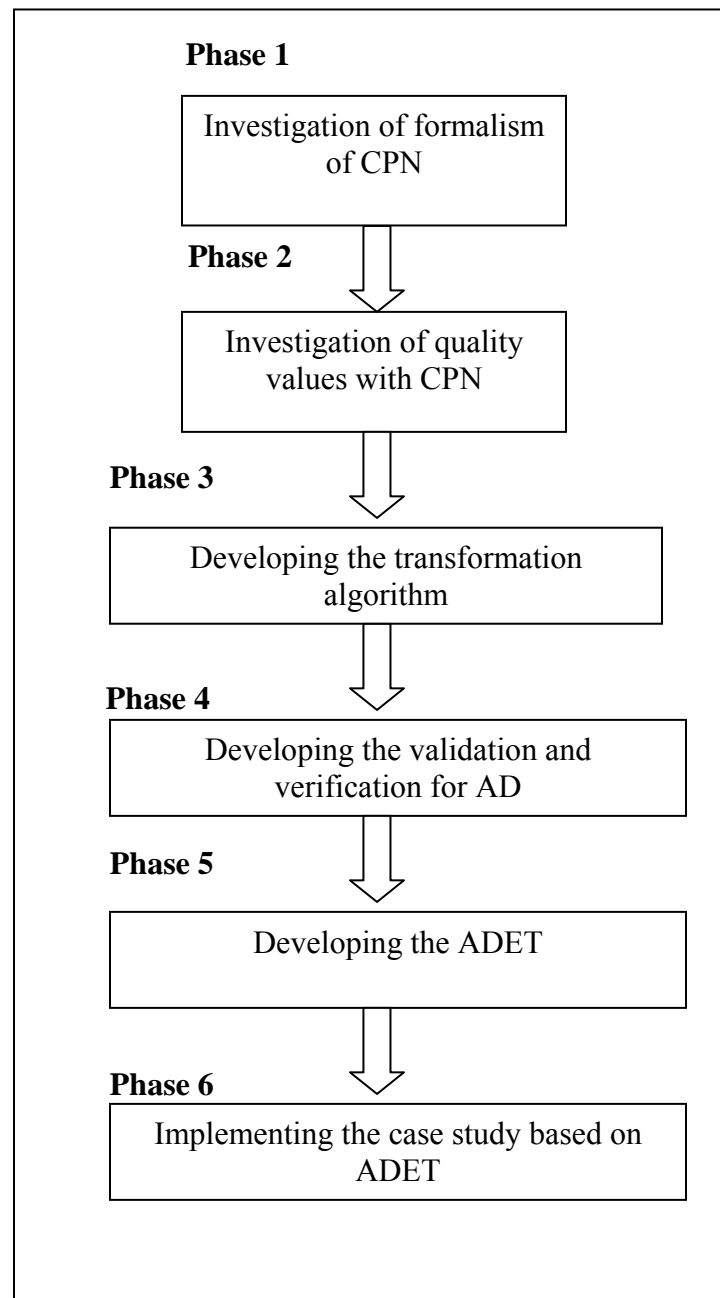
### **METHODOLOGY**

#### **3.1 Introduction**

This chapter explains the methodology that will be used in formalizing activity diagram and finding non functional parameters. In this chapter the steps of how the methodology is produced will be talked shortly. Then the selected case study is introduced and the implementation of the computer tool based on the software, hardware and equipment used is discussed briefly.

### 3.2 Phases of the Research

In the following sections the steps including in the methodology is provided. Generally six steps include the methodology process as it is considered in figure 3.1.



**Figure 3.1:** The main phases of research

### **3.2.1 Investigation of Formalization of Coloured Petri Net**

In this step the survey and investigation is done to show how we can use the formalism of coloured Petri net and how we can derive formulas with that for coloured Petri net. Basically the definition of coloured Petri net is given and then characteristics of coloured Petri net is discussed. In the definition part it is tried to define the coloured Petri net deeply and to show how it works then in another subsection the characteristics of coloured Petri net in terms of its strength in modeling and evaluating systems is given.

### **3.2.2 Investigation of Nonfunctional Quality Attributes with CPN**

In this chapter first it is tried to show that how we can derive nonfunctional parameters (quality attributes) with CPN. The concept of time lags and probability of firing is reviewed. And it is shown that how we can find quality values either nondeterministically or deterministically then in the following subsections an example of finding quality values nondeterministically is given and finally the way for calculation of five nonfunctional parameters is given. The five nonfunctional parameters in this section discussed these five nonfunctional parameters are reliability, security on the network, security on memories and files, time efficiency and resource efficiency.

### **3.2.3 Developing Transformation Algorithm**

In this section of the methodology, first some additional shape is introduced, these shapes are Or-split, Or-join, And-split and And-join for decision, merge, fork and join respectively in the Activity Diagram, then steps needed to transform an activity diagram to coloured Petri net is given and mapping table is introduced. Furthermore the flowchart that shows steps needed for conversion is given. Via this transformation algorithm one can map any Activity Diagram to the equivalent coloured Petri Net. In transformation algorithm, coloured Petri net rules are considered. It is tried to develop the transformation algorithm simply so one can understand it fast.

### **3.2.4 Developing Validation and Verification for AD**

In this section validation and verification algorithm is developed. In the validation process it is tried to shown that either the original activity diagram is designed correctly or not. After the validation of the activity diagram confirmed, the process of verification starts, at the end of this stage five quality attributes in terms of the developers input will be submitted to the user. Verification process is based on the formulas derived in coloured Petri net formalism. They will be applied directly to find the quality values. The validation is based on the method in the computer tool that is discussed in chapter 4.

### **3.2.5 Developing the ADET**

The computer tool is provided called ADET. The computer tool helps to model the Activity Diagram to coloured Petri net and based on the input given the tool provides the result. The result shows two important things. First it validates the Activity Diagram to show its correctness. After the validation of Activity Diagram is confirmed, ADET has the capability to verify the activity Diagram to find out the five non functional parameters described before. Generally speaking the provided tool is responsible to evaluate and test the Activity Diagram which has been already mapped to coloured Petri Net in terms of validation and verification based on the input given through the developer of the system.

### **3.3 Case Study**

To show the process of simulation and to show the lifecycle of tokens a case study is selected. In selection of the case study some important things were considered. It is tried to choose a case study that is straightforward to understand. Besides the case study itself the related documents needed to understand the case study better is provided.

The case study is the Activity Diagram taken from the applying UML and patterns book [62] which is a most used reference for software engineering researchers and developers. It is a beneficial book in terms of object oriented analysis and design and the unified process. The Activity Diagram is mapped to the coloured Petri Net and

the coloured Petri Net through ADET is analyzed. The result will be validation and verification of AD.

### **3.4 System Requirement for ADET**

To run the computer tool some requirements is needed. In terms of hardware the minimum of 2 megabytes of memory is needed and in terms of operating system it can be run on any windows version, the tool is provided in visual basic version 6.

It has many facilities. One can draw any kind of colored Petri nets with it, and then based on the input the computer tool receives from the user, it compiles, validates and verifies the colored Petri net. The tool has got a good user interface, so the user can simply make relation between the software and himself. It has features make the tool straightforward to use like drag and drop the components or drawing the components of activity diagram simply. The other features the tool has are: the ability to save the drawing colored Petri net, the ability to delete the components, the ability to open any saved drawings, the ability to redraw and the ability to provide a new drawing framework.

### **3.5 Summary**

This chapter has explained the related research methodology. In this chapter the steps in methodology is discussed and the process of methodology is analyzed briefly. Selected case study is mentioned and also in terms of computer tool its requirement and its capabilities discussed briefly.

## **CHAPTER 4**

### **VERIFICATION AND VALIDATION MODEL FOR ACTIVITY DIAGRAM USING COLOURED PETRI NET**

#### **4.1 Introduction**

This chapter explains the methodology that will be used in formalizing activity diagram and finding non functional parameters. Also the concept of mapping from activity diagram to coloured Petri net will be introduced.

## 4.2 Investigation in Formalism of Coloured Petri Net

In this part we try to introduce the coloured Petri net and the strengths this modeling language has. Then the formulas that have been derived from CPN will be discussed.

### 4.2.1 Definition of the Coloured Petri Net

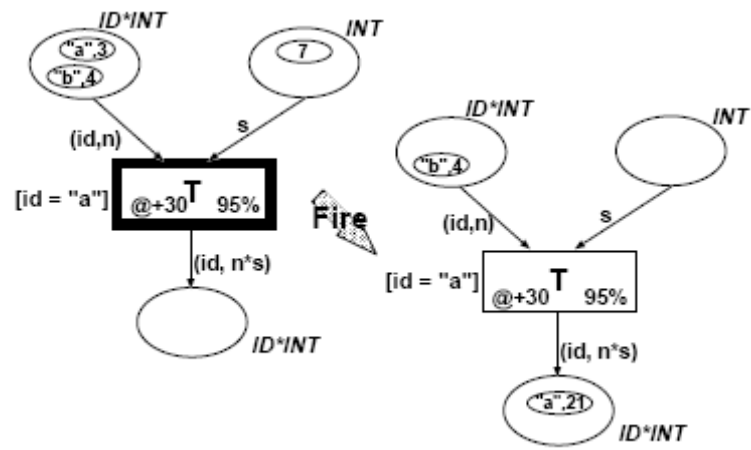
Coloured Petri Nets, proposed by K. Jensen, is an extended version of Petri Net. In addition to *places*, *transitions* and *tokens*, the concepts of *colors*, *guards* and *expressions* are introduced so that computed data values can be carried by the tokens. A transition in a CPN is able to fire (called *enabled*) if the following conditions hold:

1. Each of the places input to the transition has at least one token.
2. For the tokens in the input places, the expressions attached to the input arcs to the transition hold.
3. The guard (Boolean expression) attached to the transition holds.

Figure 4.1 illustrates a simple CPN. Ovals and rectangles stand for places and transitions respectively. Tokens put in a place are represented with smaller ovals in the places. The new concept *color* is similar to data type and specifies a set of acceptable data values. Places in a net can be typed, i.e. colors can be attached to the places. For example, suppose that the color INT is attached to a place. The tokens in the place cannot have data values except that their type is INT (integer). In the figure, the expressions “(id, n) and “s” are attached to the arc that flows to a transition. The former one stands for a pair of values (tuple type) where one is of ID type and the other one is

of INT, and these values are bounded to the variables “id” and “n” respectively. The expression  $[id = \text{“a”}]$  is called a guard of the transition. In this example, since it holds for the token  $(\text{“a”}, 3)$ , the transition is able to fire. After it fires, the token on the output arc has the value  $(\text{“a”}, 21)$ , which is the result of evaluating the expression  $(id, n*s)$ . We can attach time lags and probabilities of firing occurrences to a transition. The expression “@+30” means that it has 30 time-units until firing at a transition since the firing condition holds. We can also attach to a transition the probability of firing it. In the figure, the probability is 95% that firing at the transition T actually occurs if it is enabled, whereas the other enabled transitions, if any, fire in the probability 5%. By means of the attachment of probabilities, we can specify fault occurrence ratio in that tokens. For example, suppose that a new transition that has the same input places as T but that has no output places is added to the figure, and 5 % is attached as a firing ratio. In this case, we can result in the net whose fault occurrence ratio is 5%.

We should mention the reasons why we adopted not pure Petri Net but Coloured Petri Nets. The tokens in a place can have different values in a CPN. Thus we can distinguish and identify the tokens even if they are put in the same place. Readers will understand that we can distinct the token  $(\text{“a”}, 3)$  from  $(\text{“b”}, 4)$  in Figure 4.1. By attaching values to tokens, the occurrences of the tokens are distinctively identified. It is indispensable for describing software architectures.



**Figure 4.1:** An example of CPN

## 4.2.2 Characteristics of Coloured Petri Net

As it is mentioned in 3.2.1, Coloured Petri net has a vast capability in modeling but generally speaking CP-nets are used for three different - but closely related - purposes. First of all, a CP-net model is a *description* of the modelled system, and it can be used as a specification (of a system to be built) or as a presentation (of a system to be explained to other people, or ourselves). By creating a model we can investigate a new system before we construct it. This is an obvious advantage, in particular for systems where design errors may jeopardise security or be expensive to correct. Secondly, the behaviour of a CPN model can be *analysed*, either by means of simulation (which is equivalent to program execution and program debugging) or by means of more formal analysis methods (which are equivalent to program verification). Finally, it should be understood that the process of creating the description and performing the analysis usually gives the modeller a dramatically improved *understanding* of the modelled system - and it is often the case that this is more valid than the description and the analysis results themselves. Below, a brief description of some of the main qualities of CP-nets is given.

1. CP-nets have a graphical representation.
2. CP-nets have a well-defined semantics which unambiguously defines the behaviour of each CP-net.
3. CP-nets are very general and can be used to describe a large variety of different systems.
4. CP-nets have very few, but powerful, primitives.
5. CP-nets have an explicit description of both states and actions.
6. CP-nets have a semantics which builds upon true concurrency, instead of interleaving.
7. CP-nets offer hierarchical descriptions.
8. CP-nets integrate the description of control and synchronisation with the description of data manipulation.
9. CP-nets can be extended with a time concept.
10. CP-nets are stable towards minor changes of the modelled system.

11. CP-nets offer interactive simulations where the results are presented directly on the CPN diagram.

12. CP-nets have a number of formal analysis methods by which properties of CP-nets can be proved.

13. CP-nets have computer tools supporting their drawing, simulation and formal analysis.

With paying attention to the rule above specially rule1, rule2, rule12 and the definitions up to now we can conclude that “if we are able to derive non functional attributes from CP nets and then if we can map the activity diagram to the CP net we will be able to derive the non functional parameters of activity diagram”.

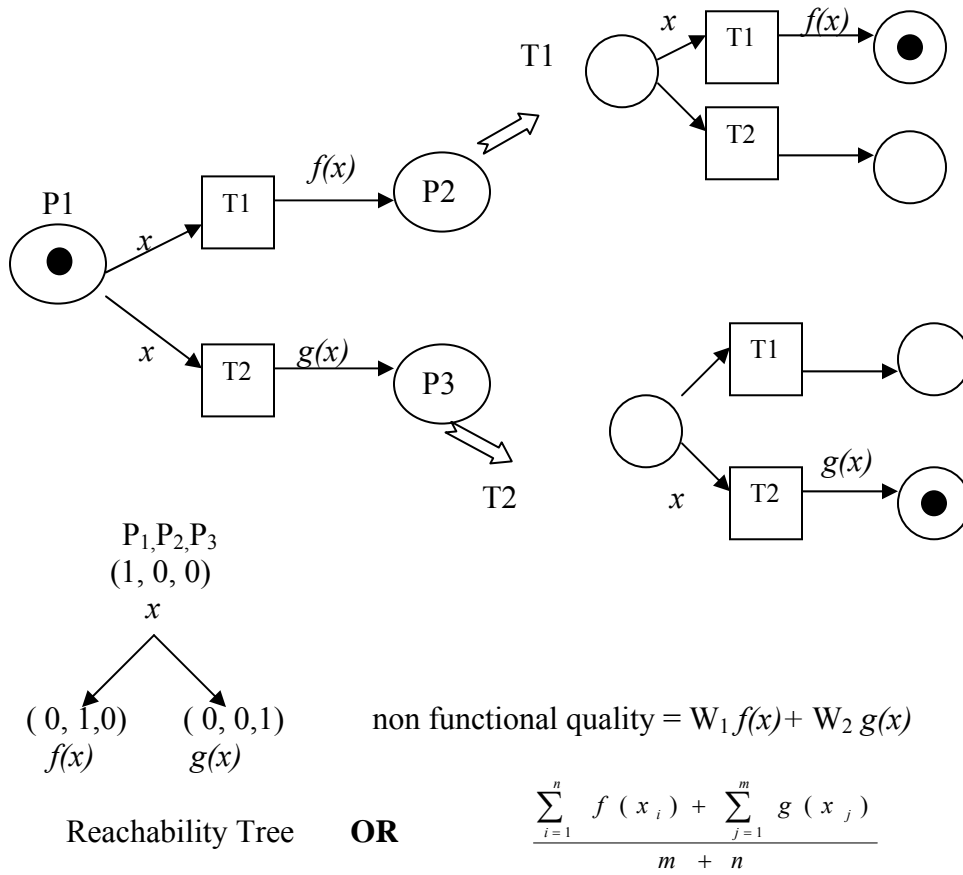
### 4.3 Investigation of Non Functional Quality Attributes with CPN

In this part first we try to see how we can formalize CPN to find out non functional parameters. Then we apply the way we can formalize CPN in terms of quality attributes to evaluate some of them. Reliability, security on the network, security on memories and files, performance and resource efficiency are the parameters, we will find quality attributes for them.

We want to see how we can find out quality attributes using cpn. One of the ways to specify non functional quality attributes in CPN is to define time lag parameters and the probability of firing as it was mentioned before in 3.2.1. However we use attach values to tokens to specify and evaluate various kind of non functional quality attributes. Here execution time and the probability of fault occurrences as nonfunctional aspects are chosen. Then their values will be calculated during execution of CPN that is sequence of firing. The result of calculations stands for the evaluation results of non functional aspect. For each execution of CPN the calculation is performed. Figure 3.2 shows how the method is going to be applied. In the figure, the value  $x$  on the token denotes the value of a nonfunctional aspect. There are two possibilities on firing; one is the firing at the transition  $t1$  and the other is at  $t2$ . After firing at the transition  $t1$ , the value is updated to  $f(x)$  and it means that the value of the non-functional aspect is newly calculated to  $f(x)$  by one-step progress of the execution.

Suppose that the value expresses time efficiency, i.e. execution time and  $f(x) = x+30$ . The value of  $x$  stands for time expiration after the system starts and 30 units time is spent in firing at  $t1$ , i.e. progressing the execution from  $p1$  to  $p2$ . Either firing at  $t1$  or at  $t2$  really occurs in non-deterministic way.

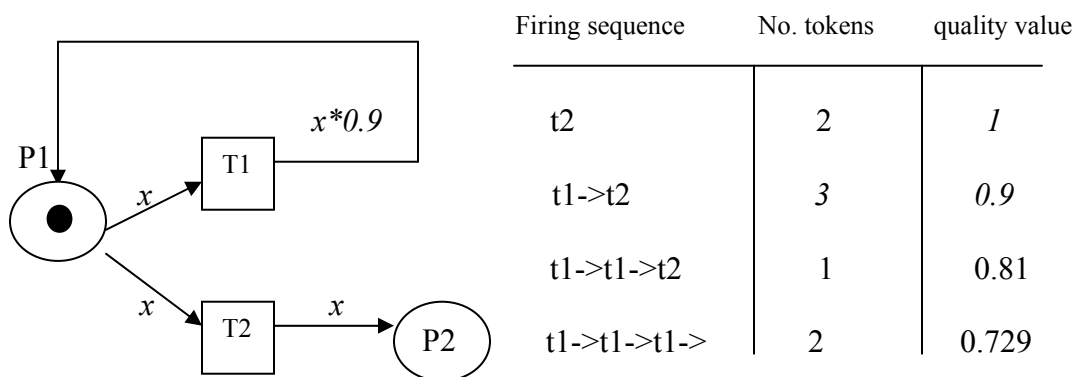
If there are some priorities on these alternatives, we can control and simulate the execution of the CPN by providing a probability for each alternative. suppose that the firing at  $t_1$  and  $t_2$  occur in the probabilities  $w_1$  and  $w_2$  respectively ( $w_1+w_2 = 1$ ). We can have  $w_1 * f(x) + w_2 * g(x)$  as a resulting value of the quality attributes of the CPN. If we do not specify the probabilities  $w_1$  and  $w_2$ . We repeat the execution of the CPN and calculate the additive average of the resulting values, each of which is calculated at one execution. In the figure 4 it is assumed that CPN executes the firing at  $t_1$  at  $n$  times repeatedly and the firing at  $t_2$  at  $m$  times. Let  $x_i$  be the quality value before firing at  $t_1$  in the  $i$ -th repetition of  $n$  times.  $X_j$  is for the firing at  $t_2$ . In this case, it calculates as the quality value of the CPN. Thus specifying  $f$  and  $g$  expresses defining non-functional quality as calculation methods. this aspect with the equivalent equations are in figure 4.2.



**Figure 4.2:** Determining quality values

### 4.3.1 An Example for Finding the Quality Value, Nondeterministically

Suppose we have a CPN like in figure 4.3. In this CPN we have two transitions,  $t_1$  and  $t_2$ .  $T_1$  resembles a transition that may lost data during its transmission. The failure rate of  $t_1$  is 0.1 and thus the success rate is 0.9. In reality 0.9 is a low number and let's say transition  $t_1$  has the low quality of transmitting data. In CPN firing occurs nondeterministically. So we need to execute CPN to see what value the token may get through transmission. Success rate is the probability of the transition so obviously we may multiply  $x$  by 0.9 as a result of one transmission. In the following figure some sort of firing and the number of tokens and their quality value are given. Example concludes nondeterminism. It means that which  $t_1$  or  $t_2$  fires are nondeterministically decided. the amount then is evaluated through the following formula [63].

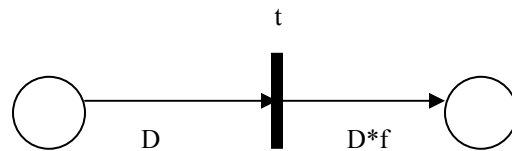


$$\frac{\sum_{i=1}^2 1 + \sum_{i=1}^3 0.9 + \sum_{i=1}^3 0.81 + \sum_{i=1}^3 0.729}{2 + 3 + 1 + 2} = 0.871$$

**Figure 4.3:** An example of calculating quality values

### 4.3.2 Reliability

For calculating reliability of the system we may define a success rate,  $f$ , for the transition in CPN. Success rate specifies the probability of firing transition if something wrong does not happen, on the other hand  $1-f$  is the failure rate, the probability of data loss. The token in the input of the transition  $t$  is assumed that carries the amount which stands for the accumulation of the success rate up to that place. When transition fires the amounts which represents the success rate will be changed to  $D*f$ , i.e. the token has got a new probability of firing;  $D*f$  instead of  $D$ . figure 4.4 shows the concept.

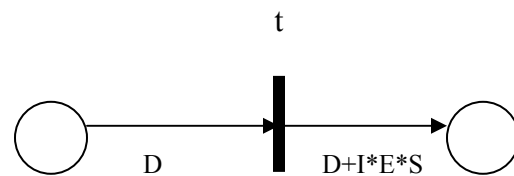


$D$ : accumulation of success rate  
 Reliability =  $D$

**Figure 4.4:** Calculation of reliability

### 4.3.3 Security on Network

The security related to networks can be considered as the elaborated version of the reliability as shown in Figure 4.5. The security is calculated in the figure through transition  $t$ . Malicious persons can copy data at the transition  $t$  and the relevant security value is calculated when the transition  $t$  fires. Then security value results from the quality of the used cryptography, the importance of data through the transition and the easiness of wiretapping at  $t$ . The security value is the sum of the values at the transitions where data are likely to be copied by the outside entities. The values  $E$ ,  $I$  and  $S$  should be given and normalized by the developer that evaluates the security.

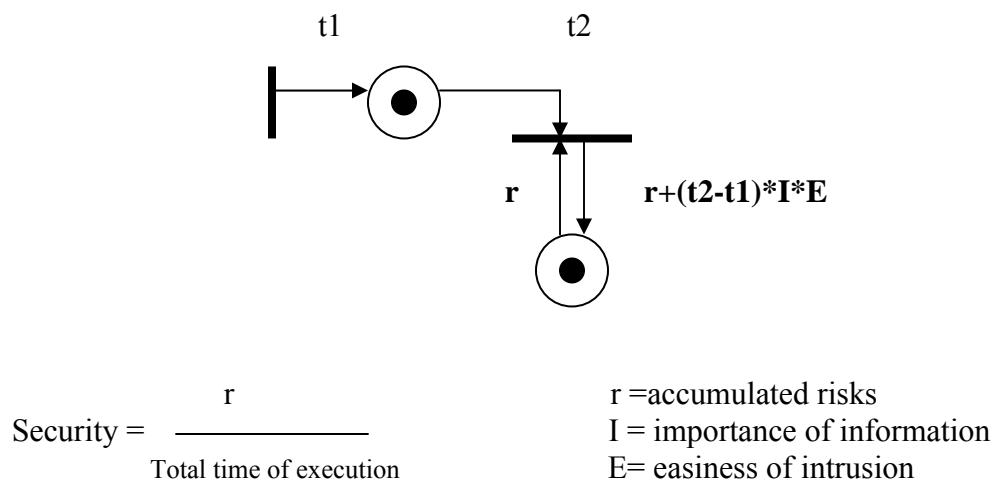


I: importance of information  
 E: weakness of the cryptography  
 S: easiness of wiretapping

**Figure 4.5:** Security on the network

#### 4.3.4 Security on Memories and Files

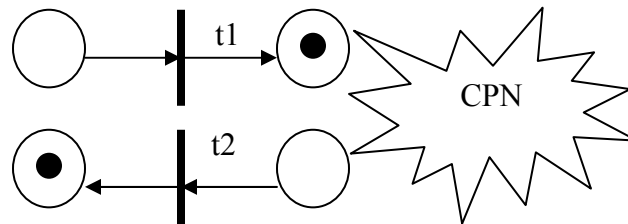
As for security on memories and on files, we should consider the time period when data are put in the memory or in the file, as well as the numerical values for security on network, because the time period when data exist in them is much longer than on network. Figure 8 shows how to quantify security based on time duration of data in memories and files. Thus we multiply the time period  $t2 - t1$  where  $t2$  and  $t1$  are time when the token is input to and time when it is output from the place  $p1$  respectively. The place  $p1$  stands for the component corresponding to the memory or file whose quality we want to evaluate. Thus we can know easily the time when the token is in ( $t1$ ) and the time when it is out of a place ( $t2$ ). The total time of the execution can be easily calculated by the time stamp that attaches on the token. Figure 4.6 shows the concept.



**Figure 4.6:** Security on memories and files

### 4.3.5 Time Efficiency

By using the time stamp of a token, as shown in Figure 3.7, we can easily calculate how long it takes in firing and moving to a destination place. Suppose that we want to evaluate the performance, i.e. the execution time from the place  $p_1$  to  $p_2$ . We just refer to the time stamps and calculate  $t_2 - t_1$  to get the time efficiency of the CPN whose places are corresponding to  $p_1$  and  $p_2$ . Figure 4.7 shows the concept.

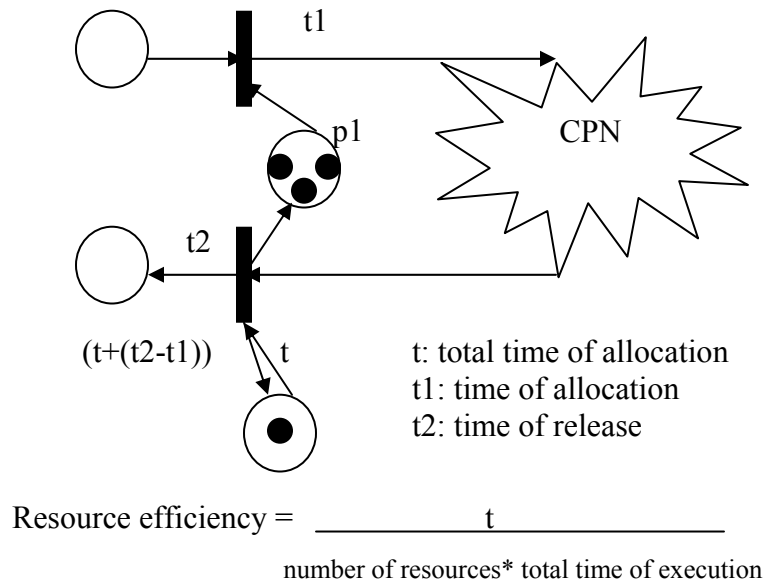


$$\text{Time efficiency} = t_2 - t_1$$

**Figure 4.7:** Time efficiency

### 4.3.6 Resource Efficiency

Resource efficiency expresses during how many percentages of the total execution time the resource is allocated to be used. The place  $p1$  in Figure 10 corresponds to the component of a resource and the tokens in  $p1$  represent the status of its allocation. If a token comes out of  $p1$ , the resource is being allocated to an executing process. The resource release means that the token comes back in to the place  $p1$ . We can calculate how long the resource is allocated by referring to the time stamps  $t1$  and  $t2$ , which denote the allocation time and the release one respectively. Figure 4.8 shows the concept



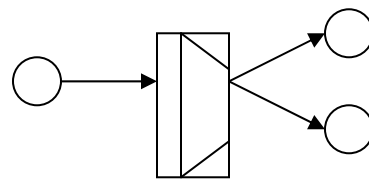
**Figure 4.8:** Resource efficiency

After finding and investigating the formalism of CPN, by converting activity diagram to CPN we will be able to verify it. The process of verification says how much our artifact meets the requirements that have been proposed before. In terms of activity diagram verification means how much our activity diagram has the non functional parameters defined before. After verification the process of validation starts that indicates either our verified activity diagram is right or wrong.

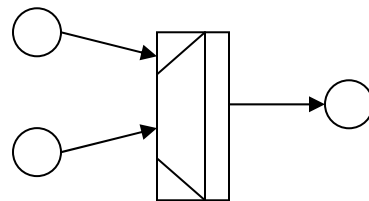
#### **4.4 Developing Transformation Algorithm**

For developing the transformation algorithm, an approach is may be to find and map the activities of an activity diagram to CPN, based on their relationship to software architectural component, in the activity diagram, each activity is defined to have a task. It means that, it calculates, performs or does something. So it is logic to map activities in activity diagram to transitions in CPN. A start place in the activity diagram on the other hand can be mapped to an initial transition that is responsible to fire tokens to the system based on some time intervals. These tokens will be saved inside a place. This place can be the incoming place of a transition. Each out coming place can be an incoming place for the next transition. Each activity can be mapped to a transition with an incoming place to buffer the tokens that need the transition and an out coming place for saving the calculated tokens through the transition. The process of fork, join, decision and merge are shown in our mapping through four new symbols. The symbols are AND-SPLIT, AND-JOIN, OR SPLIT and OR-JOIN respectively.

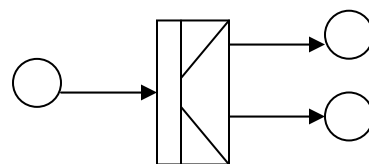
OR-JOIN, OR-SPLIT and AND-JOIN, AND-SPLIT that are a kind of transition. To show a point of path that is branched and only one way should be selected we used OR-SPLIT, and to show a point that includes some entries and only one of them should be selected we use OR-JOIN. To show a point that some flows it parallel and simultaneously, we use ANDSPLIT, and to show point that some flows enter it, parallel and simultaneously, we use AND-JOIN. Comparing to the activity diagram OR-SPLIT is decision, OR- JOIN is merge. AND-SPLIT is fork and AND-JOIN is join. The concept is depicted in figure 4.9



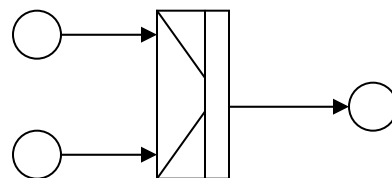
Or-split



Or-join



And-split

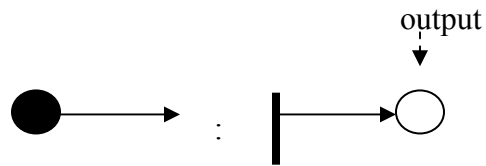


And-join

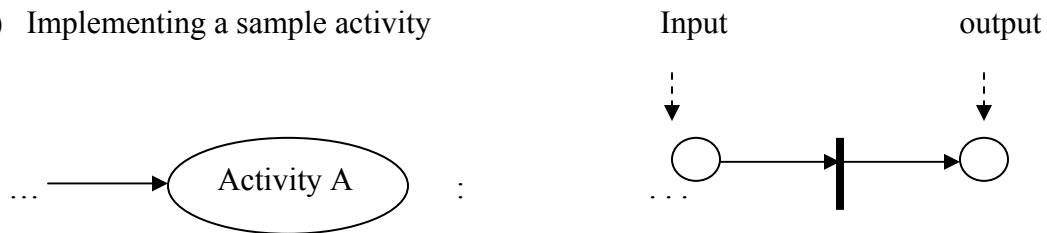
**Figure 4.9:** Additional shapes

In the following steps needed to convert activity diagram to CPN in different conditions is provided:

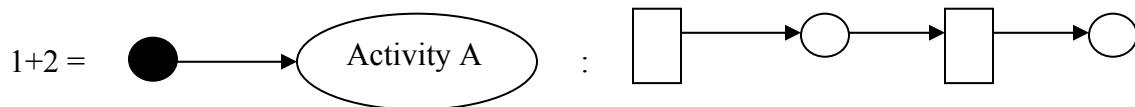
1) Implementing initial state of activity diagram



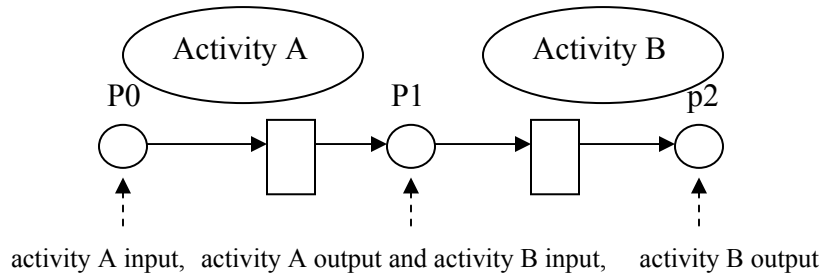
2) Implementing a sample activity



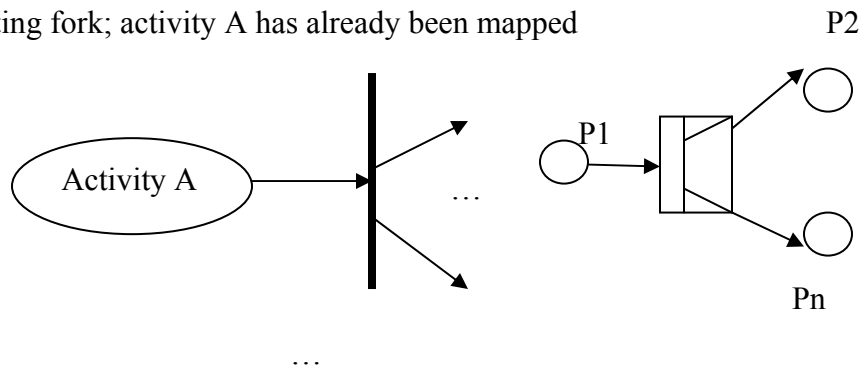
To map a new component to the currently mapped component, we use the output of the currently mapped component to be the input of the new component, i.e assume that (1) is mapped and we want to map (2):



## 3) Adding activity B to the currently mapped activity A

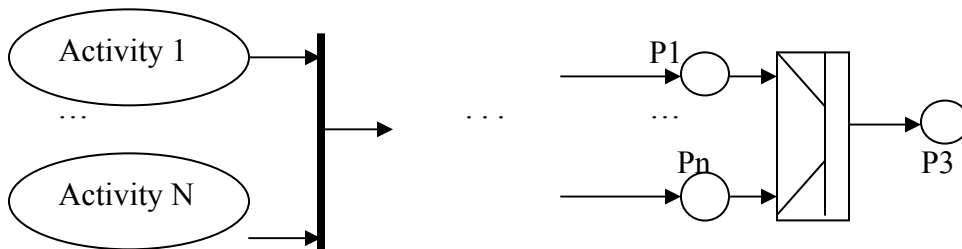


## 4) Implementing fork; activity A has already been mapped



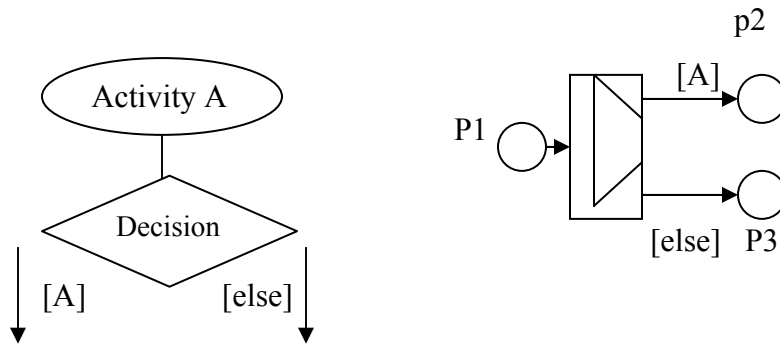
P1 = Output of activity A, P2 =fork out put, Pn =fork out put

## 5) Implementing join; activity B and C has already been mapped



P1 = output of activity 1, Pn = Output of activity N, P3= join output

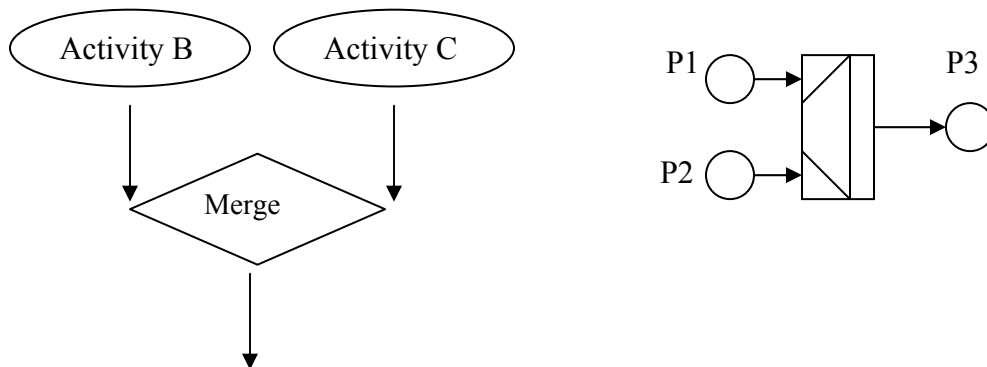
6) Implementing decision; activity A has already been mapped



P1=activity A output, P2=token position after t1 fires, P3= token position after t2 fires  
one of the transitions t1 or t2 can fire, not both

---

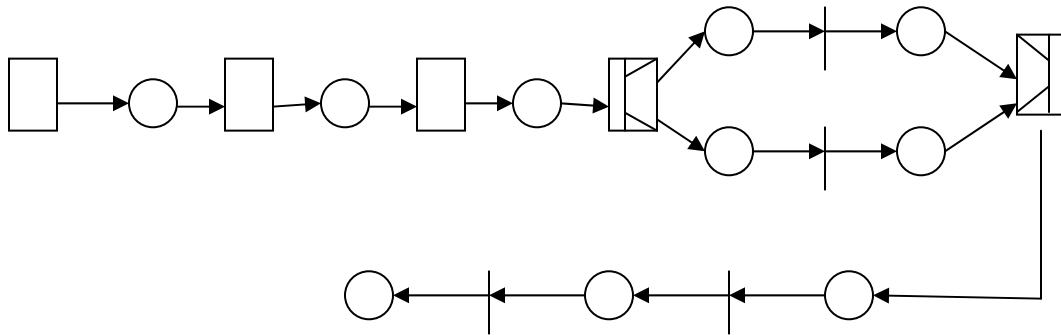
7) Implementing merge; activity B and C has already been mapped



P1 = Activity B output , P2= Activity C output , P3= merge output

---


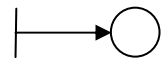



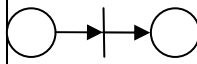

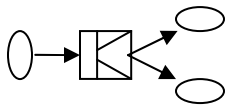

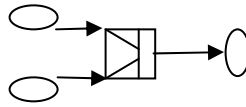

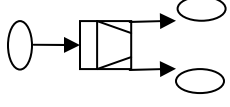
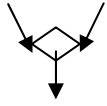
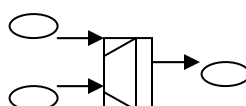
As an example the activity diagram in figure 2.2 can be converted to CPN as follow in figure 4.10:



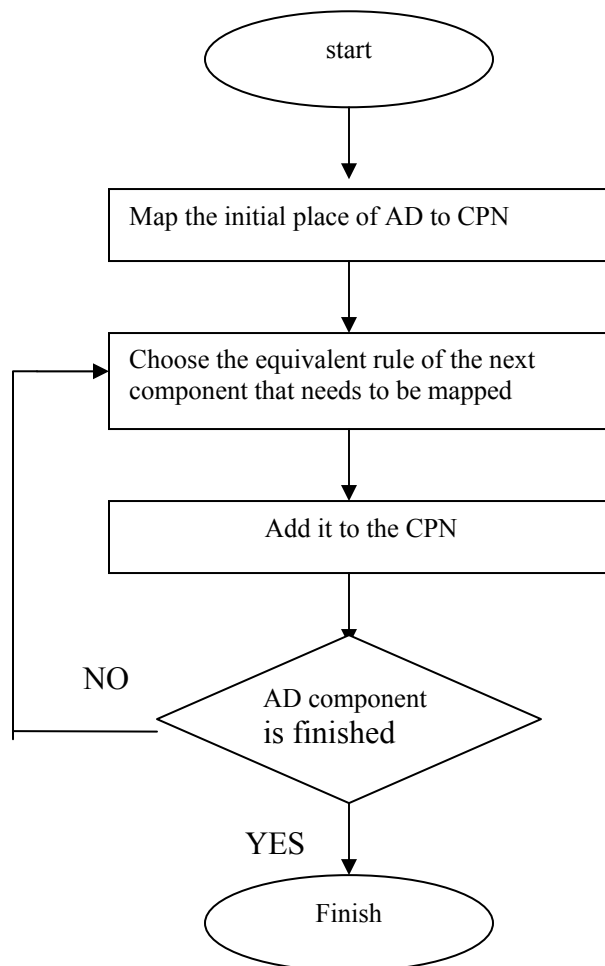
**Figure 4.10:** CPN for activity diagram figure 2.2

In the table 4.1 the mapping from an activity diagram to the coloured Petri net is provided. According to the rule of CPN we are not permitted to use two activities (task) or two places after each other. Between the same two components there should be a different component. The final place of an activity diagram is the output place of the last transition.

**Table 4.1:** Mapping table from activity diagram to CPN

Activity Diagram	concept	Coloured Petri Net
	Start place	
	Final place	
	Activity	
	fork	
	join	
	decision	
	merge	

The following algorithm (figure 4.11) is proposed for mapping from Activity Diagram to CPN:



**Figure 4.11:** Flowcharts for mapping from activity diagram to CPN

It should be noticed that there is a difference with merge and join; in merge ANY input leads to continuation. This is in contrast to a join, in which case all the inputs have to arrive before it continues.

#### **4.5 Verification of Activity Diagram**

Verification in software engineering means to see how much the artifact has the specified requirements or how much it can meet the requirements that have been introduced. Here we have defined five nonfunctional parameters. Based on the tools provided these non functional parameters are calculated and the final amount will be produced. We can say that verification means either we produce a product right or not, or either it meets the requirement we have defined or not. Security on memories and files, time efficiency and resource efficiency are calculated by the tools by simply adding a new concept to the token i.e. time stamp. Each token has an attached time stamp that makes the tools to calculate the mentioned parameters. Verification of activity diagram tells us how much the activity diagram meets the non functional parameters we defined before. The result of verification gives the developer this point of view that how he can enhance the diagram to be more reliable, secure and ... this yields reconsideration of requirements, methods and parameters of the software system. In a spiral model like RUP the requirements in the analysis part and methods and parameters in the design part will change in each iteration. So the benefit of verification is for analyst, designers and coders.

## 4.6 Validation of Activity Diagram

Validation in software engineering means how correct the final artifact is or we can say that validation means either we produce a right product or not. In validation we should take care of OR-JOIN, OR-SPLIT and AND-JOIN, AND-SPLIT. When we use an OR-SPLIT, we should use also an OR-JOIN also if we use an AND-SPLIT, we should use an AND-JOIN too. In other words, the number of equal components should be same logically in order that the diagram follows mathematically rules in iteration. The algorithm for validating activity diagram is to show either we have the balance among components (OR-JOIN, OR-SPLIT and AND-JOIN, AND-SPLIT) or not. No balance means that the original activity diagram was not drawn correctly and the designer should reconsider in drawing the activity diagram. The provided tools perform validation by attaching a stack to the token, for each (OR-JOIN and OR-SPLIT) or (AND-JOIN, AND-SPLIT) we define a special byte. As the token moves in the CPN the correspondent byte will be pushed and popped in and from the stack, if at the end place of coloured Petri net the stack is not empty, the validation process yields invalid result. If the token in the middle of its way encounters a problem to pop the correspondent byte, an error occurs that again yields the invalid result. Invalid result means that the activity diagram is not drawn correctly and the designer should reconsider drawing the activity diagram.

The process of validation shows that either the coloured Petri net that resembles the activity diagram is correct or not, if the coloured Petri net is wrong it yields that the original activity diagram is incorrect and the designer should reconsider drawing the activity diagram. The process of validation is done via the tools by simulating the system; it means that we insert some tokens in the system. Each token has a stack. The algorithm of validation is as follow.

1. When token moves through the system, as it passes an AND-SPLIT transition the correspondent byte (like 2) is pushed in the stack.
2. As the token continuous moving, if it passes OR-SPLIT transition the correspondent byte (like 4) is pushed in the stack.
3. As the token continuous moving, if it passes AND-JOIN transition an amount is popped from the top of the stack, if the amount is not equal the AND-SPLIT amount, validation process returns invalid result.
4. As the token continuous moving, if it passes OR-JOIN transition an amount is popped from the top of the stack, if the amount is not equal the OR-SPLIT amount, validation process returns invalid result.
5. At last when the token arrives the final place in Petri net, if its stack is not empty, the validation process returns invalid result.
6. At last when the token arrives the final place in Petri net, if its stack is empty, the validation process returns valid result.

#### **4.7 Summary**

Based on the previous researches and works we can not verify the current activity diagram in terms of non functional parameters. Furthermore there is no straightforward algorithm to assess the validation of the artifact. This project will concentrate on converting the current activity diagram to CPN. The related methodologies involved in this project have been described above.

## **CHAPTER 5**

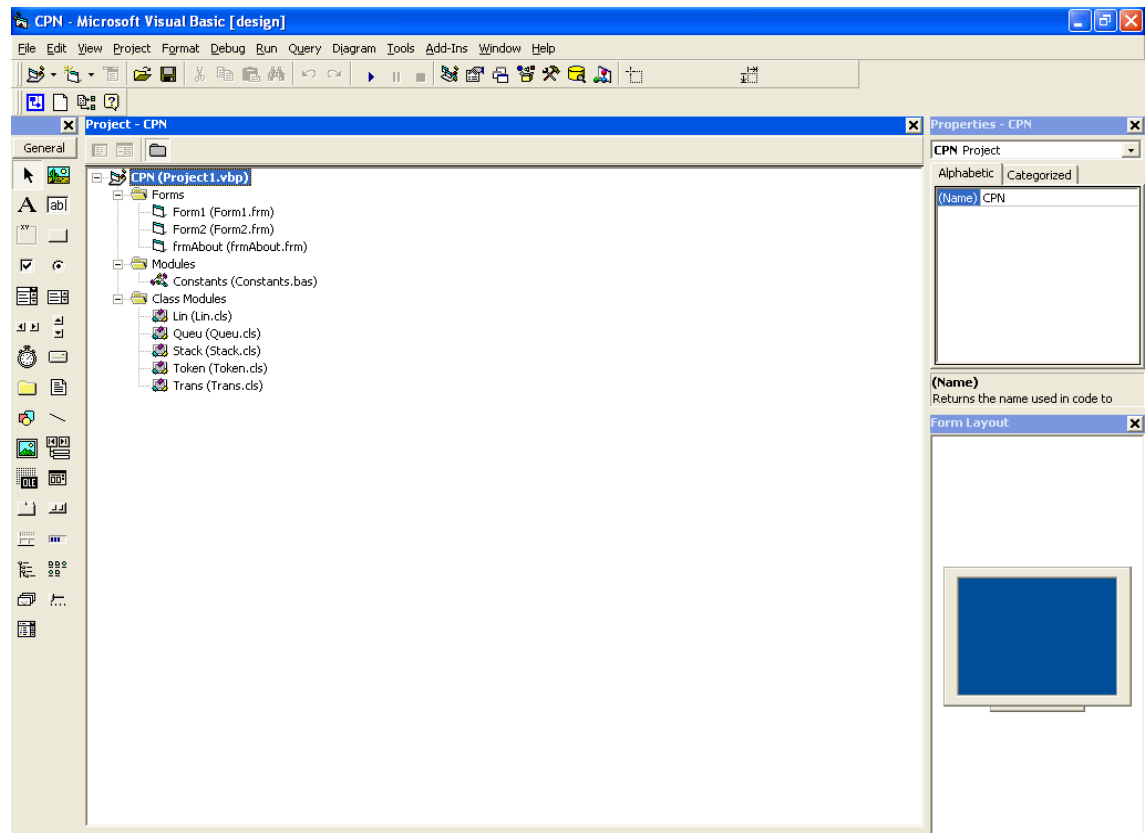
### **ADET DEVELOPMENT FOR VALIDATION AND VERIFICATION OF AD**

#### **5.1 Introduction**

This chapter focuses on the architecture of the tool, its capabilities, functionalities and features, moreover example of the tool operation is provided in the form of screens.

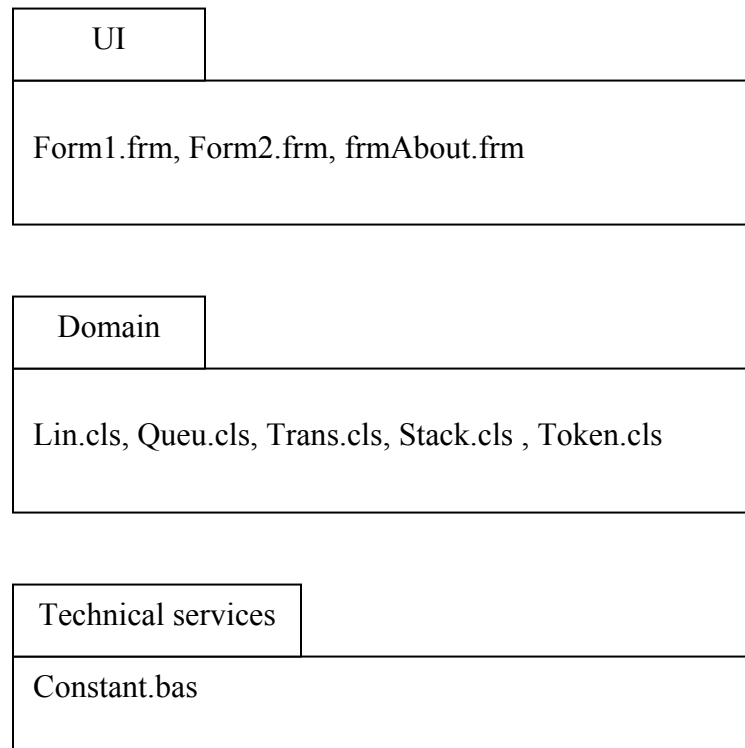
## 5.2 ADET Architecture

The computer tool is provided in visual basic version 6 platform. The tool has three forms, one module and five class modules. Five class modules are lin.cls, queu.cls, stack.cls, token.cls and trans.cls. lin.cls is for drawing arcs, queu.cls is for drawing places. Stack.cls is for simulating stacks, token .cls is for presenting tokens and finally trans.cls is responsible for drawing transitions. Stack.cls is used for the process of validation. Each token has an stack, so they can trace the number of special transitions in the system. Forms are used to provide interface with users. The schema of the tool architecture is in figure 5.1



**Figure 5.1:** Architecture of the ADET

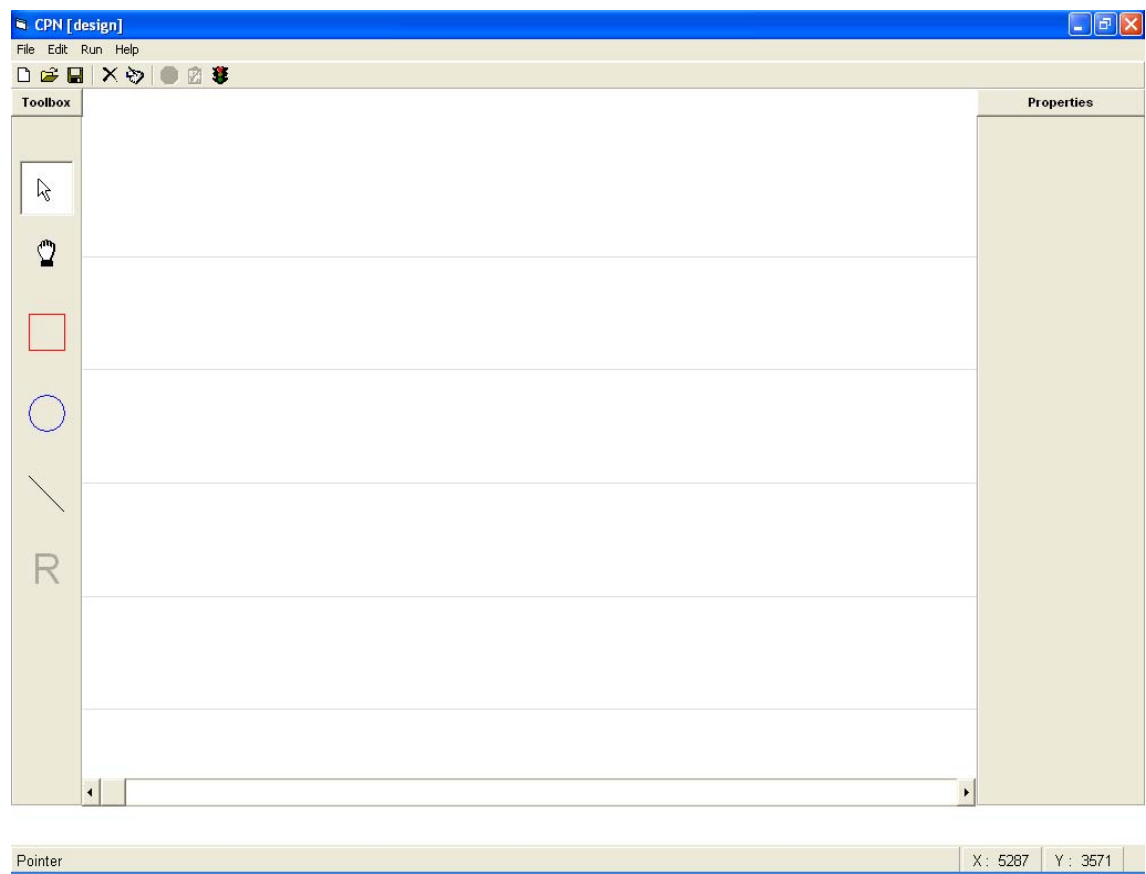
Also the logical architecture of ADET is given in figure 5.2, in the logical architecture of ADET, user interface components, domain components, and technical services components are given.



**Figure 5.2:** Logical architecture of ADET

### 5.3 ADET Functionalities

The provided tool has a vast functionalities and capabilities. By providing a good user interface, the user can make relation with the tool simply. To learn using the tool is straightforward so one can learn it easily. It has capabilities of saving, redrawing, opening the currently saved document, dragging and dropping the components of coloured Petri net, deleting the currently drawn components, switching from the report mode to the design mode, compiling, calculating the validation and finally verification of the Activity Diagram. The transitions and the places both distinguishable with their shapes and with their color. Transitions are red squares and places are blue circles. Drawing an arch is simply provided. Just by clicking the line components on the left toolbox and clicking on the source and destination components on the screen, the arch will be drawn. It is tried to provide a good graphical environment too. Through using proper icons the process of running and evaluation becomes faster. The properties side on the right part of the tool shows the properties of any componets (transitions or places). The user or the developer of the system simply by inserting the amount mentioned in the properties can give input to the system. In the properties part “**activites**” and “**architecture**” should be fulfilled by the user for each of the places and transitions. Likewise the transitions time lags should be given by the user. The tool is constructed in a way that statistics of the coloured Petri net up to each component is reachable just by clicking on the desired component, but the final result is in the final place where all the tokens may reach there. As it is mentioned the tool has the capability to switch between the report mode and the design mode. It means that while the user watches the result by switching to the design mode can manipulate the inputs and assess the new result through compiling, validating and verification. The tool interface is provided in figure 5.3. ADET program can support all kind of object oriented software, produced through RUP either critical, embedded or real time. Through ADET the activity diagram of these kinds of software can be analyzed, the quality parameter can be extracted or even the software can be tested after its final cycle in RUP as a final product.



**Figure 5.3:** The ADET Interface

## 5.4 Example of Operations

In this section the example of operation of a sample Petri net is given. In figure 5.4, user drew the coloured Petri net using the mapping table 4.1 for a relevant dream activity diagram and the inputs for the places in terms of I and E are as follow:

$$q_0=(0,0), q_1=(0,0), q_2=(1,1), q_3=(0,0), q_4=(1,2), q_5=(0,0), q_6=(0,0)$$

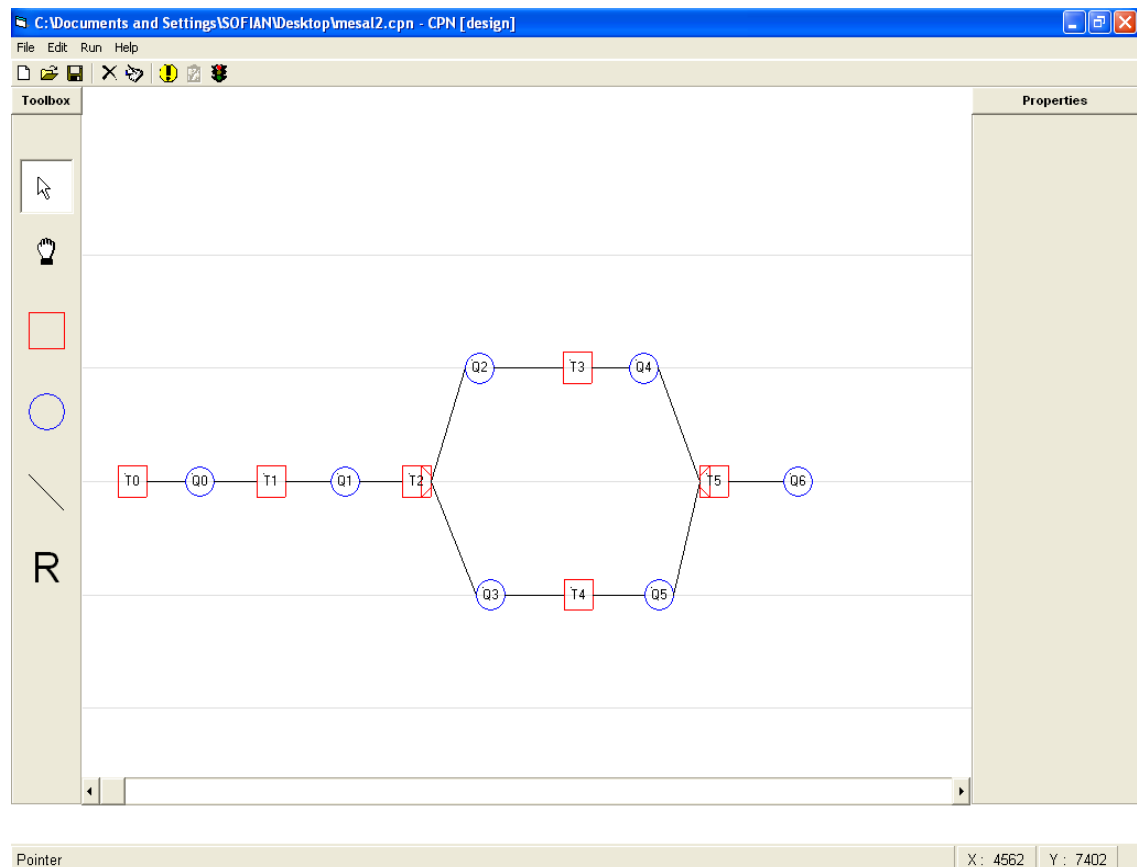
Likewise suppose the input for transitions in terms of failure ratio, I, E are as follow:

$$t_0=(0,0,0,0), t_1=(0,0,0,0), t_2=(4,1,1,2), t_3=(5,2,1,1), t_4=(1,0,1,2), t_5=(3,0,1,2)$$

and the transition times are:

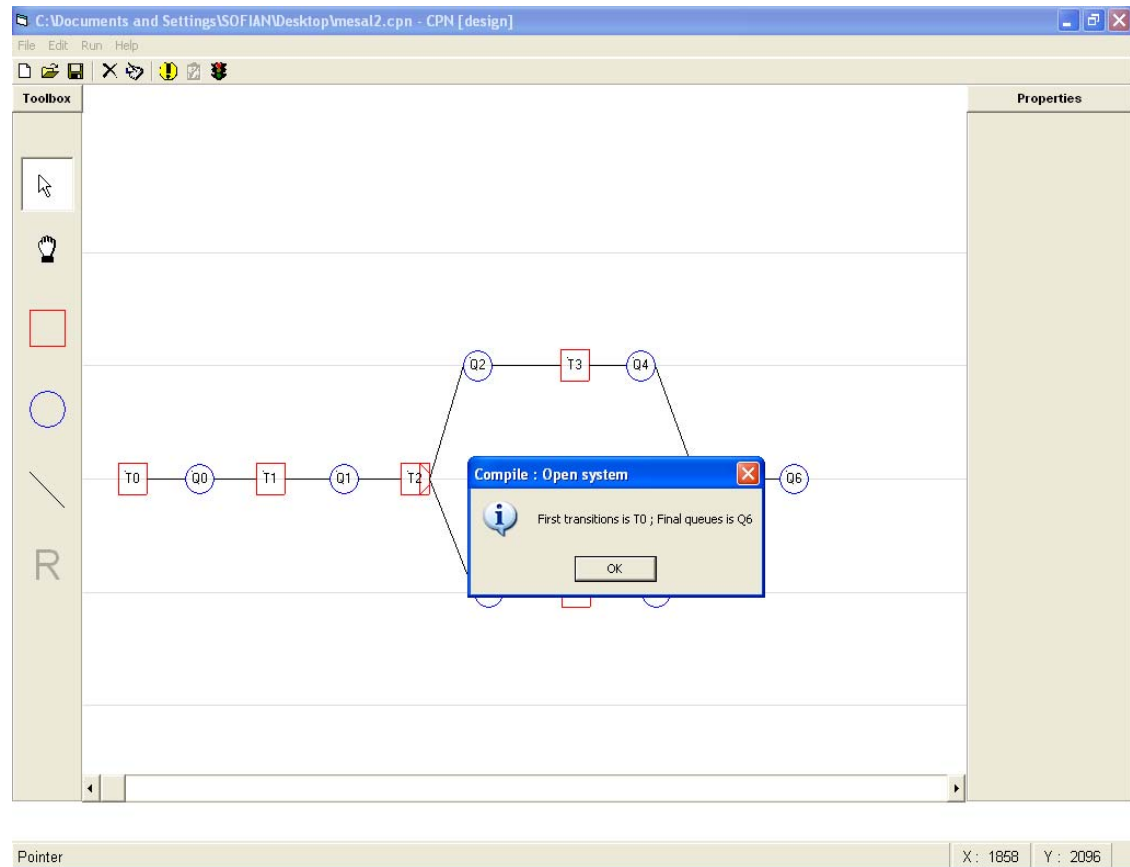
$$t_0=(10,15), t_1=(9,14), t_2=(8,13), t_3=(10,15), t_4=(9,15), t_5=(7,12)$$

Then the tool will compile, validate and verify the coloured Petri net.

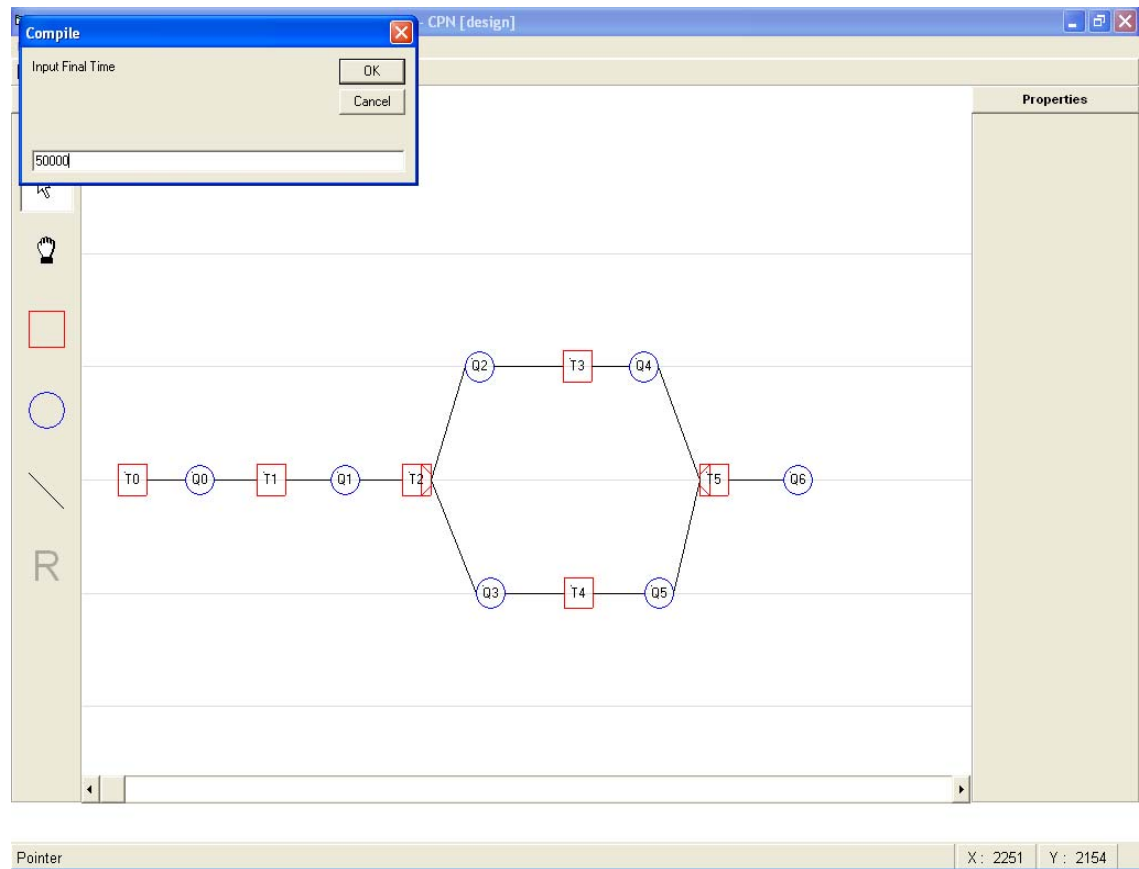


**Figure 5.4:** Sample coloured Petri net

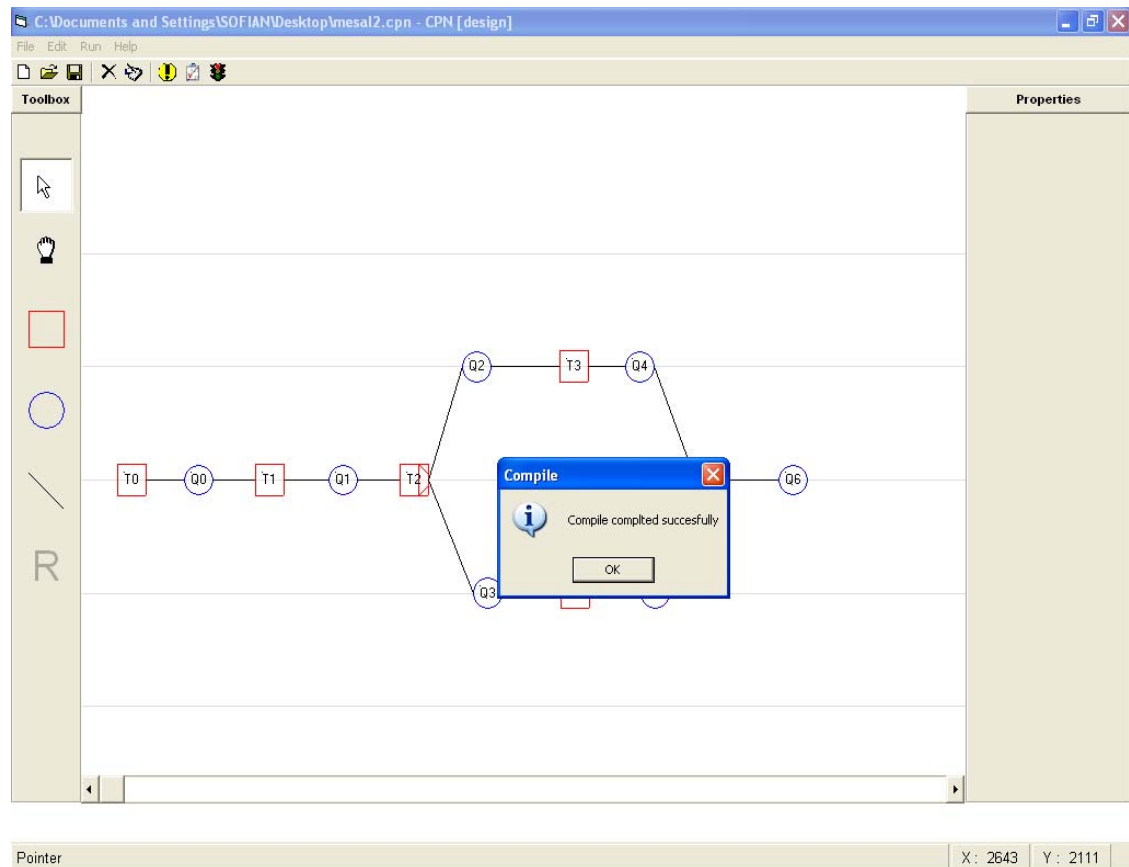
The process of compiling itself contains three parts, first the tool asks about the first transition and the last place, then the tool asks user to enter the input final time. Through this time tool will the number of tokens into the system. Assume that this is entered 50000 by the user. And at last tool reports successfully compiling. The first part of the compiling is in figure 5.5 and the second part is in figure 5.6 and the third part is in figure 5.7.



**Figure 5.5:** First part of compiling

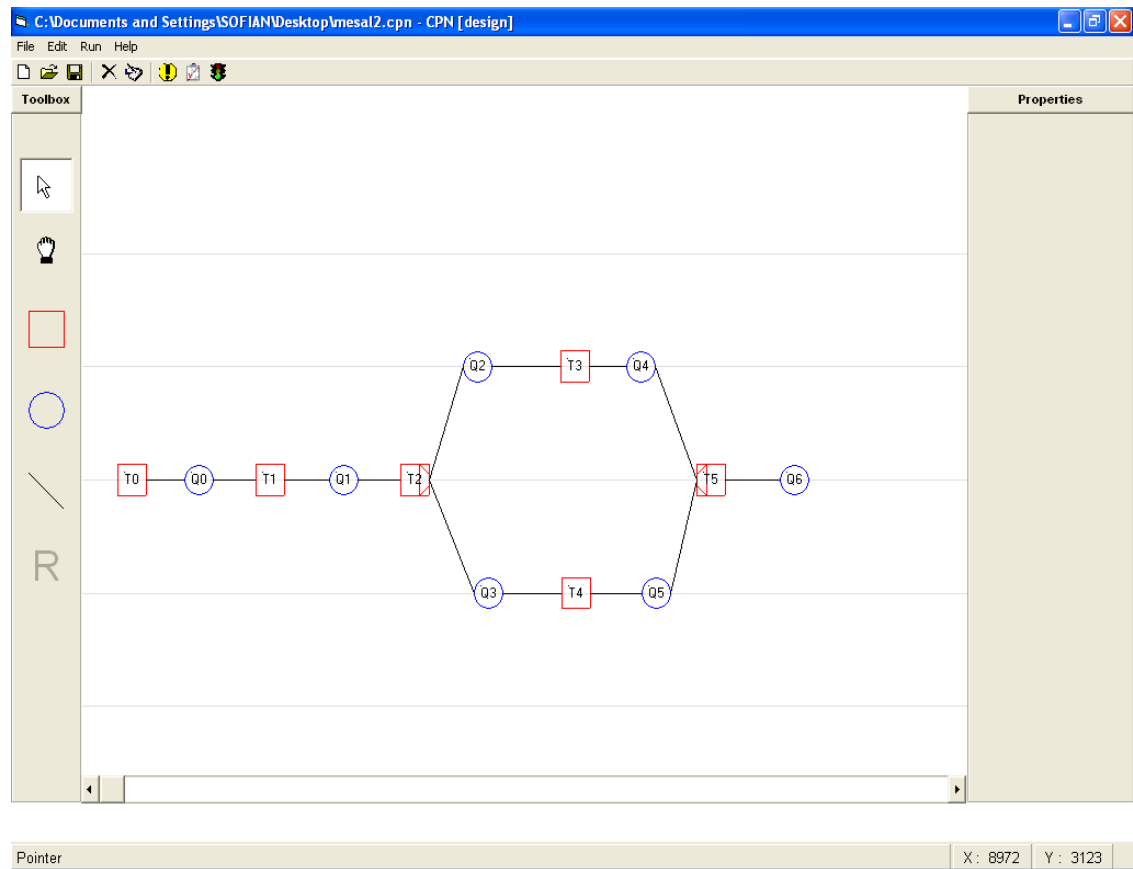


**Figure 5.6:** Second part of compiling



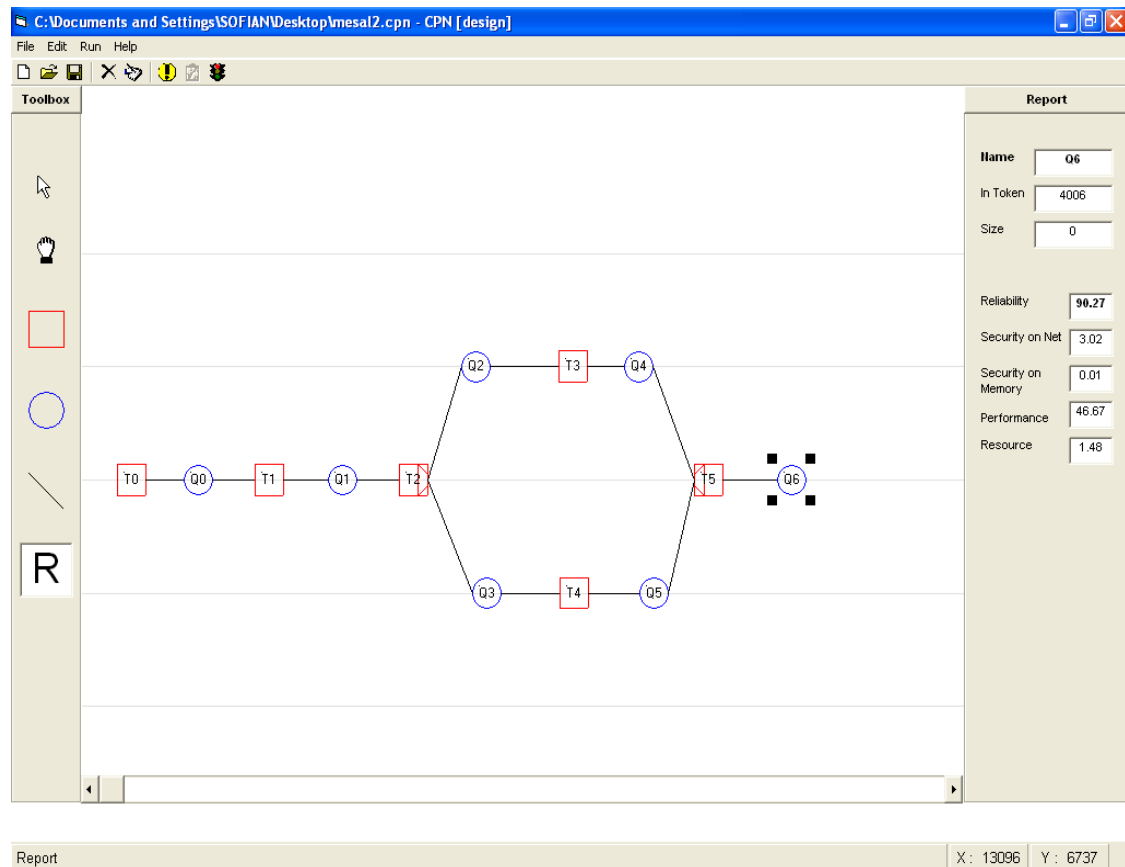
**Figure 5.7:** Third part of compiling

In validation part, by pressing F7 or the related icon, the tool will validate the coloured Petri net. If the result is true then the traffic light's green light will be on otherwise the error will be reported on the screen, in our case the coloured Petri net's validation's result is true so no error will be reported by the tool. The correspondent figure is figure 5.8



**Figure 5.8:** Validation

And finally by pressing F5 or related icon, the verification of coloured Petri net will be done, at the end of this stage five quality values will be submitted to the user. Simply by clicking the last place of coloured Petri net, the user can see the final result of verification. The process of verification is depicted in figure 5.9



**Figure 5.9: Verification**

As it is obvious in figure 5.9 after the verification process in the last place of the cpn contains total statistics about the entire system. At the end of this stage ADET submits five nonfunctional parameters to the user which are reliability, security on the network, security on memory, performance and resource efficiency. Moreover ADET submits more information to the user besides the five mentioned nonfunctional parameters. Three additional information are name, in token and size. Name describes which component we are analyzing right now. In token shows the number of tokens that the component which has been already identified by name has received. Size shows the number of resident tokens in the component after simulation finished.

## **5.5 Summary**

In this chapter the tool architecture, functionalities and operations are shown. The architecture of the tool in terms of used modules as well as tool's functionalities in terms of being user interface and its capabilities was discussed. Finally the operation of the tool is shown step by step with assumed input through related figures.

## **CHAPTER 6**

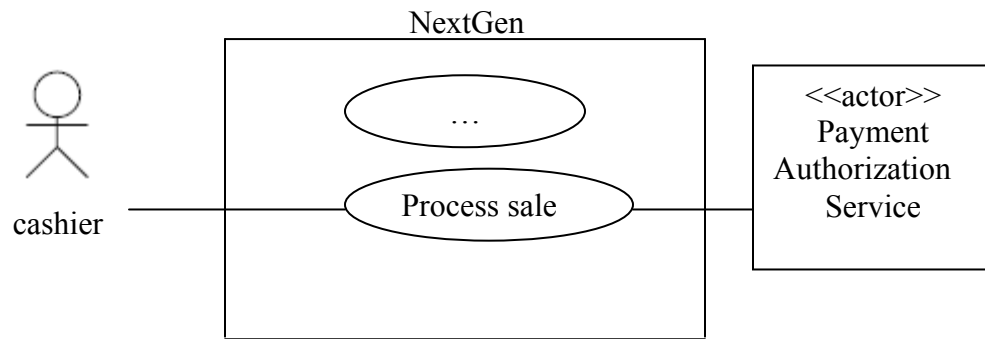
### **RESULT AND DISCUSSIONS**

#### **6.1 Introduction**

Of course after providing the transformation algorithm it should be assessed in different features; like the data that are going to be used as inputs, the process of the final coloured Petri net activity diagram, the flow of tokens inside the final Petri net until reaching the final state. This chapter focuses on providing a case study to examine the methodology and find the results.

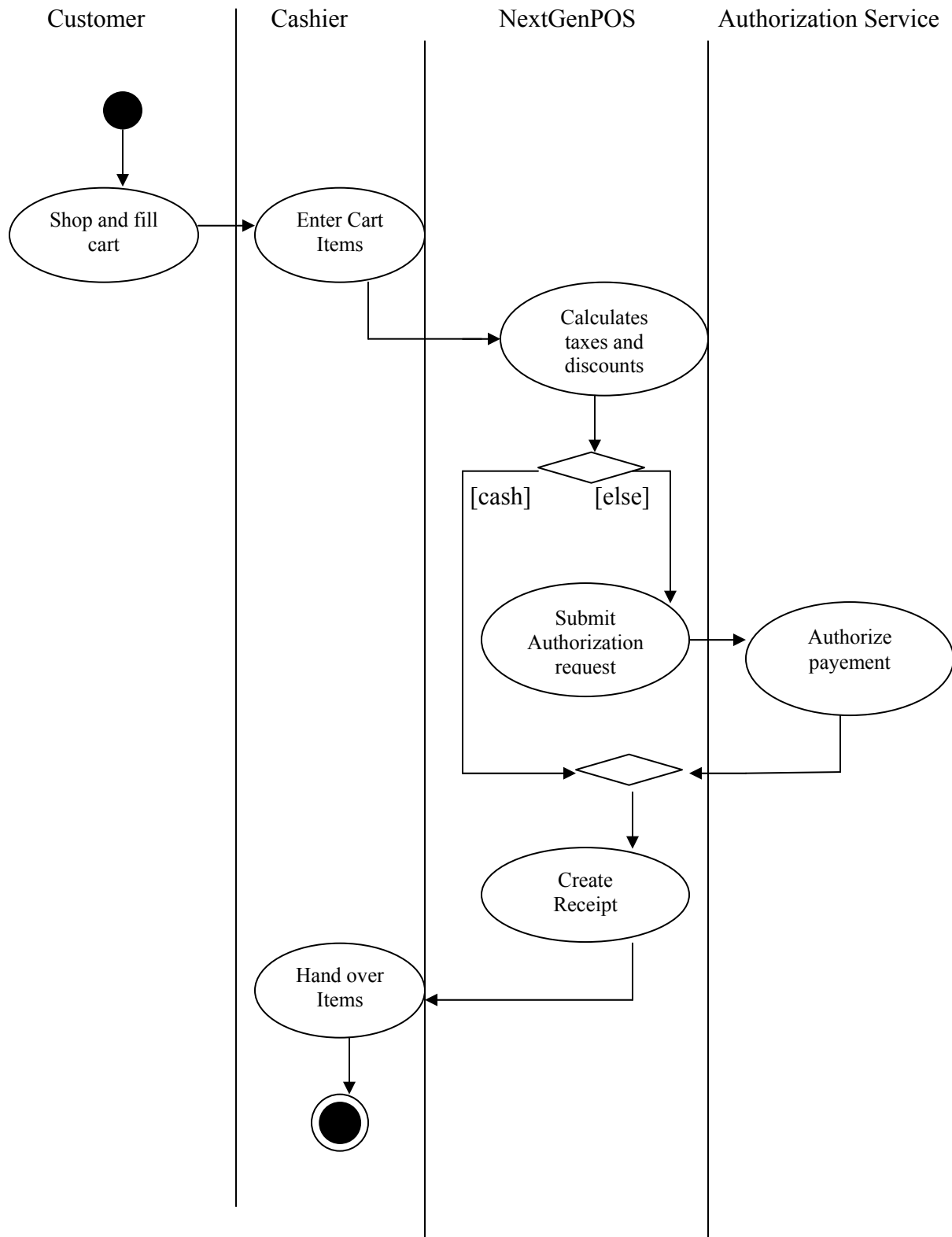
## 6.2 Selected Case Study

The case study that is going NextGenPos, it is chosen because it is a real world application. The main concept of to be used here is NextGenPos activity diagram is depicted in figure 4.1. The activity diagram and all its related documents as a case study is chosen from the applying UML and patterns book [62]. Indeed, this problem was chosen because it is familiar, but rich with interesting design and architectural problems. In addition, it is a realistic problem; organizations really do write POS systems using object technologies. A POS system is a computerized application used (in part) to record sales and handle payments; it is typically used in a retail store. It includes hardware components such as a computer and bar code scanner, and software to run the system. It interfaces to various service applications, such as a third-party tax calculator and inventory control. These systems must be relatively fault-tolerant; that is, even if remote services are temporarily unavailable (such as the inventory system), they must still be capable of capturing sales and handling at least cash payments (so that the business is not crippled). A POS system increasingly must support multiple and varied client-side terminals and interfaces. These include a thin-client Web browser terminal, a regular personal computer with something like a Java Swing graphical user interface, touch screen input, wireless PDAs, and so forth. The activity diagram in figure 4.1 is in the swimlane format. The use case is process sale that has relation with two actors. One is cashier that is a primary actor the other is payment authorization service that is the supporting actor. The concept is in figure 6.1 [62]



**Figure 6.1:** Relation of process sale with its two actors

To developers this is the use case text that is important, the use case diagram is trivial, but since it is graphical it can be a good artifact to describe the use cases graphically, but as it is mentioned, to develop software artifacts such as system sequence diagram and activity diagram and others we need to have the information use cases texts. Moreover in the following use case text for process sale is given [62]. This use case is for a real POS and shows the ability of use cases to capture complex real world requirements and deeply branching scenarios. In the next page the activity diagram is given in the format of swimlane.

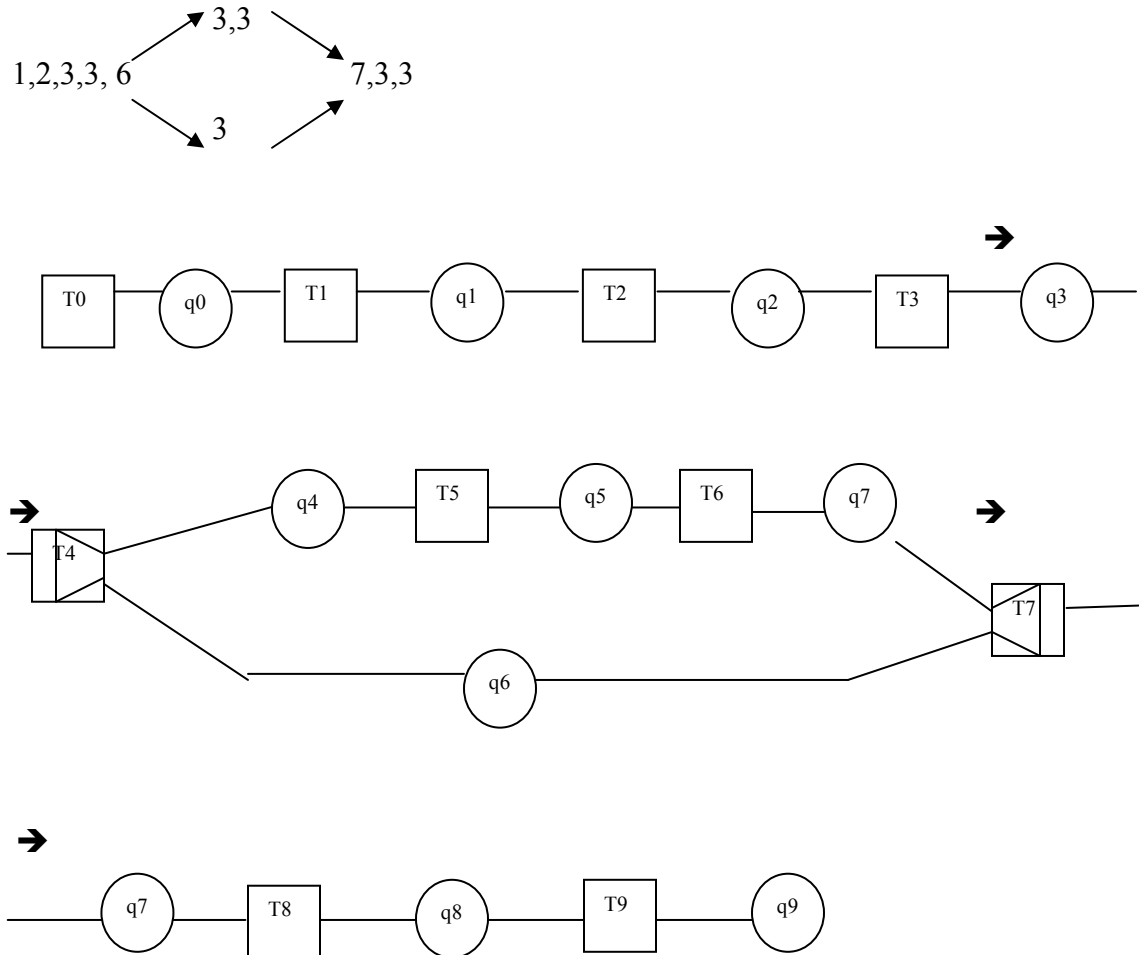


**Figure 6.2:** Process Sale Activity Diagram

### 6.3 Mapping

In order to verify and validate, as it is mentioned in the methodology we have to map the activity diagram to the equivalent coloured Petri net. The mapping will be done by the following rules number and by the algorithm provided in the methodology part.

The concept is in figure 6.3



**Figure 6.3:** The mapped activity diagram

The CPN above contains nine transitions and nine places,  $t_4$  and  $t_7$  are OR-SPLIT and OR-JOIN resembles decision and merge. The tool is provided for drawing the CPN based on the activity diagram.

## 6.4 Validating and Verifying

In order to verify the CPN we have to simulate the CPN with many tokens. The tool that has been provided does that by compiling the CPN. The first step for running the tool is compiling the tool. The tool works if the diagram starts with a transition and ends with a place. In each place and transition you may see the statistics but the statistics in the final place is the result of simulating the Activity Diagram. By validation we want to say that is the final activity diagram that the analyst drew is correct based on the activity diagram rules or not. By validation the tool checks the number of OR-SPLIT and OR-JOIN, AND-SPLIT and AND-JOIN, to see either these components mach each other or not. For our case study we have all these three steps; compiling, validation and verification.

Suppose that we are in the third iteration in RUP and the developer enters the parameters as it will be mentioned later. The tool compiles by asking the user about the first transition and the last place, if it is true the tool asks the final input time, by giving this time to the tool actually the user indicates how many tokens he wants the tool be simulated with. In our case it is assumed that the input final time is entered 400,000 by the user. We know the minimum and maximum transition T0 time, so we can calculate the number of input tokens via the input time. The minimum time for T0 is 24 seconds and the maximum time is 76 seconds and the average is 50. It means that each 50 seconds a transition T0 may fire a new token to the system. By knowing that we can say each 400,000 seconds T0 may fire roughly 8000 tokens in the system. After compiling its time for validating the CPN. The result of validating for our case study is TRUE because the number of or OR-SPLIT and OR-JOIN matches each other. After validation it's time to verify the diagram. Through verification we can find the non functional parameters of the activity diagram. At the end of this stage five non functional parameters will be presented in the output place Q10 based on the numbers the user entered for each parameter. The developer of the system is responsible for the input numbers.

The minimum and maximum transition time is given as follow:

$$t_0=(24,76), t_1=(19,57), t_2=(18,52), t_3=(9,12), t_4=(8,10), t_5=(6,15), t_6=(6,11), \\ t_7=(7,9), t_8=(6,9), t_9=(5,9)$$

Also the user should input three other parameters I, E and W to the system. The parameters easiness of intrusion and importance of information are given for each place as follow:

$$q_0=(0,0), q_1=(0,0), q_2=(0,0), q_3=(1,2), q_4=(1,2), q_5=(1,2), q_6=(0,0), q_7=(1,3) \\ ,q_8=(1,2), q_9=(1,2), q_{10}=(0,0)$$

Four other parameters related to the transitions (failure ratio, W, I and E) are given as follow for each transitions;

$$t_0=(0,0,0,0), t_1=(0,0,0,0), t_2=(0,0,0,0), t_3=(10,2,2,2), t_4=(8,2,1,2), t_5=(7,1,1,2), \\ t_6=(0,0,0,0), t_7=(9,2,1,1), t_8=(10,2,2,2), t_9=(0,0,0,0)$$

The given input for I, E and W depends on the developer view of the system and based on the input we can analyze the output. It is assumed that the developer defines ranks for these parameters between zero and three as follow:

$$0=\text{none}, 1=\text{a little}, 2=\text{somewhat}, 3=\text{very}$$

By the above translation we may assess transition 7 as follow:

It has failure ratio of 9, its weakness of cryptography is somewhat, its importance of data is a little and its easiness of wiretapping is a little. Then based on what we have assumed first we may assess the output. For example we can say if the final security on

The network is more than 60 it is very non secure, if it is between 40 to 60, it is somewhat non secure, if it is between 20 to 40, it is a little non secure. And if it is less than 20 it has Good security. Based on the formula we defined in the methodology, the less the result for security is the more secure the activity diagram is. After verifying the activity diagram we may have the following results for the activity diagram as in table 6.1. The results in the table 6.1 are based on the input given in this chapter and formulas in chapter 4 and the provided tool, ADET, in chapter 5.

**Table 6.1:** Results of activity diagram verification in its third iteration using the tool

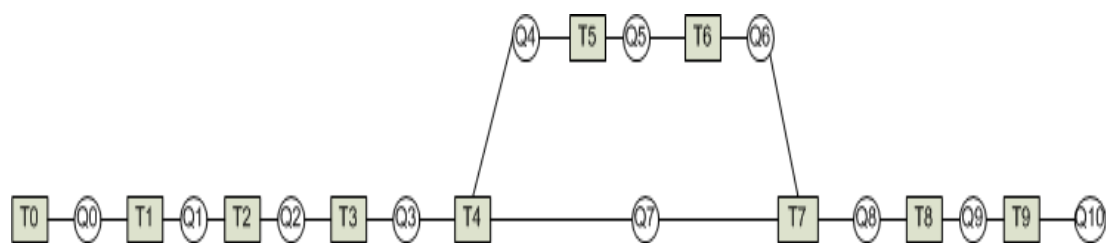
Reliability	Security on net	Security on memories	Performance	Resource efficiency
66.44	22.57	0	131.2	0.95

## 6.5 Discussions on the Results

Any dissatisfaction from the result leads to reconsider some requirements in the RUP process or re programming the software. All depends on the developers and customers need. The tool gives us some more information besides defined the outputs. The most important one is the size for each place. After the simulation is finished you will be able to see how much tokens remained in special places. If the number of tokens is very much, it means that the transition that is located in front of the place is slow. An Activity diagram with these kinds of transitions is not useful. Because as the time

increases the number of tokens in the place increases too and this congestion may effect the activity diagram to have a bad efficiency.

We don't want our tokens be waited in places more but if we have such transitions time of allocation for each token will be increased. But in terms of the output calculated in the table, we can say that the reliability of activity diagram which is 66.44 is less if we assume 100 % reliability for an ideal reliability. And it is because of the reliability of transitions. As it is shown in the figure 6.4 because of five transitions, T3, T4, T5, T7, and T8 with failure rate 10, 8, 7, 9 and 10 respectively. The developers should develop an activity diagram with more reliable software components on the other word these transitions as software components should be programmed more precise. For security on the net that is 22.57 we can say based on the definition we had for I, W and E, the activity diagram is somewhat secure. If more security is wanted, the developer should develop an activity diagram with better I, W and E. the transition that affects the security on the network are T3, T4, T5, T7 and T8 with their given inputs in section 6.4. Security on memories and files are Q3, Q4, Q5, Q6, Q7, Q8 and Q9. The total execution time is 131.2 seconds that seems good for such an activity diagram, if promotion is needed developers need to decrease the time needed for doing each activity. Performance depends on all the transitions in CPN. The less the resource efficiency the better. In this case 0.95 is an acceptable result. Resource efficiency relates to I and E as well as total time of execution. By managing them we may get better results. All the formulas used for calculation are given in chapter 4.



**Figure 6.4:** The mapped activity diagram

## 6.6 Summary

This chapter has explained the related case study and also examined the case study in terms of given input. The verification and the validation process were reviewed. The case study was mapped to CPN and in terms of input parameters the output was calculated. The results were calculated through the tool. And finally the calculated results were discussed to see how it effected the activity diagram. .

## **CHAPTER 7**

### **CONCLUSION AND FUTURE WORK**

#### **7.1 Conclusions**

The study concludes to validate the activity diagram as well as verify it based on non functional parameters. The previous works and related literature are reviewed to have a better understanding of the related research areas. The previous study have shown us that we can not derive non functional parameters from a functional diagram and this limited developers to find out the quality values of activity diagram and likewise they couldn't validate it. But through the formalism of the coloured Petri net, analysis of activity diagrams in terms of non functional parameters is done and it was because we used a formal modeling language to model an informal modeling language. At the end the result of these experiments were assessed. This work followed by four main objectives that through the report each of them has been analyzed completely. The first three objectives have been studied in chapter 4, and the last objective has been described in chapter 5. The contribution of all the objectives finally leads to project aim.

## 7.2 Future works

In this research the experiments are fully depended on extracted formulas over Coloured Petri net. The more accurate formulas for nonfunctional parameters, the better and more exact results. As a future work improving and enhancing the quality values formulas is a matter of concern. Hopefully the output of the research can be a motivation for further investigation in validation and verification process in UML diagrams with considering more nonfunctional parameters like dependability, usability and so on. By improving the output, it will help in finding more exact quality values.

Although applying the process of validation and verification in other diagrams and architectures like model driven architecture and service oriented architecture can be done. In addition, further works to improve the output of the research is suggested in terms of using new algorithm and new formulas and using new modeling languages besides Coloured Petri net.

Hopefully this research gives at list an idea and spreading the knowledge in the research field, especially in other behavioral UML diagrams, software architectures. Also, it is hoped that the developed system and tool can be used by other researchers to develop more researches in software engineering.

## REFERENCES

1. M. Jeng and W. Lu “Extension of UML and its conversion to Petri nets for semiconductor manufacturing modeling” IEEE, Volume 3, 11-15 May 2002, Page(s):3175 – 3180
2. H.Motameni, A.Movaghar and B.Kardel “Verifying and Evaluating UML Activity by Converting to CPN” " Proc of SYNASC'05, Romania, Sep 2005
3. M. Uzam A.H. Jones and N. Ajlouni “Conversion of Petri Net Controllers for Manufacturing Systems into Ladder Logic Diagrams” IEEE, 1966
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
6. IEEE Std 1320.1-1998. IEEE Standard for Functional Modeling Language—Syntax and Semantics for IDEF0. New York, IEEE, 1998
7. M.Hause “The SYSml modeling language” Fifth European Systems Engineering Conference, 18-20 September 2006
8. M.T.Brown “A picture is worth a thousand words: energy systems language and simulation”, Ecological Modelling, 178: 83-100, 2004
9. Stephen A. White, Business Process Modeling Notation (BPMN), IBM Corporation, Version 1.0 - May 3, 2004
10. A. Knöpfel, B. Gröne and P.Tabeling “Fundamental Modeling Concepts: Effective Communication of IT Systems “, Germany, 2008

11. X.He ea “A metamodel for the notation of graphical modeling languages”, in: Computer Software and Applications Conference. COMPSAC 2007 - Vol. 1. 31st Annual International, Volume 1, Issue , 24-27 July 2007 Page(s):219 – 224
12. Atkinson, Th.Kuhne “model driven development: a metamodeling foundation” IEEE software, published by the IEEE society 2003
13. D.C. Schmidt “model driven engineering”, published by IEEE society 2006
14. P.Mohagheghi, J.Agedal “Evaluating Quality in Model-Driven Engineering”, International Workshop on Modeling in Software Engineering (MISE'07), IEEE, 2007
15. D. Gracanin, Sh.A. Bohner, M. Hinchey “Towards a Model-Driven Architecture for Autonomic Systems”, Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04), 2004
16. A.E. Bell ; “Death by UML Fever” , the Boeing Company, Vol. 2, No. 1 - March 2004
17. J.W. Reeves, “Code as Design: Three Essays ”, developer. The Independent Magazine for Software Professionals, 2005
18. D.Jörg and J. Gabriel "What Is a Petri Net? Informal Answers for the Informed Reader",Hartmut Ehrig et al. (Eds.): Unifying Petri Nets, LNCS 2128, pp. 1-25, 2001
19. P.J. Haas “Stochastic Petri nets for modelling and simulation ”, IBM Almaden Research Center, Proceedings of the 2004 Winter, Simulation Conference, 2004
20. T.Gu, P. A. Bahri, G. Cai” Timed Petri net based formulation and an algorithm for The optimal scheduling of batch plants”, Int. J. Appl. Math. Comput. Sci, 2003, Vol. 13, No. 4, 527–536
21. W.M. Zuberek” Timed Petri Net Models of Cluster Tools”, IEEE 2000
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
25. B.Bostan-Korpeoglu, A. Yazici” A fuzzy Petri net model for intelligent databases”, Data & Knowledge Engineering, Elsevier, 2006

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
27. The Object Primer 3rd Edition” Agile Model Driven Development with UML 2 Cambridge University Press, 2004 ISBN#: 0-521-54018-6
28. G.Booch” The Unified Modeling Language User Guide”, Addison Wesley ISBN 0-201-57168-4, 2005
29. K.Jensen ”Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.”Volume 1 (Monographs in Theoretical Computer Science. An EATCS Series) Springer ISBN 3-540-60943-1, 1999
30. P.J.Haas ”Stochastic Petri Nets”, Springer, ISBN 0-387-95445-7 , 2002
31. A.Drathen “Compact Modeling of Manufacturing Systems with Petri nets”, IEEE, 2007
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
34. Ferrarini & luigi, 2004; “Automatic synthesis of multiple place resources models with Petri net”, Proceeding of the 2004 American Control Conference Boston. Massachusetts June 30 -July 2, 2004
35. Tricas & Ezpeleta, 2003; “Some results on siphon computation for deadlock preventio in resource allocation systems modeled with petri net ”, IEEE, 2003
36. Hanzalek, 1998; “Algorithm modeling with Petri nets-comparison with data dependence graphs”, IEEE, 1998
37. Juanole & faure, “On gateway for internetworking through ISDN: architecture and formal modeling with Petri nets”, IEEE, 1989
38. Bernardi & donatelli & merseguer “From UML Sequence Diagrams and Statecharts to analysable Petri netModels ”, WOSP '02, July 24-26, 2002 Rome, Italy ACM ISBN 1-1-58113-563-7 02/07, 2002
39. Baresi & Pezze “Improving UML with petri nets”, Electronic Notes in Theoretical Science 44 No. 4 Published by Elsevier Science, 2001

40. Saldhana & shatz “UML Diagrams to Object Petri Net Models: An Approach for Modeling and Analysis”, Department of Electrical Engineering and Computer Science University of Illinois at Chicago, Seke, 2000
41. H.Kang , X.Yang , Sinmiao Yuan “Modeling and Verification of Web Services Composition based on CPN”, IEEE, IFIP International Conference On Network and Parallel Computing – Workshops, 2007
42. C.H Yang, J.X Liu, H.H. Chen, X.S. Luo “Using CPN Modeling and Simulating the Task-Oriented C2 Organization”, Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22, August, IEEE, 2007
43. P. Lei, W. Lei , Y. Yalan, Y. Fengqi , Y. Hai” CPN Modeling and Analysis of HMIPv6”, IEEE, Proceedings of the 2007 International Conference on Integration Technology March 20 - 24, Shenzhen, China, 2007
44. A. Kovatcs, E. Nemeth, K. M. Hangos” Modeling and Optimization of Runway Traffic Flow Using Coloured Petri Nets”, IEEE, Intemational Conference on Control and Automation (ICCA2005) June 27-29, Budapest, Hungary, 2005
45. T.E. Lee , J.S. Wu , C.H. Lin “A colored Petri-net model for load-transfer in MRT power systems”, IEEEJune 23-26, Bologna, Italy, 2003
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
48. Aorim, L, Maciel, P, Nogueira, M, Barreto, R. Tavares, E” A Methodology for Mapping Live Sequence Chart to Coloured Petri Net ”, Federal University of Pernambuco, 2005
49. C.H. Lin “Distribution network reconfiguration for load balancing with a coloured Petri net algorithm ”, IEEE, Proceedba &ne no. 20030 IB, 2003
50. Sh.Yao, M. Zhou,J. Zen “State Space Analysis on an AI Problem with CPN Model ” Proceedings of ICSP2000, 2000

51. Sh.Yao, J.Billington, M.Zhou “WebMUX Protocol State Analysis With CPN Modelling”,IEEE, 2000
52. C.Pin Lin, Y.Pin Lin, M.Jeng “Design of Intelligent Manufacturing Systems by Using UML and Petri Net”, IEEE, International Conference an Networking, Sensing & Control Taiwi, Taiwan, March 21-23, 2004
53. Zhou, M.C.; DiCesare, F. “A Petri net design method for automated manufacturing systems with shared resources”, IEEE, 1990
54. S. Castano , P. Samarati , C.Villa ” Verifying System Security Using Petri Nets ”, IEEE, 1993
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
56. I.S. Lima, A. Perkusicht ,J.C. A. de Figueiredol “An Interactive Petri Net Tool for Analysis and Simulation of Complex Systems”, IEEE, 1996
57. S. Distefano, A. Puliafito, M. Scarpa “GridSPN: a Grid-based non Markovian Petri Nets Tool”, Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05), 2005
58. J.Billington, G.R. Wheeler , M.C. Wilbur-ham, “PROTEAN: A High-level Petri Net Tool for the Specification and Verification of Communication Protocols” IEEE transactions on software engineering, vol. 14, No. 3, March 1988
59. H.Chen, Amodeo.L, F. Chu, Labadi, K.” Modeling and performance evaluation of supply chains using batch deterministic and stochastic Petri nets” IEEE transactions on automation science and engineering, vol. 2, No. 2, April 2005
60. G.K. Janssens, A.Caris, K. Ramaekers ” Time Petri nets as an evaluation tool for handling travel time uncertainty in vehicle routing solutions” Expert Systems with Applications, In Press, Uncorrected Proof, Available online 17 July 2008
61. Osama S. Youness, W.S. El-Kilani, W. F. Abd El-Wahed” A behavior and delay equivalent petri net model for performance evaluation of communication protocols ” Computer Communications, Volume 31, Issue 10, 25 June 2008, Pages 2210-2230;
62. C.larman “applying UML and patterns” an introduction to object oriented Analysis and design and the unified process, second edition, 1998

63. K.Fukuzawa, M. Saeki”Evaluating Software Architectures by Coloured Petri Nets, SEKE’ Fourteenth International Conference on Software Engineering and Knowledge engineering, 2002