MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET LA RECHERCHE SCIENTIFIQUE
Université 20 Août 1955 Skikda
Faculté des sciences - Département d'informatique

Thèse de doctorat de 3° cycle en informatique
Option: Computation et cognition des systèmes informatiques
Présentée par : Amel Dembri
Intitulée:

Plateformes et Outils pour la modélisation et la simulation des systèmes à évènements discrets: Application aux systèmes industriels

---

Défendue publiquement : Le 6 février, 2024
devant le jury composé de:

Abdelhak Mansoul	MCA	Université 20 Août 1955- Skikda	Président
Mohammed Benmohammed	Professeur	Université Abdelhamid Mehri Constantine 2	Examinateur
Yacine Kissoum	MCA	Université 20 Août 1955-Skikda	Examinateur
Mohammed Redjimi	Professeur	Université 20 Août 1955- Skikda	Encadrant

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
Université 20 Août 1955 Skikda
Faculty of science- Department of computer science

PHD Thesis in Computer Science, 3rd Cycle
Option: Computation and Cognition of Computer Systems
Presented by : Amel Dembri

Entitled:

Platforms and Tools for modeling and simulation of discrete-event systems: Application to industrial systems

---

Publicly defended: On February 6, 2024
Examination Committee:

Abdelhak Mansoul	MCA	University of 20 Août 1955 Skikda	President
Mohammed Benmohammed	Professor	University of Abdelhamid Mehri Constantine 2	Examiner
Yacine Kissoum	MCA	University of 20 Août 1955 Skikda	Examiner
Mohammed Redjimi	Professor	University of 20 Août 1955 Skikda	Supervisor


بسم الله الرحمن الرحيم

الحَمْدُ لله عدد ما خلق، الحمد لله ملء ما خلق، الحمد لله عَدَدَ مَا فِي السَّمواتِ وَمَا فِي الأرض، الحمد لله عَدَدَ مَا أَحْصَى كِتابُهُ، والحَمْدُ لله على مَا أَحْصَى كِتابُهُ، والحَمْدُ لله عَدَدَ كل شيء، والحمد لله مِلْءَ كلِّ شيء

اللهم صل و سلم على محمد عدد ما ذكره الذاكرون و عدد ما غفل عن ذكره الغافلون

الى روح أمي الغالية،

إلى أبي حفظه الله

حرصتما كل الحرص على تربيتنا و تعليمنا

كنتما أعظم النعم علينا جزاكما الله عنا خير الجزاء

ربي اغفر لهما وارحمهما كما ربياني صغيرا

Acknowledgment

First and foremost, I would like to praise Allah the Almighty, the Most Gracious, and the Most Merciful for His blessing given to me during my study and in completing this thesis. May Allah's blessing goes to His final Prophet Muhammad (peace be up on him), his family and his companions.

Thanks to everyone who taught me, my mother, she was my first teacher; may Allah reward her with the highest levels of paradise. To all my teachers, from elementary school to university, and to all those from whom I have gained knowledge, including researchers and writers, thank you.

I would like to express my deepest gratitude to my Ph.D supervisor, Pr Mohammed Redjimi, for having supervised me, advised, directed, reassured and especially supported during these years of thesis.

My sincere thanks go to the gentlemen of the Examination Committee member: Doctor Mansoul Abdelhak for accepting to be the president of the jury. I express my deepest thanks also to Professor Benmohammed Mohammed and Doctor Kissoum Yacine for their time dedicated to evaluating, commenting and making suggestions regarding the content of this thesis manuscript.

I also extend my heartfelt gratitude to my family and friends who have supported me. Your belief in me has been my greatest motivator.

Constantine, February 6, 2024

Abstract

Model-driven engineering (MDE) supports designers and assists them in the realization of their applications by referring only to the model artifact. The case of industrial systems takes on a particular framework which takes into consideration aspects of real time, confidentiality, rigor and security. Thus, one of the major objectives defended in this work consists of providing a set of simulation and modeling solutions for such systems. Among these solutions, an original approach is developed in this work concerns an extension of Petri nets (APN: Agent Petri Nets), this makes it possible to formalize the behaviors of complex systems at a high level of abstraction and to validate them. The proposed approach concerns the development of a solution that combines a formal method (APN) and model-driven engineering tools. A tool is deployed in the form of a Cinco plugin. This work allows designers to manipulate APN models and get an early experiment with them using simulation capabilities. Finally, a comparison covering some of the most interesting model driven architecture (MDA) modeling solutions is developed. This approach would help and guide designers to choose among the most effective solutions to design and develop their applications.

Keywords: : Abstraction, Modeling and simulation, Model-driven engineering (MDE), formal methods, Petri Nets, Cinco tooling suite, EMF, GMF, Agent Petri Nets (APN).

Résumé

L'ingénierie dirigée par les modèles (IDM), accompagne les concepteurs et les assiste dans la réalisation de leurs applications en se référant uniquement à l'artefact modèle. Le cas des systèmes industriels revêt un cadre particulier qui prend en considération des aspects temps réel, confidentialité, rigueur et sécurité. Ainsi, un des objectifs majeurs défendus dans ce travail consiste à fournir un ensemble de solutions de simulation et de modélisation pour de tels systèmes. Parmi ces solutions, une approche originale est développée dans ce travail concerne une extension des réseaux de Pétri aux agents (APN : Agent Petri Nets), ceci permet de formaliser les comportements des systèmes complexes à un niveau d'abstraction élevé et de les valider. L'approche qui été proposée dans ce travail concerne le développement d'une solution qui combine une méthode formelle(APN) et des outils de l'ingénierie dirigée par les modèles. Un outil est déployé sous forme de plugin Cinco. Ce travail permet aux concepteurs de manipuler des modèles APN et de les expérimenter grâce à des fonctionnalités de simulation. Enfin, une comparaison couvrant les solutions de modélisation MDA (model driven architecture) parmi les plus intéressantes est élaborée. Cette démarche permettrait d'aider et de guider les concepteurs à choisir des solutions, parmi les plus efficaces, pour concevoir et développer leurs applications.

Mots clés: Abstraction, Modélisation et simulation, Ingénierie dirigée par les modèles (IDM), méthodes formelles, Réseaux de Pétri, Cinco tooling suite, EMF, GMF, Agent Petri Nets (APN).

الملخص

تدعم الهندسة المبنية على النماذج (MDE) المصممين وتساعدهم في تحقيق ذلك من خلال استعمال القطع النموذجية.تكون حالة نمذجةالأنظمة الصناعية إطارًا خاصًا يأخذ في الاعتبار جوانب الوقت الفعلي والسرية والدقة والأمن. وبالتالي، فإن أحد الأهداف الرئيسية التي تم الدفاع عنها في هذا العمل يتمثل في توفير مجموعة من حلول المحاكاة والنمذجة لهذه الأنظمة. من بين هذه الحلول، تم تطوير نهج أصلي في هذا العمل ويتعلق بإحدى إمتدادات شبكات بيتري (APN: Agent Petri Nets)، وهذا يجعل من الممكن إضفاء الطابع الرسمي على سلوكيات الأنظمة المعقدة على مستوى عالٍ من التجريد والتحقق من صحتها. يتعلق هذا العمل بتطوير حل يجمع بين الأساليب الرسمية والأدوات الهندسية القائمة على النماذج. يتم نشر الأداة في شكل مكون إضافي لتطبيق Cinco. يتيح هذا العمل للمصممين التعامل مع نماذج APN وتجربتها باستخدام إمكانيات المحاكاة. وأخيرًا، تم تطوير مقارنة تشمل بعض حلول البنية المعتمدة على النماذج (MDA) الأكثر إثارة للاهتمام. ومن شأن هذا النهج أن يساعد المصممين ويرشدهم إلى الاختيار من بين الحلول الأكثر فعالية لتصميم تطبيقاتهم وتطويرها.

الكلمات المفتاحية: التجريد، النمذجة والمحاكاة، الهندسة القائمة على النماذج (IDM)، الطرق الرسمية، شبكات بيتري، مجموعة أدوات CINCO، EMF، GMF، وكيل بيتري نتس (APN).

Contents

List of Figures

List of Tables

List of Abbreviations

APN	Agent Petri Nets
ATL	Atlas Transformation Language
CIM	Computation-Independent Model
PIM	Platform-Independent Model
M2M	Model to Model
CPN	Coloured Petri Nets
DEDS	Discrete Event Dynamic Systems
DES	Discrete Event Systems
DEVS	Discrete event system Specification
EMF	Eclipse Modeling Framework
GEF	Graphical Editing Framework
M2T	Model to Text
MBE	Model Based Engineering
MDA	Model Driven Architecture
MDD	Model driven development
MDE	Model Driven Engineering
MDSE	Model-Driven Software Engineering
MGL	Meta Graph language
MOF	Meta Object Format
MSL	Meta Style language
OMG	Object Management Group
OPN	Objects Petri Nets
PSM	Platform-Specific Model
PT	Place/Transition Nets
SysML	System Modeling Languages
UML	Unified Modeling language

Part I

Thesis Overview

Chapter 1

General Introduction

1.1 Context and Problematic

As any engineering fields, the industrial system engineering is a discipline that carried on designing, analysing and implementing system; the crucial purpose of this discipline is to improve human life and avoid unsafety situations to be happening. Manufacturing, robotic, transportation and healthcare are the principle application domains of this engineering[1]

Formal methods and statistical techniques have been strongly adopted by the industrial system engineering; formal methods are utilized to check the correctness of the modeled systems and improve their performance; the statistical techniques contribute in the analysis of data that could solve industrial market problems.

Industrial system often involves complex behavior where critical situations frequently can characterise the behavior of some part of this system. The design of industrial system using formal methods helps designers to correctly check the correctness of the system. In addition of verification, simulation also helps designers to predict the behavior of system by providing an animation tool for a target system.

Master the complexity of systems using facilities that support oriented object and multi agent systems are subject of several researches. Many software applications and programming languages are supporting these approaches;

the choice of a suitable modeling language that supports these approaches have a huge advantage on the development of such applications (a support and simplification of the transaction from model to its implementation).

The petri nets formalism[2] is an old mathematical language where a panorama of researches proves its efficiency in the designing, analysing, simulating and checking various kinds of systems, particularly for the discrete event systems. The formalism is employed successfully in the resolution of numerous industrial modeling examples in many domains as transportation and business process[3].

Coloured Petri nets[4], objects petri nets[5] and agent petri nets[6] are some brilliant petri nets extensions adopted by several researches to fill some gaps of developed system complexity, they contribute by offering a powerful modeling solution for a target class of problem. One of the advantages of this adoption is that the model components could be interpreted in a common world semantic; tokens can be interpreted as objects or agents and the transition system is corroborated by object functions or agent relations.

The assistance of a satisfactory software has a positive effect in the development of applications. Petri Nets and its extensions are corroborated by modeling software that help designers to concretise their PN models and benefit from the automatic verification and analysis functionalities offer by these software.

Despite the availability of a huge number of software solutions that support the original formalism and its extensions, the extending of the formalism, still a conundrum for designers that want to adopt a new PN extension in the development of their applications where no support software existed.

Agent Petri nets(APN), as the name implies, dedicates to design multi agent systems, the main purpose of this extension is to offer a satisfactory modeling language to represent the characteristics of this domain. The formalism doesn't possess a software support to assist designers in the development of modeling solutions that enable them to specify APN models.

Developing a modeling solution from scratch decreases the productivity of system development and increases the cost of this process. Tools are competing to provide automatic facilities to assist computer users to get an end user application through abstract specifications, where platforms are responsible to

provide all guarantees of automatic transitions from abstract representation to binary code, this is known as Model-Driven Software Engineering (MDSE)[7].

Increasing the abstraction level of model components will ensure their reuse by other applications; this consequently elevates the productivity of development process. Selecting the suitable solution to assist designers in the adoption of convenient modeling approach still a challenging issue.

1.2 Objectives

The integration of Petri nets and their classes in the MDE will act very positively in the productivity of the modeling solution where the definition of new language for specific domain could easily develop. Getting a new application by only focusing on modeling stage shapes the new vision of computing; modeling is considered as a crucial step in the development of applications, it allows designers to put system in a high level of abstraction. More having applications that support abstracting computing artifact, non-engineering designers will have the opportunity to create their own tailored applications; irrelevant details such technical aspects should be automatically taken on consideration by the corresponding platforms.
The main objectives addressed in this thesis cover the following points :

- Providing a valuable reference for PN designers community by exposing the available software solutions, highlighting the capabilities of these tools and emphasizing the deficiencies of them. These facilities will give a helping hand for designers to identify the convenient supported tools for their applications.

- Assisting Model driven users in the selection of suitable MDE modeling solutions for their systems

- Exporting the capabilities of selected Eclipse modeling frameworks through a practical exposure and providing precious study report that assists designers in selecting and the developing of their applications within these frameworks.This study offers also a valuable resource for non-engineering designer; a worthwhile report that will encourage them to employ this technique in the development of their own applications.

- Developing an extending modeling-simulation solution for APN models that could be easily adapted to any future changes. The approach will be an application of MDE.

1.3 Thesis Organization

The first part of this thesis gives an overview about the both formal method and model driven engineering solutions. In the second chapter (2), an understanding about what is Petri Nets formalism is introduced, the design capabilities of the formalism are presented and promoted by examples, the classification of different derived classes and the state of arts of general analysis techniques are introduced. The third chapter (3), deals with the introduction of the model driven engineering methodology, concepts around this technology are exhibited.

The second part of this thesis is devoted with our contributions:

In chapter 4, we expose a valuable report about the released petri net software, we elaborate an analysis of these solutions.

In chapter 5, we propose an analysis study for targeted workbenches, enumerated criteria are proposed to deal with this work.

In chapter 6, our approach to develop a platform for Agent Petri Nets formalism is presented, we demonstrate the adopting approach end we present our implemented simulator; a real critical use case are employed in this demonstration.

Finally, to wrap this work, we present the achieved points and we highlight the future ones that appear in the horizon of this contribution.

Chapter 2

A Graphical Formal Language

2.1 Introduction (Industrial systems and formal methods)

Enhancing software quality by the adopting formal methods tools in the developments of industrial systems is well known [8]. Formal methods ensure a rigour validation of the modeled system that encourage designers and software developers to adopt such formalism in the development of critical systems in order to avoid an unsafety situation to take place.

From informal viewpoint, using graphical tools promote the understanding of the model between team members project. Intuitively, graphical methods are more suitable than textual ones for this mission. Petri Nets (PN) and Discrete Event System Specification (DEVS) are typically two big graphical formal methods that have been adopted in the industrial system development approaches. Continuously, these formalisms are extending their application domains by providing series of derived classes that keep up them with the newly in computer science technologies.

Regarding readability characteristic , PN model could be considered more privilege due to its readability feature; the formalism has the potential to successfully represent structural concurrent system features in a comprehensive representation as communication, mutual exclusion, parallelism, and resource share situation. In addition, users could graphically execute a token

play game to verify simple model execution scenario without a software support. To adapt the PN to new development requirements (via proposed new extension), many extensions have been proposed; so a continuously enhancing PN capabilities still viewing. This characteristic encourages designers to adopt this formalism in their project development approach (it will keep up their project evolution in the future).

In several industrial projects around the world, Petri nets are used in the specification and validations of panorama of projects including several domains: communication protocols, networks, transportation and business process[3]. The PN represents best practices for designing and analysing critical systems.

From a mathematical viewpoint, the tool promotes the verification of modeled systems proprieties through several equations and from a visual viewpoint, the concrete representation increases the readability of the modeled system.

Regarding to Petri nets concepts and the proposed extensions that adapt the tool to specific class of problem, studies proved the efficient of the Petri Nets (PNs) in the design and analyze of discrete event systems (DES) [9]. Several Petri Nets extensions have been adopted successfully in the design and verification of several kinds of systems ranging from discrete, concurrent and parallel ones.

The remainder of the first chapter is structured as following: in section2, we start by highlighting the formal and informal definitions of Petri Nets. In section 3, the design capabilities of Petri Nets are emphasized with a short description and modeling examples. In section 4, a classical real life example is designed using Petri Net. Following that, the definition of both behavioral and structural PN properties are presented. Section 5, deals with the classification of PN classes. Finally, a conclusion of this chapter is provided.

2.2 Petri Nets Definitions

The Place Transition net (shortly PT) is a specific type of Petri net widely used in the design of concurrent and distributed systems. Although there are many extensions of Petri nets, the PT is the most well-known and widely used method for introducing the basic elements of Petri net. It provides a clear and

concise way to convey the core concepts of Petri net modeling, making it easier for researchers, students, and practitioners to understand and work with Petri net. The following definitions will specifically address this extension.

2.2.1 Informal definition

Sixty years ago, the dissertation of Carl Adam Petri [2] gave rise to the Petri Nets formalism. The graphical representation of the well knowing Petri Nets didn't appear in the Adam dissertation [10] ; three years later, this mathematical contribution is used and referenced it by Petri nets Formalism [10] . After that, the tool have been widely used by researchers in multiple disciplines as a modeling and simulation tools for many kinds of systems.

Graphically, Places, transitions, and arcs are used to design modeling solutions using Petri Net formalism. These structural components of the net are represented respectively by Circle and rectangle shapes to concretize state and transition elements. Links are used to represent arcs that establish the connection between these elements with the respect to forbid connections between elements that belong to the same kind (no connection between Places and no ones between transitions).

2.2.2 PT Formal definition

Definitions and notation

The structure and the behavior of the Place/Transition nets is well defined in theses researches:[11, 12].

Definition 1.1 (nets structure)

A structure K of PT net is a bipartite graph described by the following 4-tuple:$K = (P, T, IP, OP)$ [11, 12] ; Where:

- P and T are respectively two finite sets of Places and Transitions where $P \cap T = \emptyset$.
- IP and OP are respectively the input and the output functions that describe the available relations between P and T which will be concretized

by going edges from Places to Transition and vice versa .

* IP :P×T→ N , is the input function that represents the relation between Places and Transitions.

* OP :T×P→ N , is the output function that represents the relation between Transition and Places; concretized by an arc.

Informally, the events that lead the system are represented by Transitions (graphically depicted by bars or rectangles) and the corresponding conditions that allowing these events to be happening are represented using Places (graphically depicted by circles).

As it is well known in graph theory, the incidence matrix is one of the methods that are used to represent a graph in the machine (there are other representations: adjacency matrix and list)[13].

Petri net belongs to the bipartite directed graph category (the bipartite feature is related to its nods kind and the directed feature is related to the nature of the relation between these kinds that is oriented).

Any PN can be structured by an incidence IM matrix with q×r dimension (q and r, are respectively numbers of Places and Transitions), their values are obtained by subtracting the output matrix values from the input values ones. IM , is defined by the following equation: $IM = IM^{+} - M^{-}$

Where $IM^{+}$and M^{-} are the associated post and pre incidences matrix related to IP and OP functions .

Matrix rows show the relation between an element of P and all elements of T.

Matrix columns show the relation between an element of T and all elements of P.

System dynamic part definitions

In order to describe the system dynamic, functions related to system marking and transition firing rules are given by the flowing definitions.

Definition 1.2 Marking function

Let take the petri net structure K, The distribution of tokens on the net is given by MK function $MK = P \rightarrow N$ where for each net Places a positive value is attributed; this distribution of tokens defines the state of the system [11, 12].

Definition 1.3 marked Petri Net

$R = (K, M_0), K$ is the net structure and M_0 is the related making of this net at the initial point.

Definition 1.4 (activate Events)

A system event can be occur when the condition related to this event is satisfied; in the case for a marked PN, a transition t_y can be activated for a marking MK_n if $\forall\ IP(p_i, t_y) \neq$ and $MK_n(p_i) \geq IP(p_i, t_y)$, the new marking $MK_{n+1}(p_i) = MK_n(p_i) + OP(p_i, t_y) - IP(p_i, t_y)$.

Definition 1.5 (firing sequence)

Given that: $MK_0 \xrightarrow{t1} MK_1$, $MK_1 \xrightarrow{t2} MK_2$, $MK_2 \xrightarrow{t3} MK_3$,$MK_3$...,$MK_n \xrightarrow{tn} MK_\mu$ if $\sigma = t1t2t3...tn$ a sequence of firing transitions from MK_0 to MK_μ noted by we say that MK_μ is a reachable marking from MK_0 by executing the transition sequence σ.

Definition 1.6 PN reachability set

Given PK is a marked PN, MK_0 is an initial marking of PK , a reachability marking set of PK noted by $RM(MK)$ is the set of the possible produced marking from the initial one ; a marking MK_m is reachable if exists a firing sequence leads to this marking from the initial marking. A reachability set of PK could be infinite when no explicit final marking is reachable, the case of unbounded system.

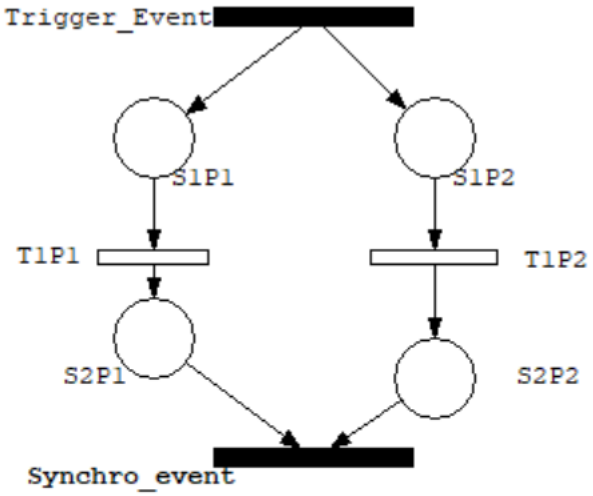

Figure 2.1: Parallelism structure example

2.3 PN modeling capabilities

The PN plays an important role in the design of discrete event systems (DES); PN has been considered as a suitable solution to design the characteristic of DES applications due to its design capabilities that support explicitly the design of parallelism, synchronisation (shared resource, exclusion mutual, critical section) and communication. Several research papers envisage examples of these capabilities; here we give a short description of each ones with a PN structure example and we enumerate some researches that contain these structures.

Parallelism:Petri net allows designers to model simultaneity situations between processes (see Figure 2.1, an illustrative example). For this mission, a transition should be used as a trigger of a simultaneity situation (TriggerEvent); places appeared in this process have only one input and output arcs.

Synchronisation:different synchronisation scenario could be designed using PN: mutual exclusion, RDV, producer consumer, reader writer, etc. The mechanism of synchronisation is expressed by a common transition that enforces two processes to reach some tasks simultaneously (case RDV see Figure 2.2), or a condition state that avoids executing some tasks in parallel (mutual exclusion). In the contribution of [14], examples of synchronisation models are specified using PN.

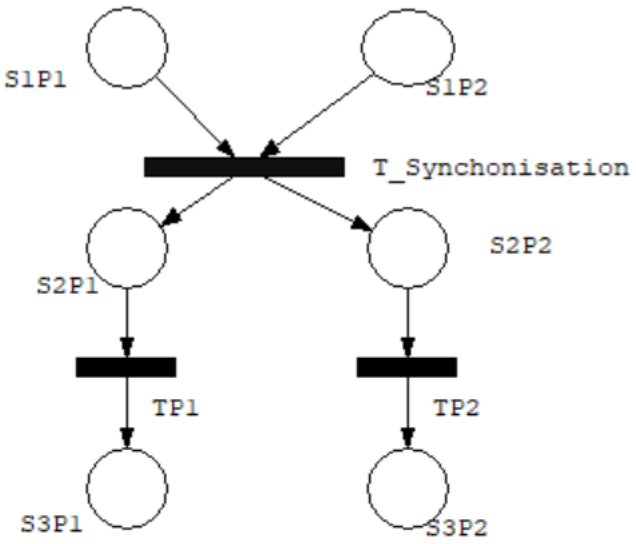

Figure 2.2: Synchronisation model structure

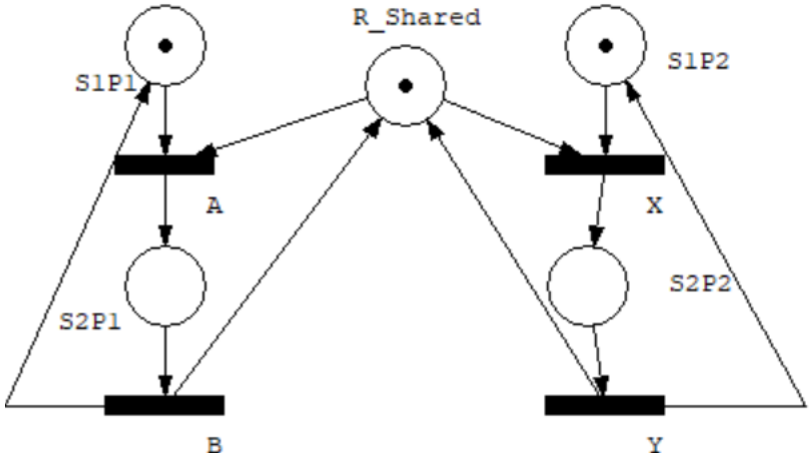

Figure 2.3: Shared Data structure

Shared data: A common shared resource is designed in PN models via a place that contains a token and, this resource is required for the work of some processes in concurrence (the place have output and input connection with parallel tasks) and only one should consume and, give back the token to the shared place (see Figure 2.3). Some researches treated the data sharing example [15].

Communication: To concretise this concept in PN model, the PN transitions are used to represent the signal sending and receiving, the PN places are used as exchange buffers (an exchange buffer plays the role of input place for the sender and an output one for the receiver) see Figure 2.4. The use of PN in the

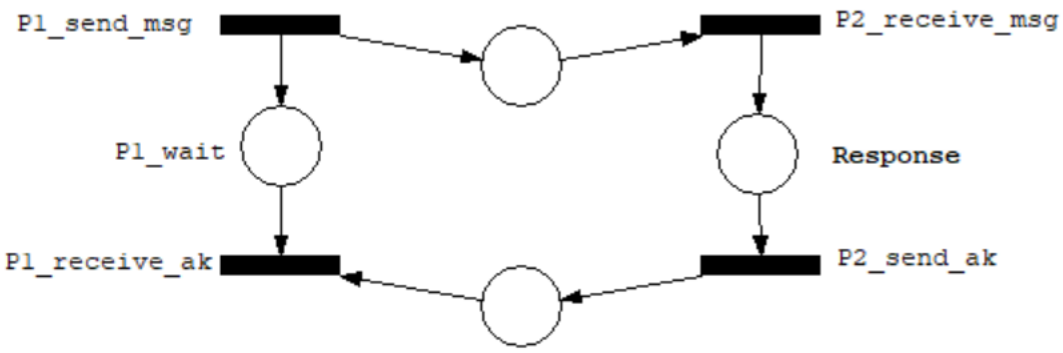

Figure 2.4: Comminication structure

protocol communication area is discussed in several researches; we mentioned here the book of [16] that presents some model examples of processes communication using PN, and in the contribution of [17] the interaction protocol is specified using a high level PN extension.

2.4 PN and a safe situation: a system model example

To highlight some petri net design capabilities, we choose a well knowing process model in literature, which will be presented here: traffic light process model.

A traffic light system is a classical critical system where the unsafe situation could lead to a catastrophic result (cause damages lives and property). Using PN to design this application helps designers to ensure mathematically the correctness of the system.

2.4.1 The structure of the PN example

The model could be represented by given an explicit description of its components and their relations; an incidence matrix could be also useful to represent the traffic light model. Three places are used to represent the possible situation of the traffic light process named respectively RD, GN and OG and also three transitions are used to design a switch from a state to another named RTG, GTO and OTR.

- P= {RD, GN, OG},
- T= {RTG, GTO, OTR},

The available relations between PN nodes are described using IP and OP functions as follows :

- IP=(RTG,RD),(GTO,GN),(OTR, ORG) ,
- OP=(RTG, GN),(GTO, ORG),(OTR, RD) ,
- IP(RTG)={ RD },
- IP(GTO)={ GN },
- IP(OTR)={ OG },
- OP(RTG)={ GN },
- OP(GTO)={ OG },
- OP(OTR)={ RD},

Figure2.5 shows the corresponding PN graph of the above description.

IM is the incidence matrix representation of this model is done by the following matrixes: $IM = IM^{+} - M^{-}$,

IM^{+}	RTG	GTO	OTR
RD	0	0	1
GN	1	0	0
OG	0	1	0

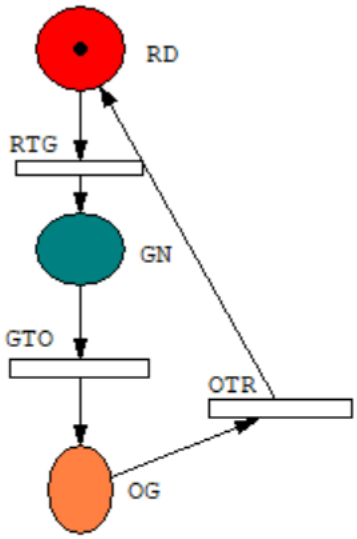

Figure 2.5: TL PN model

IM^-	RTG	GTO	OTR
RD	1	0	0
GN	0	1	0
OG	0	0	1

IM	RTG	GTO	OTR
RD	-1	0	1
GN	-1	1	0
OG	0	1	-1

2.4.2 The model Behavior of the PN example

The model elements: places, transitions and connections design the static representation of the Traffic light system model, the process of moving tokens from a place to other ones (by activation of transitions) leads to dynamics of Traffic light systems. An initial state of this process is proposed, see Figure 2.6 two possible reachable marking from this initial state will be achieved see Figure 2.7 and Figure 2.8 By firing RTO and GTO transitions.

In the second example, two traffic light models will be designed see Figure 2.9. By visual inspection, the development of the system state from the initial situation to another one leads to unsafe state; both of them achieve green state simultaneous (see Figure 2.10). The proposed model leads to an unsafe

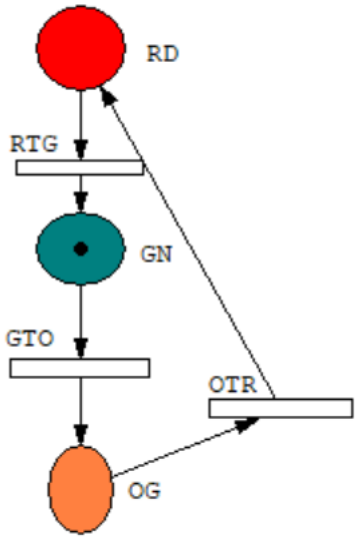

Figure 2.6: Reachable marking distribution B

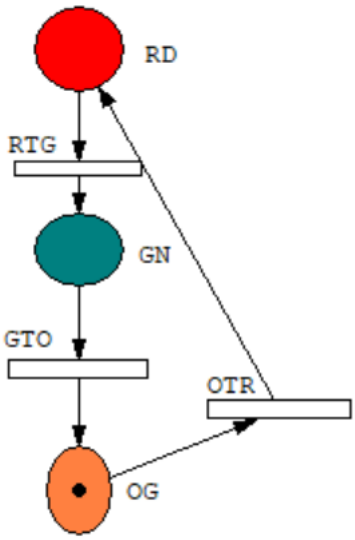

Figure 2.7: Reachable marking distribution C

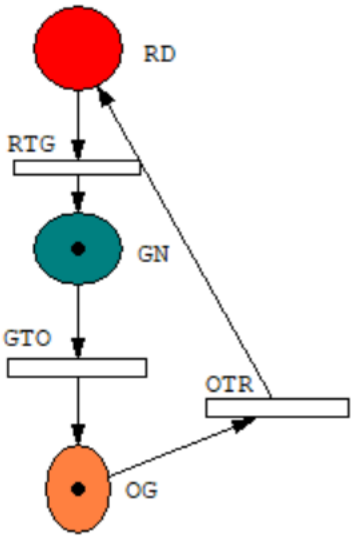

Figure 2.8: An unreachable marking distribution

situation. A correction of this unsafety situation is proposed in Figure 2.11; the place Ctrl ensures that in the same time, only one of them switch to a green state.

If the model has a simple PN representation, designers could evaluate their models by processing a quick visual inspection; a simple token play game helps them. When the system is more complex, the use of verification mechanism of PN is required

2.5 Analyse Petri Nets models

2.5.1 PN properties

In the engineer science, we cannot design the ideal model for a given system but with the help of mathematical relations witch build the model, a rigour study of this system can be process and a judgement about desirable or undesirable behavior could be delivered; so a good system could be implemented.

In the case of petri net formalism, a mathematical definition of proprieties have been proposed. These proprieties could be classed in two main categories: behavioral and structural properties, this classification is based on the depending with the initial marking and the net structure.

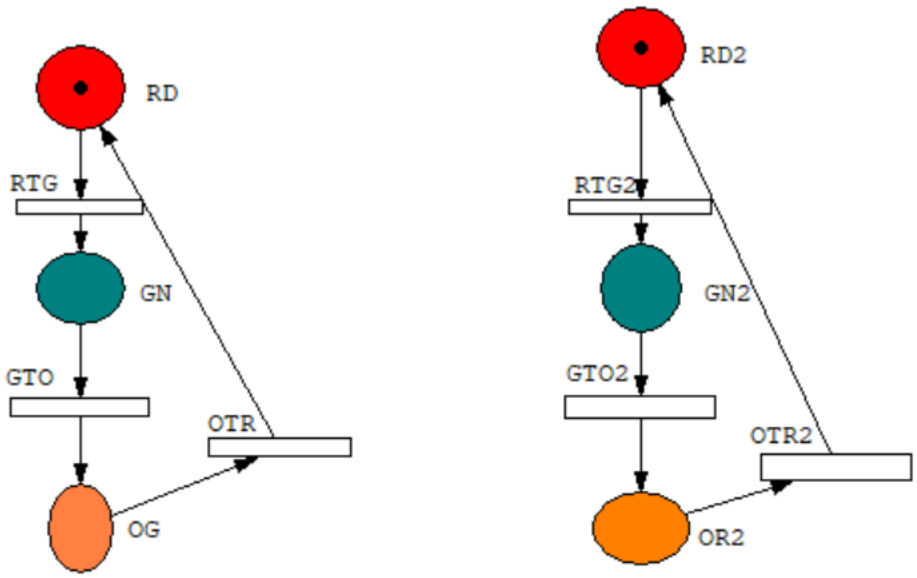

Figure 2.9: Two PN models

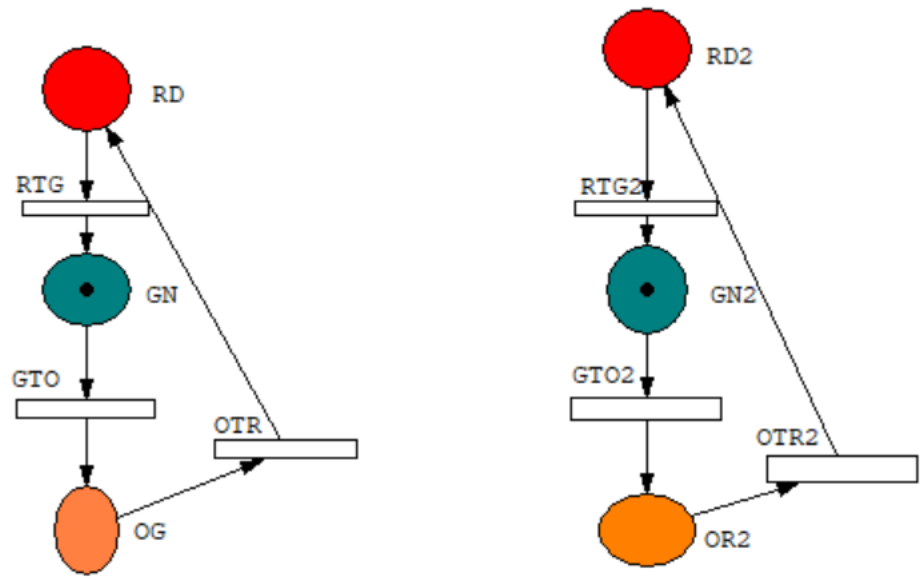

Figure 2.10: Unsafe Situation

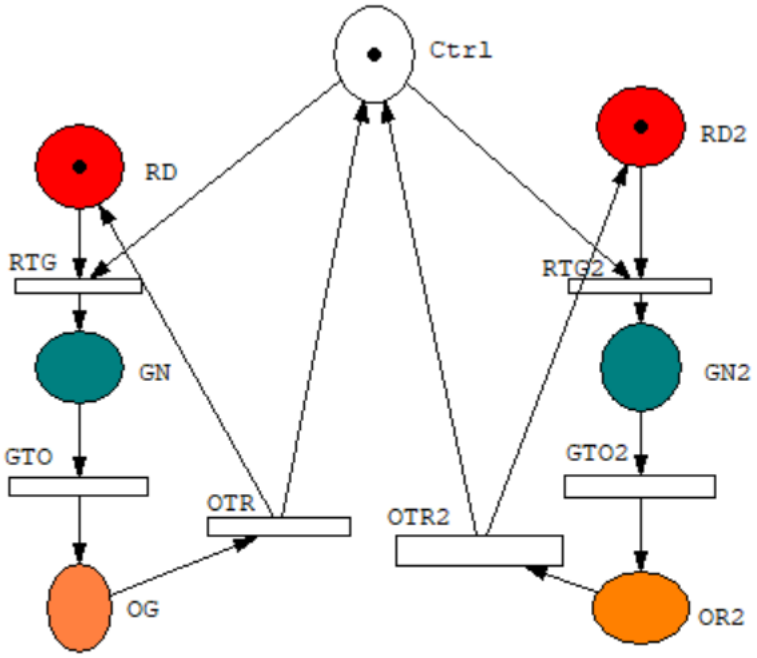

Figure 2.11: Safe Models

Behavioral proprieties

The most usual behavioral properties of PN are liveness, boundedness, conservation [16]. Liveness and deadlocking could be considered as the most important properties to detect failure situations for a given petri net system.

Liveness If a transition has always the opportunity to occur in a transition sequence which belongs to a reachable marking, we say that this transition is live. A petri net possesses a liveness property when all its transitions are live [16] .

This property leads to a deadlock-freedom for a given system; the execution of a part of a system is guarantee but this property doesn't mean that other parts have the potential to be activated.

Liveness considered as the most difficult to be decidable in the analysis of Petri nets and its studies usually cost for large system [18], so a lot of definitions of this property in many levels are proposed in literature; from zero transition firing opportunity to infinitely ones . The following definition is the general one. **Definition 2.1** Let PK is a marked PN, MK_0 is its initial marking, $SM(MK)$ is a smallest set of reachable marking and $t \in T_n$.

PK is live if $\forall t \in T$: t is live;a transition t is live if $\forall MK \in SM \exists MK^{'} \in SM$ such that t is enabled in $MK^{'}$ [18].

A deadlock property for petri net, means that this system has the potential to reach a dead marking where no transition could be executed for it.

Boundedness This property means that the maximum numbers of tokens in places are well-known. A petri net is V-bounded when the max value of tokens in its net places is less or equals to V [11].

This property is useful to check if the number of tokens in each place grows indefinitely or no (practically it is used to check the potential increase or overflows of resource which is designed by tokens).

Definition 2.2
Let PK is a marked PN, MK_0 is its initial marking and $p \in P_n$, *PK* is *V-Bounded* iff $\forall p \in P \ \exists V(P) \in N$ such that $\forall MK \in SM$, $MK(p) \leqslant V(p)$. If $V = 1$,PK said as a safe net.

Structural properties

Structural properties are only related to the net topology, not to the initial marking like behavior ones. The following properties are the most usual: repetitive, conservation structural liveness and structural boundedness [19, 20].

The above liveness and boundedness properties are also defined in the structural properties of a petri net model where the analyse is independent to the initial marking.

Definition 2.3 Structural liveness is concluded from the liveness of an initial marking MK_0; if this last is live, the property is satisfied but the reverse is not true.

Definition 2.4 Structural boundedness is concluded from the boundedness of all possible initial markings MK_0.

Definition 2.5 Conservation This property checks the conservation of resource moved in net places; its aims, to test if each system event gives what

exactly it takes as tokens (quid pro quo rule: tokens consumed should be produced). This characteristic leads to a bounded system but the reciprocal is not guaranteed; a bounded net do not lead to conservative ones.

Repetitiveness
Let *PK* is a marked *PN*, MK_0 is its initial marking and $t \in T$; *PK* is repetitive if all its transitions (t) are repetitive [11, 21].

This property checks the potential to execute a series of events many time; consequently the lived system is a repetitive one but the reverse it is not true [21] (the system could have a repetitive character and in the same time don't possess a liveness property)

2.5.2 Analyse PN proprieties

A model is considered good for a given system; if desirable situations have opportunities to be happening and undesirable ones do not have a chance to occur. In the case of a Petri net system model, the study of the desirable or undesirable situation is done through mathematical analysis of model properties.

In literature, analysis approaches are applied to the petri net model in the mission to obtain a correct model; enumeration by reachability set, linear algebra and net reduction techniques are used in this process. These methods are complementary applied in order to accelerate the analyse process or/and to simplify the application of another approach [21].

Reachability tree To check properties of a Petri Net model, a common technique is to construct the reachability tree, which involves envisioning all possible reachable markings in the net. The process starts with the initial marking as the starting node. By executing all possible activated transitions, all child nodes stemming from the initial node are obtained. For each child node, if a dead marking is reached, it implies that no further reachable markings can be derived from that node. If it is not a dead marking, the same process continues to check for new activated transitions, with the child node becoming the new starting node, until reaching a dead marking. In the case of an unbounded Petri Net model, a coverability graph should be developed.

Linear algebra At the design level, the structural proprieties of Petri net model like structural liveness, structural boundedness, consistency could be verified using linear algebraic technique; this method is based on incidence matrix and fundamental equation [9]. The advantage of this technique resides in its strength to process a structural analysis of the net independently to its behavior, this particularity allows to establish an analysis of Petri net system proprieties in a generalization way than a specific one (the study aims to cover every marking than a given one) [22]. Studies are proving the efficiency of this technique to analyse Petri net properties [20]; its success is coming from the facilities to pick up invariants which are used to promote the analyse of Petri net model by giving useful information used in the analyse process [19].

Reduction of the net: The main idea of this approach is to simplify the original net by equivalent one in order to accelerate and simply the analyse process as possible. The transformation technique should conserve the proprieties of the original Petri net model; it accepts as input the initial net and process a series of reduction rules to explore an output one more simple to manipulate. The case of irreducible system, net reduction techniques could not be applied.

2.6 PN classification and extensions

2.6.1 Categories

An Effort is done by [23] to classify Petri Nets extensions in three distinguished levels; the representation of tokens is the criteria of this classification. The study is inspired from the work of [24]. Monika Trompedeller have enumerated the following categories:

Category A: we can see in this category the Petri Nets Classes that are proposed to represent logical systems as Elementary Net and Condition-Event System where each state can hold maximally one unstructured token.

Category B: For this category, the Petri Places could hold a value that expresses the resource number in each state ; the Place-Transition Nets (shortly P/T nets) is the main example of this category , derived nets form this class could be found as Ordinary Petri nets (P/N), free Choice Nets and 1-Safe Nets Systems, etc.

Category C: Here the token take a high-level representation, it could be an object, a list of symbol, a PN net structure, etc. As examples of nets in this category, we find Coloured Petri Nets, Algebraic Petri Nets, Predicate /Transition Nets, hierarchical Petri nets and Object Petri Nets.
Other classification is widely appears in studies where researches mostly use the key words low and high levels to characterize between the categories of Petri nets extensions. Category A and B belong to the low level Petri nets and category C belongs to the high level nets extensions.

2.6.2 High level Petri nets extensions

Sixty years of the Carl Adam contribution in the theory of Discrete Event Dynamic Systems(DEDS) [10]. Interesting extensions from the initial version of petri nets have been proposed to fill gaps of the limitations of the Place Transition version to answer to design requirements in several classes of problems; here, we are interested in the proposed ones to master the complexity of systems, extensions that addressing oriented object and multi agents development methods.

Two General classes of Petri nets are known: ordinary version and high level ones. The main purpose to extend PN is the requirement to adapt the formalism for a specific class of engineering problem where an extended version will be able to express a solution for this problem. Coloured Petri nets, Object Petri Nets and Agent Petri Nets represent solutions to master the complexity of systems; tokens in these extensions are able to express a data structure and methods are provided by these extensions to allow users to specify concepts of object or agents.

The Coloured Petri Net (shortly CPNs) [4], is a petri net type where tokens are specified with distinctive data values designed by colours. This PN extension is used successfully in designing concurrence system where the validation of such systems is a major challenge; the strength of this modeling language lies in the grouping of power of PN concepts with the capabilities of ML [1] [4]. The formalism is extended to other sub-classes as: Timed Coloured Petri Net and Hierarchical Petri Net. Any Hierarchical Petri Net could be unfolded to CPN [25] . Many numbers of project are used this extension to design a target system; the website of Aarhus University provides a reference list of contributions where CPN has been used in the industrial systems domain [3]. The formal

[1] functional programming language

definition of the main structure and the behavior of this extension is clearly explained in this reference [16].

The Object Petri Net (OPN) is simply a petri net extension where token design the structure and the behavior of an object using a place transition net. This class of petri net is proposed to master the complexity of the system where the object oriented paradigm solution is adopted to design and implement a target system [5]. In the mid-nineties, the first workshops that deal with OPN and the object oriented paradigm was being scheduled; the contributions presented in these workshops were published in a voluminous book [26]. Regarding to the research of [5], this Petri nets extension has many formal marking definitions, so different semantics have been presented: value semantics and reference ones. The application of Object nets in industrial systems has been the subject of several researches; we cited here some of them : the contribution of [27]in Industrial Electronics, the model application of automated manufacturing [28] and the contribution of [29]in the reconfigurable manufacturing context.

The third Petri extension named: The Agent Petri Net, this formalism has been introduced in the research of [6] as a PN class to support a natural development paradigm. Multi agent systems have been the target application area of this extension. The idea is to provide a formalism that represents the feature of multi agent systems; tokens are interpreted as agents and the circulation of agents from a site to another one is naturally controlled by a series of agent functions and relations. In literature, two contributions have been adopted this extension; the first one in the specification of interaction protocol [17], the second one in the specification and analyse of Molecular Biofilms formation process [30].

2.7 Conclusion

In this Chapter, we have presented an overview about both informal and formal definitions of petri nets formalism and real world example is used to design a safe model. Then, some structural design capabilities of this formalism are shortly illustrated by many structure examples. These examples express several modeling situations as parallelism, communication, resource share, synchronisation, and others. We also enumerate some researches that envisage examples of these capabilities. Classifications of Petri net extensions are discussed in this chapter; in particular, we have summarized the main

concept and purpose of three brilliant high level Petri nets that support object oriented and multi agents paradigms.

Chapter 3

Model Based Solution

3.1 Introduction

Human mind applies a cognitive process to create and understand knowledge that perceives from the world. It continuously builds a mental representation of the reality by constructing strong generalization and building a powerful abstraction that result in rich models which used to represent a reality [31].

Abstraction is a key concept in all disciplines: science, art, and technologies, etc. However, the concept of abstraction in art is something difficult; music and paint are two typical examples used in art to represent a part of reality, started with a simple vision to more complicated ones. In contract, abstraction in science is a mechanism of simplification. Scientists are applying abstraction to simplify the understanding of phenomena by construction models. As mentioned in [32], the abstraction mechanism is used to perform two principal roles: removing irrelevant details (focusing on some aspect that seems important from a particular vision and neglect other ones) and generalization specific features of instances (from common features of instance formulates general concepts).

The revolution of computing is a result of the history of adding layers of abstraction to master complexity; building things on top of other things by hiding complexity of the low level. Simple user, software developer, designer, hardware developer, they all have different visions about this machine and

needed to work with the appropriate layer of abstraction according to the details they should work on. When simple users manipulate applications they focus on mastering the use of this application and how they can exploit all application features, they do not need to think about how this application was written(case of desktop applications) or how information is flying from a site to another (case of web application). In computer, data is represented using Zero and One; nevertheless, software developers practically do not need to care on about binary code; they need to master a programming language that plays the intermediate role between programmer and machine. Therefore, a layer of abstraction is added between machine language and programmer through programming language. Hardware developers have another vision and need to care on about other details; in reality, in computer there is no zero and one, structures of fluctuations of magnetic fields are abstracted by a series of zero and one representation. Levels of abstraction are added to hide complexity of system and to simplify the experience of every range of computer users.

Models as a partial representation of reality, it plays a principal role in the revolution of computing abstraction. The use of model is known in all engineering disciplines due to their potential to decrease the complexity of the developed system by focusing on the most important and neglect the irrelevant ones (in computer science, the irrelevant parts are automatically implemented by the target platform and usually related to technical aspect). To make abstraction a reality in computer science engineering world, tools are designed to perform the adequate transformation of software artifact from a representation to another (from the high level of abstraction to the lower level). Compilers, interpreters, generator are means to enable the mapping between various abstraction levels.

Model-Driven Software Engineering, as the name suggests, this technology leads the development process of applications in all the software life cycle. It based on a high-level abstraction of computer artifacts. In this chapter, a brief overview of Model-Driven Software Engineering is elaborated; key concepts that shape this technology are highlighted. The second section deals with the transformation mechanism that leads the application of mapping between models. Finally, we conclude this chapter.

3.2 MDSE key Concepts

3.2.1 Model driven Software engineering concepts

Model-Driven Software Engineering (MDSE) or brief Model Driven Engineering (MDE) is a methodology knowing as a model based approach, which aims to improve the experience of software developer within the construction of applications [7] . The practical experience showed clearly the capabilities of MDSE tools and proved the efficiency of the adoption of this approach in the development of applications. Researchers are the forerunners who adopt this methodology in the concretization of their approaches; MDE plays the miracle role to enable them to get quickly a prototype to their applications through an automatic code generation process. In the last years, this technology is also adopting in the development of applications related to industry domain; studies show clearly the potential of MDE in the development of software business solution [33, 34].

MDE includes four fundamental aspects: concepts, notations, process, and tools [7].

- Models and transformation are two fundamental pillar concepts of MDE.
- Notations used to describe how the concepts will be represented, called in MDE, modeling languages.
- Process designs the follow path to get an end user application.
- Tools (software platforms) that support the development of MDE applications covering modeling and transforming activities.

We will discuss the fundamental pillar concepts of MDE (models and transformation) in the following sections.

3.2.2 Models

Models are means to represent a part of a reality, human always builds a mental representation to better understand his world and the world around him. Modeling activity is applying in daily human life as well as in science to better understand phenomena and theories. In computing, models are used

to deal with reality and to simplify the complexity of application development; proceed an evaluation through analyze, verify and validation of model system is cheaper than addressing directly the application system.

Traditionally, the use of models in development process limited only on documentation that simplified the development of systems and help designers to understand and structure data that is used in the implementation of application. Therefore, this usage is considered by most programmers as a burden due to its limitations (as it is only used as documentation support); a need of facility supports that adopt an automatic transition from models to implementation is required. lots definitions of models in the engineering disciplines world; no common one has been adopted by engineering researchers, we explore here the most relevant ones :

- **Definition 1:**
  Models are means to represent system under study (SUS) through a set of statements where these statements defined by expressions [35].

- **Definition 2**
  Models are means to understand the essence of SUS easily, and hide its complexity by removing irrelevances details and highlighting the most relevant ones for a given viewpoint [36].

- **Definition 3** :
  A model represents a simplified or a reduced realize of a target system to enable system to be developed and analyzed, it obtained by applying an abstraction process on real system elements [37, 38] .

3.2.3 Meta-models

To get a model system, an abstraction process is applied to eliminate as possible the complexity of system from a specific viewpoint. In this process, a modeling language or meta-model is needed to describe models. Like model, many definitions of the meta-models also found in the literature; no common one has been adopted, here some popular ones:

- **Definition 1:**
  “A meta model is a model of models” [38].

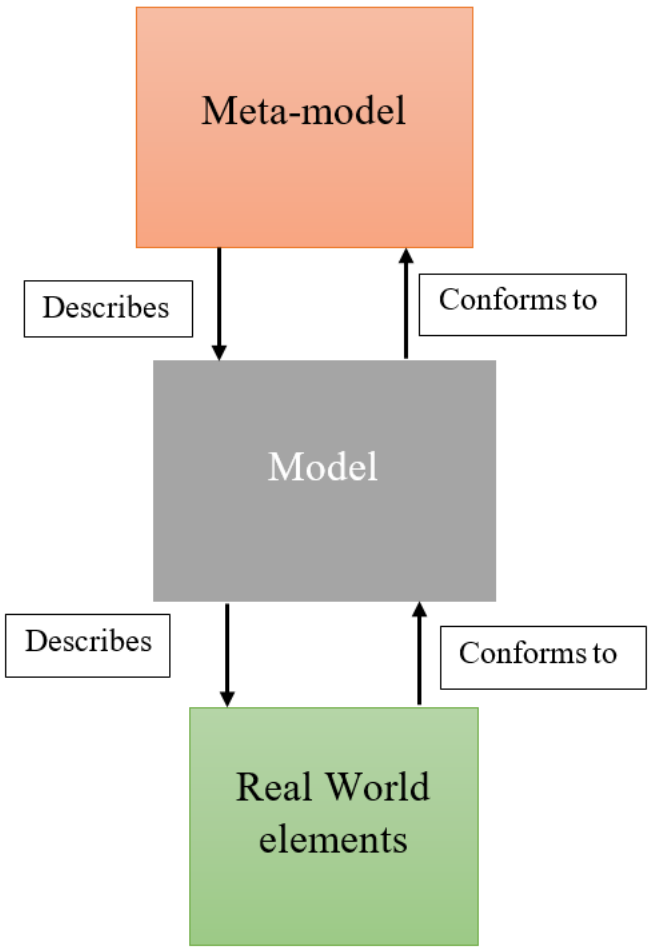

Figure 3.1: Model and Meta-model relations

- **Definition 2:**
  A meta model or meta-language is a is a model that defines the language for expressing a model [10].

- **Definition 3:**
  A meta-model is an abstraction of the model where meta-model constitutes the language that describes the models. 'Describes' and 'conforms to' are two relations design the nature of connection between the meta model and the model in differences levels of abstraction [7]. See Figure3.1.

3.2.4 Abstraction Layers

As it mentioned above in this section, the relation between models, Meta models and real world objects is leaded by two reverse relations 'conforms to' and 'describes'. In model driven engineering, these elements are ranged in

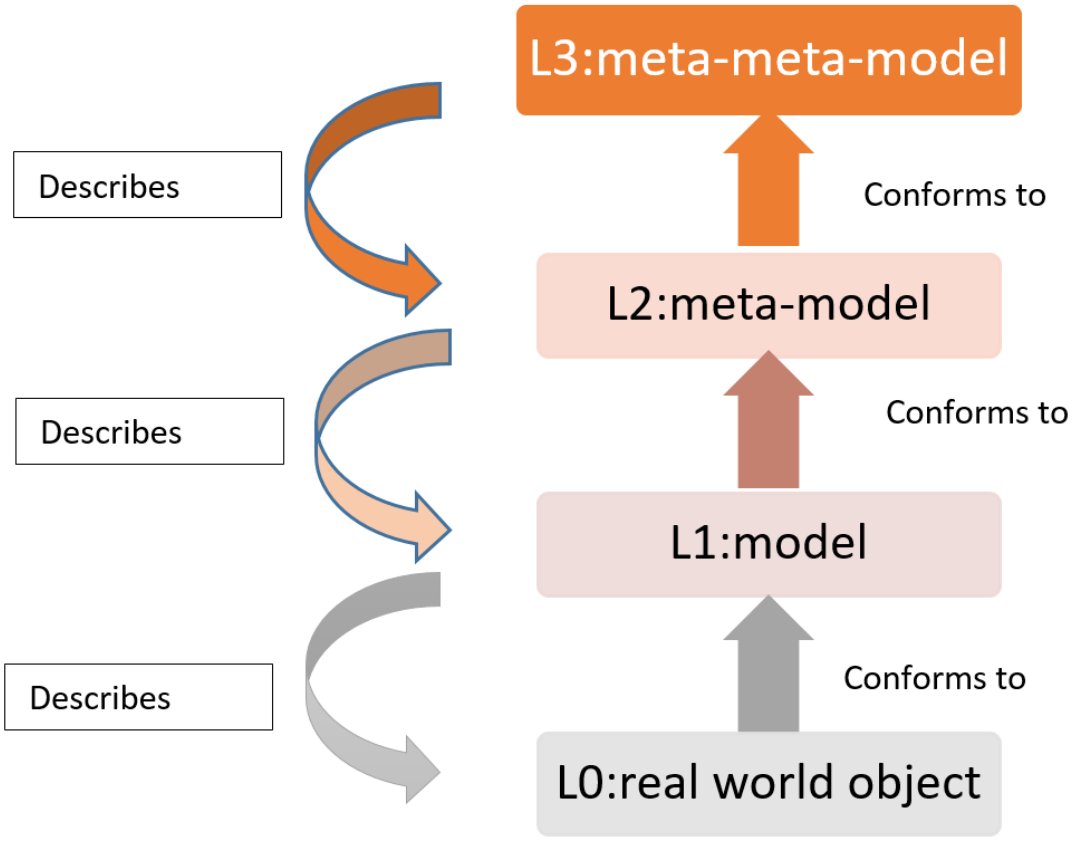

Figure 3.2: The architecture layers

four layers defined by The Object Management Group (shortly OMG) see Figure 3.2.

Each element has a position in a specific layer. The top layer is the most abstract one; it contains the concepts and relations between these concepts that will be used to define the language (meta-model). At the next position, we find the meta-model; this MDE layer element used to define models. The third layer is devoted to model; the definition of this element should be conformed to the meta-model. In the low level (L0), the representation of real world objects it conforms to their models.

3.2.5 MDSE Acronyms

Regarding the state of art of model driven approach, lots of acronyms and terminologies have been traded in model based engineering world, which sometimes caused a misunderstood to model driven practitioners[7] ; some of these acronyms are standardized by the OMG to promote their integration in other software development tools. Here most popular variants acronyms of model driven approach:

Model driven development (MDD) The MDD initiative was initiated in 2003 in the paper entitled model driven development [38]. MDD is considered as a paradigm of development where the models play the centric role in the development process [7] started with a model artifact to get an end user application through an automatic code generation mechanism.

Model driven architecture (MDA) The announcement of the MDA approach was been in 2000 in the research that held the same name Model Driven Architecture [39]. MDA is considered as a particular version of model driven development, it is a MDD implementation supported by the OMG standard specifications that assists designer to work in a high level of abstraction and relives him form any implementation burdens related to a technical platform specification details [7]. Every model specified in MDA tools should respect the Meta Object Format (MOF) [20], this adoption promotes the interoperability of MDA models in software platforms, the standardized specification Unified Modeling language (UML)and XML Metadata interchange specification (XMI) are supported by the MDA [40].

Model driven engineering (MDE) The publication of [41] brings out the main purpose of model driven engineering technologies, MDE considers as a suitable approach to master the complexity of the development of applications based on two crucial steps: modeling and transformation activities. IN contrast to MDA where standardized tools are recommended to use, MDE support other software developments features.

Model based engineering (MBE): The MBSE initiative has been launched in 2007 in the International Workshop of INCOSE with the aim to accomplish the MBSE 2020 Vision [42]; the appears of the MBSE acronym in literature was found in the specification of system modeling languages (SysML) project) [42]. Nevertheless, the earliest paper referring to MBE has been published in 1993 entitled model based system engineering [43].

Figure 3.3 represents an MBE classification effort according to their literature communication date and the extent addressed by each one. The classification is done based on the result of these works: [44, 7].

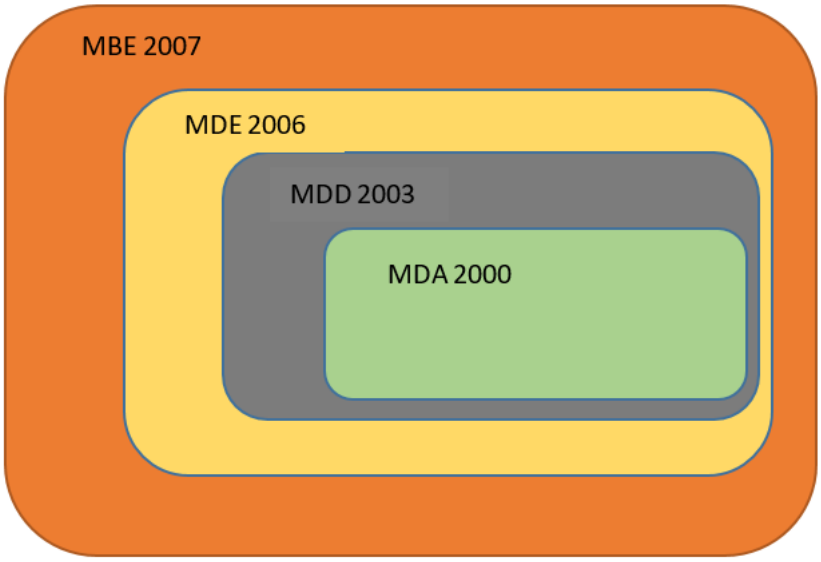

Figure 3.3: Acronyms classification

3.2.6 The MDA models

Recently, the model driven architecture is the most adopted solution in industry [7]. MDA is a trademark posed by the OMG that allows model driven engineering approach to be a reality; the provider standards, and the separation between models by putting them in different abstraction levels promote the adoption of this solution in the development of industrial systems.

Specific architecture layers have been defined by the MDA, in the top level, we found the Computation-Independent Model (shortly named CIM), fellows by the Platform-Independent Model (shortly named PIM) and in the lower layer, Platform-Specific Model (shortly named PSM) takes place [7].

The CIM is the most abstract representation compared to PIM and PSM, it is devoted to outlining the abstract representation of systems without regardless of behavior details. The second model called PIM, in this model, both structure and the behavior of system are described without regard to any the platform requirements. The PSM, the third model in this architecture, is devoted to describe the system model according to a specific platform; in contrast to PIM, platform requirements are supported in this model [7]. The structure of the relation between these models is shown in Figure 3.4 .

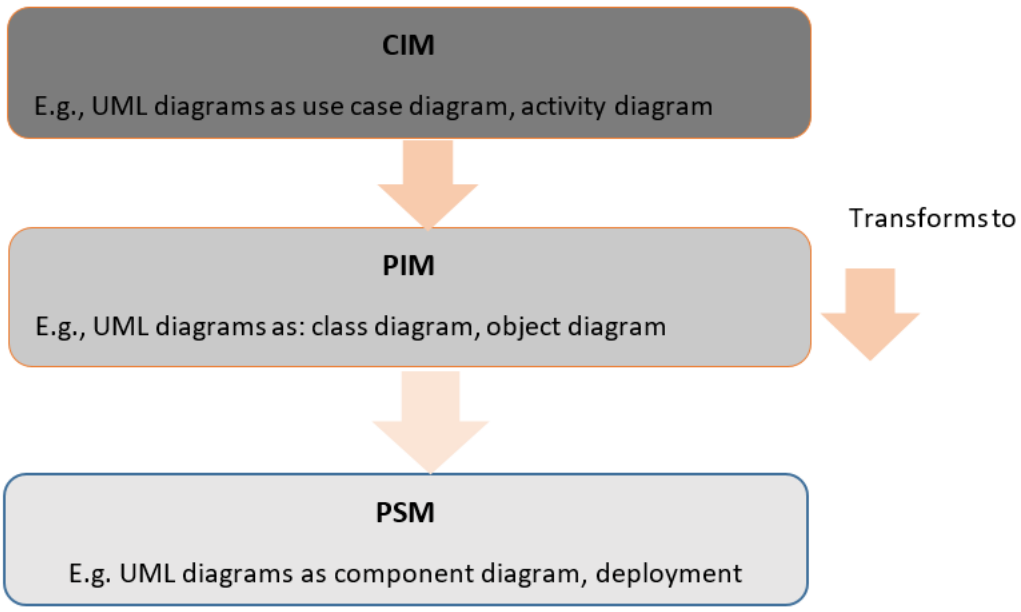

Figure 3.4: MDA models: examples and relation

3.3 Transformation

As it mentioned before in this chapter, the fundamental pillars of MDE methodology are models and transformation. As modeling, transformation plays a crucial role in development process of applications. One of the modeling requirements to adopt MDE in an industrial company, is the potential of tools to support model transformation [34].
The models in MDA could be:

- Moved to another site.

- Restructured by applying refactoring process (keeping the external behavior and improving the internal structure).

- Simplified by applying refined process.

- Translated to other specification

The enumerated operations are included in model transformation process [7].

In the literature, two kinds of the transformation in MDE [7] :

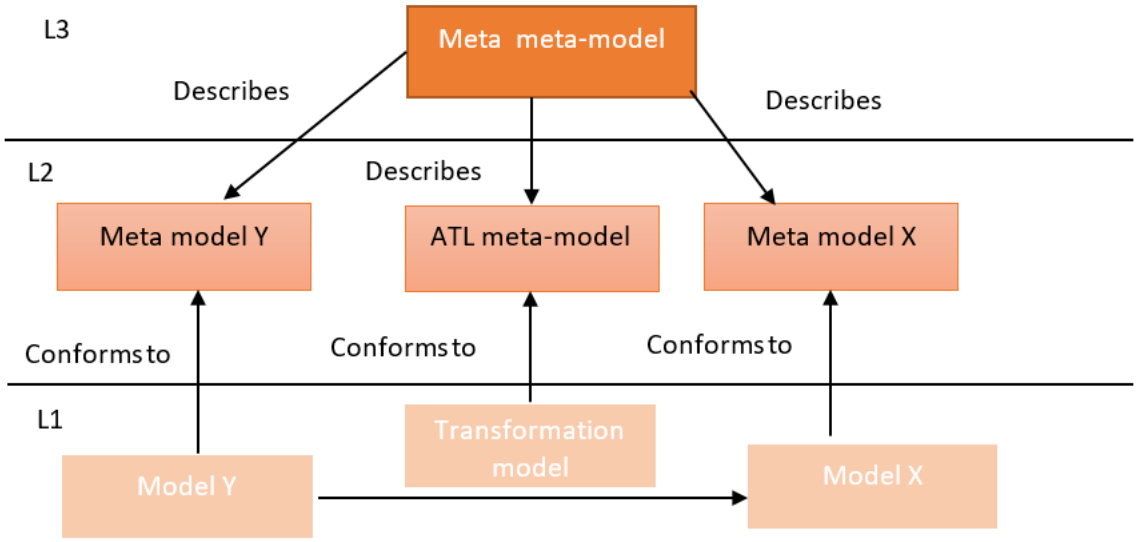

Figure 3.5: Transformation in OMG layer, inspired from [45]

1. Model to Model (shortly M2M) transformation; the case of mapping from a source model to target one [7].

2. Model to Text (shortly M2T) transformation; the case of translation of model to textual representation (text code) [7].

3.3.1 From Model to another

Transformation languages are available on MDE platforms to help designers to specify rules that lead the mapping between a source model and the corresponding target one; these rules are defined conforming to the meta model of these languages. The Atlas Transformation Language (shortly ATL) considered as the most popular transformation language supported by MDE tools [45] This language allows designers to define various kinds of transformation: between one model as source and other as target, or between set of models (a set of source models to a set of target ones)[7].

The definition of transformation and its application follow the OMG layers. In L2 (second layer of OMG model Layers), the definition of mapping between models takes place and the application of this transformation will take place in the following layer (L1) see fig3.5.

3.3.2 From Model to Textual representation

Having a running system by only specifying a series of models is the main purpose of MDE. To realize this mission, a code generation mechanism is integrated with MDE software tools to get automatically the implementation of a specific application. The concept of code generation is an oldest technique in computer science engineering. Without doubt, it didn't have birth with the coming of MDE, compilers traditionally translate developer programmers to machine language (form text written in programming syntax to binary code). In MDE, applying this process follows the OMG layers, firstly the mapping is suggested to be from a more abstract representation called meta-model (or modeling language) to a latter level called model then, from model to a corresponding code (written in a target programming language).

3.4 Conclusion

The software development methodology presented in this chapter is widely adopted by industrial domain in the last dedicates. The idea to get a software implementation by only based on an abstract representation encourages designers to adopt this technology on the development process of their applications. Model driven engineering approach, as it name suggests, based on modeling in the first stage of software development process, the transformation mechanism is a second pillar of this approach. MDE software tools promote the definition of abstract specification of models in different level of abstraction and provide mechanism to help user to translate this specification to another representation and to get the corresponding source code of the modeled application.

Part II

Contributions

Chapter 4

PN Software Tools Evaluation

4.1 Introduction

A huge modeling-simulation software have been implemented in petri nets world to accompany engineers in the development of their applications at modeling and simulation stages. According to [46]; more than 95 modeling- simulation applications are developed to facilitate the application of different petri nets extensions. These tools promote the specification, simulation and the analysis of PN models for one or multiples petri nets classes. Informatik website [46], offers a valuable reference of the existences petri nets software solutions; based on this reference, we aim in this chapter to establish a statistical analysis using IBM SPSS Statistics software[1] .

In our study, we endeavor to provide a helpful reference related to petri nets tools that comprises multiple important points: supported extensions, tool licences, functionalities, platforms and other requirements etc.; all these points and more will be evaluated in this analysis work. The study has devised on two-step:

- Creating database (variable definitions and cases filling)
- Analysing data (execute requests).

[1] A statistical software solution created by IMB; allow user to manage and analyse data [**?**]

In the first section, we create the required models of our database and we fill data as well. In the second section, we execute the adequate request to get the analysis result for each specified request. Finally, we conclude this chapter.

4.2 PN software tools in SPSS

4.2.1 Preparing database

Models creation

To establish the statistical analysis of the selected petri nets tools, we need to define the models that store theses information in SPSS platform. We firstly, specify the variable declaration for the manipulated data through SPSS variable view; for this mission, we choose to define two files[2]: *classes-platforms.sav* (see Figure 4.1) and *functionality.sav*(see Figure 4.2)

Classes-platforms.sav and functionality.sav are created to enable the store of the following information:

- NT (Id tool).
- Tool (Name of tool).
- FreeCommercial (Tool license).
- Supported petri nets extensions.
- Other Classes.
- Supported environments.
- Other requirements.
- Available functionalities.

The FreeCommercial variable is used to specify tool licence; four possible values are specified for this variable (C, C&AD, C&FD and FC) where:

[2] *.sav* is the extension of file created within SPSS

- C: is used to represent Commercial licence
- C&AD: is used to represent commercial licence with academic discounts.
- C&FD : is used to represent commercial licence with academic Free distribution .
- FC: is used to represent free charge distribution.

To identify that the current case (a PN tool) supports a PN extension of not, two possible values (S and US) are used for each PN extension variable where:

- S: is used to represent supported.
- US: is used to represent unsupported.

To identify the availability of series of functionalities for each PN case, two possible values (AV and NAV) are used for each PN available functionality variable where :

- AV: is used to represent the availability of a feature.
- NAV: is used to represent the unavailability of a feature.

Filling Cases

After specifying variables, we fill data through SPSS data view. 90 cases are filled; tools with their identifier, name, supported PN classes, supported platforms, and the available features. We have chosen to classify Tools regarding to their distribution licences; the commercial tools are classified in first places following by tools that provide discounts for students, then the ones that provide a free access for academic uses and finally open tools for all users.

Using sort cases[3] features in SPSS , we obtain the following result(see table 1 and table 2). We choose to split theses tables to other sub tables (to clearly

[3]A feature that allows users to classify cases based on criteria.

	Name	Values	Label	Type	Measure
1	NT	None	NT	Numeric	Scale
2	Tool	None	Tool	String	Nominal
3	FreeCommercial	{C, Commercial}...	Free/Commercial	String	Nominal
4	PlaceTransitionNets	{0, US}...	Place/Transition Nets	Numeric	Nominal
5	HighlevelPetriNets	{0, US}...	High-level Petri Nets	Numeric	Nominal
6	PetriNetswithTime	{0, US}...	Petri Nets with Time	Numeric	Nominal
7	StochasticPetriNets	{0, US}...	Stochastic Petri Nets	Numeric	Nominal
8	ObjectorientedPNs	{0, US}...	Object-oriented PNs	Numeric	Nominal
9	StochasticSymmetricNets	{0, US}...	Stochastic Symmetric Nets	Numeric	Nominal
10	HybridPetriNets	{0, US}...	Hybrid Petri Nets	Numeric	Nominal
11	OtherClasses	None	Other Classes	String	Nominal
12	Windows	{0, NA}...	Windows	Numeric	Nominal
13	SunOS	{0, NA}...	SunOS	Numeric	Nominal
14	Linux	{0, NA}...	Linux	Numeric	Nominal
15	MacintoshMacOSX	{0, NA}...	Macintosh, Mac OS X	Numeric	Nominal
16	HPHPUX	{0, NA}...	HP, HP-UX	Numeric	Nominal
17	DigitalUNIX40	{0, NA}...	Digital, UNIX 4.0	Numeric	Nominal
18	SiliconGraphicsIRIX	{0, NA}...	Silicon Graphics, IRIX	Numeric	Nominal
19	Java	{0, NA}...	Java	Numeric	Nominal
20	Solaris	{0, NA}...	Solaris	Numeric	Nominal
21	otherrequirements	None	Other requirements	String	Nominal

Figure 4.1: Classes-platforms model

	Name	Type	Label	Values	Measure
1	NT	Numeric	NT	None	Nominal
2	Tool	String	Tool	None	Nominal
3	FreeCommercial	String	Free/Commercial	{C, Commercial}...	Nominal
4	GraphicalEditor	Numeric	Graphical Editor	{0, NA}...	Nominal
5	TokenGameAnimation	Numeric	Token Game Animation	{0, NA}...	Nominal
6	FastSimulation	Numeric	Fast Simulation	{0, NA}...	Nominal
7	Codegeneration	Numeric	Code generation	{0, NA}...	Nominal
8	CondensedStateSpaces	Numeric	Condensed State Spaces	{0, NA}...	Nominal
9	StateSpaces	Numeric	State Spaces	{0, NA}...	Nominal
10	InterchangeFileFormat	Numeric	Interchange File Format	{0, NA}...	Nominal
11	RapidPrototyping	Numeric	Rapid Prototyping	{0, NA}...	Nominal
12	RareEventSimulation	Numeric	Rare Event Simulation	{0, NA}...	Nominal
13	Modelchecking	Numeric	Model checker	{0, NA}...	Nominal
14	StructuralAnalysis	Numeric	Structural Analysis	{0, NA}...	Nominal
15	PlaceInvariants	Numeric	Place Invariants	{0, NA}...	Nominal
16	TransitionInvariants	Numeric	Transition Invariants	{0, NA}...	Nominal
17	WorkflowManagementSystem	Numeric	Workflow Management System	{0, NA}...	Nominal
18	NetReductions	Numeric	Net Reductions	{0, NA}...	Nominal
19	SimplePerformanceAnalysis	Numeric	Simple Performance Analysis	{0, NA}...	Nominal
20	AdvancedPerformanceAnalysis	Numeric	Advanced Performance Analysis	{0, NA}...	Nominal

Figure 4.2: Functionality model

present data). The first table ranges tools with their PN supported extension and platforms, this one is split into four parts: Table 4.1, 4.2, 4.3 and 4.4. The second table, ranges the tools with their available functionalities, this one is split into three ones 4.5, 4.6, 4.7.

Tool	FreeCommercial	PlaceTransitionNets	HighlevelPetriNets	PetriNetswithTime	StochasticPetriNets	ObjectorientedPNs	StochasticSymmetricNets	HybridPetriNets	OtherClasses	Windows	SunOS	Linux	MacintoshMacOSX	HPHPUX	DigitalUNIX40	SiliconGraphicsIRIX	Java	Solaris	otherrequirements
ALPHASim	C	US	S	S	US	US	US	US	/	AV	AV	NA	NA	NA	NA	NA	NA	AV	/
MISS-Rdp	C	US	US	S	S	US	US	US	coloured petri	AV	AV	NA	NA	NA	NA	NA	NA	NA	/
Nevod	C	US	US	US	US	US	US	US	Inhibitor Petri nets	NA	NA	NA	NA	NA	NA	NA	NA	NA	MS DOS
Opera	C	US	US	S	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	MS DOS
Simulaworks, Petri Nets Simulator	C	S	US	S	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
Artifex	C&AD	S	S	S	US	S	US	US	/	AV	AV	AV	NA	AV	NA	NA	AV	NA	/
COSA BPM	C&AD	US	S	US	US	US	US	US	/	AV	AV	AV	NA	AV	NA	NA	NA	NA	/
ExSpect	C&AD	S	S	S	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
FLOWer	C&AD	S	US	US	US	US	US	US	/	AV	NA	AV	NA	AV	NA	NA	AV	NA	/
F-net	C&AD	US	US	S	S	US	US	US	/	AV	NA	AV	NA	NA	NA	NA	NA	NA	/
Kontinuum	C&AD	US	S	S	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
PACE	C&AD	S	S	S	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
Petri.Net Simulator	C&AD	S	US	S	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
Petri Net Toolbox	C&AD	S	US	S	S	US	US	US	Generalized Stochastic Petri Nets	NA	NA	NA	NA	NA	NA	NA	NA	NA	/
PROTOS	C&AD	S	US	S	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
SimHPN	C&AD	US	US	US	US	US	US	S	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	MATLAB 7.0 or higher
GDToolkit	C&AF	S	S	US	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
GreatSPN	C&AF	US	S	S	S	US	US	US	/	NA	AV	AV	NA	NA	NA	NA	NA	NA	/

Table 4.1: Part 1 of filling data:PN tool supported extension and platforms

Tool	FreeCommercial	PlaceTransitionNets	HighlevelPetriNets	PetriNetswithTime	StochasticPetriNets	ObjectorientedPNs	StochasticSymmetricNets	HybridPetriNets	OtherClasses	Windows	SunOS	Linux	MacintoshMacOSX	HPHPUX	DigitalUNIX40	SiliconGraphicsIRIX	Java	Solaris	otherrequirements
Netlab	C&AF	S	US	US	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
SPNP	C&AF	US	S	US	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
SYROCO	C&AF	US	S	S	US	US	US	US	/	/	/	/	/	/	/	/	/	/	C++
TimeNET	C&AF	S	S	S	S	US	US	US	/	AV	NA	AV	NA	NA	NA	NA	NA	NA	/
AlPiNA	FC	US	S	US	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
ARP	FC	S	US	S	US	US	US	US	/	NA	AV	NA	NA	NA	NA	NA	NA	NA	MS DOS
CoopnBuilder	FC	US	S	US	US	S	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
Cosmos	FC	S	US	US	S	US	S	US	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	/
CPN-AMI	FC	S	S	US	US	US	US	US	/	NA	NA	AV	AV	NA	NA	NA	NA	NA	/
CPN Tools	FC	US	S	S	US	US	US	US	/	AV	NA	AV	NA	NA	NA	AV	NA	NA	/
Disc Software Platform	FC	S	US	US	US	US	US	US	/	AV	NA	NA	NA	NA	NA	AV	NA	NA	/
ePNK	FC	S	S	US	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	/
Fluid Survival Tool	FC	US	US	S	S	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
Geist3D	FC	S	S	S	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	/
GHENeSys	FC	S	S	S	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	AV	NA	/
GPenSIM	FC	S	S	S	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	/
LoLA	FC	S	S	US	US	US	US	US	/	NA	AV	NA	NA	AV	NA	AV	NA	NA	/
Maria	FC	S	S	US	US	US	US	US	/	NA	AV	NA	NA	AV	AV	AV	NA	NA	/
Mercury	FC	US	US	US	S	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
mist	FC	S	US	US	US	US	US	US	/	NA	NA	AV	NA	NA	NA	NA	NA	NA	/
mist2	FC	S	US	US	US	US	US	US	/	NA	NA	AV	AV	NA	NA	NA	NA	NA	/

Table 4.2: Part 2 of filling data:PN tool supported extension and platforms

Tool	FreeCommercial	PlaceTransitionNets	HighlevelPetriNets	PetriNetswithTime	StochasticPetriNets	ObjectorientedPNs	StochasticSymmetricNets	HybridPetriNets	OtherClasses	Windows	SunOS	Linux	MacintoshMacOSX	HPHPUX	DigitalUNIX40	SiliconGraphicsIRIX	Java	Solaris	otherrequirements
ORIS	FC	US	US	S	S	US	US	US	/	AV	NA	AV	AV	NA	NA	NA	AV	NA	/
P3	FC	S	US	US	US	US	US	US	Upgraded Petri Nets	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
PED	FC	S	US	US	US	US	US	US	/	NA	AV	AV	NA	NA	NA	NA	NA	NA	/
PEP	FC	S	S	S	US	US	US	US	/	NA	AV	AV	NA	NA	NA	NA	NA	NA	/
PetitPetri	FC	S	US	S	US	US	US	US	/	AV	NA	AV	AV	NA	NA	NA	NA	NA	/
Petrigen	FC	S	US	US	US	US	US	US	/	AV	NA	AV	NA	NA	NA	NA	NA	NA	/
Petri-lld	FC	US	US	US	US	US	US	US	Elementary Nets	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
Petri-Net Kernel	FC	S	S	US	S	US	US	US	DAWN-Nets	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
PetriSim	FC	S	S	S	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	NA	NA	MSDOS
Petruchio	FC	S	S	S	S	US	US	US	Transfer Petri Nets	AV	AV	AV	AV	AV	NA	NA	AV	NA	/
Platform Independent Petri Editor2	FC	S	US	US	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
PNEditor	FC	S	US	US	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
PNetLab	FC	S	S	S	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
QPME	FC	S	S	US	S	US	US	US	(Hierarchical) Queueing Petri Nets	AV	AV	AV	AV	AV	NA	NA	NA	NA	/
Renew	FC	S	S	S	US	S	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
SamaTulyata	FC	S	S	US	US	US	US	US	/	NA	NA	AV	NA	NA	NA	NA	NA	NA	/

Table 4.3: Part 3 of filling data:PN tool supported extension and platforms

Tool	FreeCommercial	PlaceTransitionNets	HighlevelPetriNets	PetriNetswithTime	StochasticPetriNets	ObjectorientedPNs	StochasticSymmetricNets	HybridPetriNets	OtherClasses	Windows	SunOS	Linux	MacintoshMacOSX	HPHPUX	DigitalUNIX40	SiliconGraphicsIRIX	Java	Solaris	otherrequirements
SIPN-Editor	FC	US	US	US	US	US	US	US	Interpreted Petri Nets	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
SNAKES	FC	S	S	US	US	US	US	US	PBC and M-nets family	AV	AV	AV	AV	AV	NA	AV	NA	NA	/
Snoopy	FC	S	US	S	S	US	US	US	Continuous Petri Nets	NA	AV	AV	AV	NA	NA	NA	NA	NA	/
StpnPlay	FC	US	S	US	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
TAPAAL	FC	US	S	S	US	US	US	US	/	AV	AV	AV	AV	AV	NA	AV	AV	NA	/
The Model checking Kit	FC	S	S	US	US	US	US	US	/	NA	AV	AV	NA	NA	NA	NA	NA	NA	/
Tina	FC	S	US	S	US	US	US	US	/	AV	AV	AV	AV	NA	NA	NA	NA	NA	/
Visual Object Net ++	FC	S	US	S	US	US	US	US	Hybrid Dynamic Nets and Hybrid Object Nets	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
VisualPetri	FC	S	US	US	US	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
WebSPN	FC	US	US	US	S	US	US	US	Non Markovian Stochastic Petri Nets	NA	NA	AV	NA	NA	NA	NA	AV	NA	/
Wolfgang	FC	S	S	US	US	US	US	US	/	NA	NA	AV	AV	NA	NA	NA	AV	NA	/
WoPeD	FC	S	US	US	US	US	US	US	Workflow Nets	NA	NA	NA	NA	NA	NA	NA	AV	NA	/
XRL/flower	FC	US	S	US	US	US	US	US	WF Nets	AV	NA	NA	NA	NA	NA	NA	AV	NA	/
Yasper	FC	S	US	S	S	US	US	US	/	AV	NA	NA	NA	NA	NA	NA	NA	NA	/
YAWL	FC	US	S	US	US	US	US	US	/	NA	NA	NA	NA	NA	NA	NA	AV	NA	/

Table 4.4: Part 4 of filling data:PN tool supported extension and platforms

Tool	FreeCommercial	GraphicalEditor	TokenGameAnimation	FastSimulation	Codegeneration	CondensedStateSpaces	StateSpaces	InterchangeFileFormat	RapidPrototyping	RareEventSimulation	Modelchecking	StructuralAnalysis	PlaceInvariants	TransitionInvariants	WorkflowManagementSystem	NetReductions	SimplePerformanceAnalysis	AdvancedPerformanceAnalysis
ALPHA/Sim	C	AV	AV	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
MISS-RdP	C	AV	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Nevod	C	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Opera	C	AV	AV	AV	NA	AV	AV	AV	NA	NA	NA	AV	AV	AV	NA	AV	AV	AV
Simulaworks petri Net Simulator	C	AV	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Artifex	C&AD	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	AV
COSA BPM	C&AD	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA
ExSpect	C&AD	AV	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	AV	NA	AV	AV
FLOWer	C&AD	AV	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA
F-net	C&AD	AV	AV	AV	NA	NA	AV	NA	NA	NA	NA	AV	AV	AV	NA	NA	AV	AV
Kontinuum	C&AD	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	AV	NA
PACE	C&AD	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA
Petri .NET Simulator	C&AD	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Petri Net Toolbox	C&AD	AV	AV	AV	NA	NA	AV	AV	NA	NA	NA	AV	AV	AV	NA	NA	AV	AV
PROTOS	C&AD	AV	NA	AV	NA	NA	NA	NA	AV	NA	NA	AV	NA	NA	AV	NA	AV	NA
SimHPN	C&AD	AV	NA	AV	NA	NA	AV	NA	NA	NA	NA	AV	AV	AV	NA	NA	AV	NA
GDToolkit	C&FD	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
GreatSPN	C&FD	AV	AV	AV	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Income Suite	C&FD	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA
MISTA	C&FD	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Netlab	C&FD	AV	AV	AV	NA	AV	AV	AV	NA	NA	NA	AV	AV	AV	NA	NA	NA	NA
SPNP	C&FD	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
SYROCO	C&FD	AV	NA	AV	AV	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
TimeNET	C&FD	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	AV
AlPiNA	FC	AV	NA	NA	NA	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
ARP	FC	NA	NA	AV	NA	NA	AV	NA	NA	NA	NA	AV	AV	AV	NA	NA	AV	NA
CoopnBuilder	FC	AV	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Cosmos	FC	NA	NA	AV	NA	NA	NA	AV	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA
CPN-AMI	FC	AV	NA	AV	NA	AV	AV	AV	NA	NA	NA	AV	AV	AV	NA	NA	NA	NA

Table 4.5: Part 1 of filling data: PN tool functionalities

Tool	FreeCommercial	GraphicalEditor	TokenGameAnimation	FastSimulation	Codegeneration	CondensedStateSpaces	StateSpaces	InterchangeFileFormat	RapidPrototyping	RareEventSimulation	Modelchecking	StructuralAnalysis	PlaceInvariants	TransitionInvariants	WorkflowManagementSystem	NetReductions	SimplePerformanceAnalysis	AdvancedPerformanceAnalysis
PNetLab	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	AV	AV	AV	NA	NA	AV	NA
PNML Framework	FC	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
PNSim	FC	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA
PNtalk	FC	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
Poses++	FC	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Predator	FC	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	AV	AV	NA	NA	NA	NA
PROD	FC	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA
ProM framework	FC	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA
QPME	FC	AV	NA	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
Renew	FC	AV	AV	AV	NA	NA	NA	AV	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA
SamaTulyata	FC	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SIPN-Editor	FC	AV	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SNAKES	FC	NA	NA	AV	NA	AV	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	AV	NA
Snoopy	FC	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
StpnPlay	FC	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
TAPAAL	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
The Model-checking Kit	FC	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA
Tina	FC	AV	AV	NA	NA	AV	AV	NA	NA	NA	NA	NA	AV	AV	NA	NA	NA	NA
Visual Object Net++	FC	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA
VisualPetri	FC	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
WINSIM	FC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
Wolfgang	FC	AV	AV	NA	NA	NA	AV	AV	NA	NA	NA	NA	NA	NA	AV	NA	NA	AV
WoPeD	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	AV	NA	NA	AV	NA	NA	NA
XRL/flower	FC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA
Yasper	FC	AV	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	AV	AV	AV	NA
YAWL123	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA
PNetLab	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	AV	AV	AV	NA	NA	AV	NA
PNML Framework	FC	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
PNSim	FC	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA

Table 4.6: Part 2 of filling data:PN tool functionalities

Tool	FreeCommercial	GraphicalEditor	TokenGameAnimation	FastSimulation	Codegeneration	CondensedStateSpaces	StateSpaces	InterchangeFileFormat	RapidPrototyping	RareEventSimulation	Modelchecking	StructuralAnalysis	PlaceInvariants	TransitionInvariants	WorkflowManagementSystem	NetReductions	SimplePerformanceAnalysis	AdvancedPerformanceAnalysis
PNetLab	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	AV	AV	AV	NA	NA	AV	NA
PNML Framework	FC	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
PNSim	FC	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA
PNtalk	FC	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
Poses++	FC	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Predator	FC	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	AV	AV	NA	NA	NA	NA
PROD	FC	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA
ProM framework	FC	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA
QPME	FC	AV	NA	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
Renew	FC	AV	AV	AV	NA	NA	NA	AV	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA
SamaTulyata	FC	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SIPN-Editor	FC	AV	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SNAKES	FC	NA	NA	AV	NA	AV	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	AV	NA
Snoopy	FC	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
StpnPlay	FC	AV	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA
TAPAAL	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
The Model-checking Kit	FC	NA	NA	NA	NA	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA
Tina	FC	AV	AV	NA	NA	AV	AV	NA	NA	NA	NA	NA	AV	AV	NA	NA	NA	NA
Visual Object Net++	FC	AV	AV	AV	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA
VisualPetri	FC	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
WINSIM	FC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV
Wolfgang	FC	AV	AV	NA	NA	NA	AV	AV	NA	NA	NA	NA	NA	NA	AV	NA	NA	AV
WoPeD	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	AV	NA	NA	AV	NA	NA	NA
XRL/flower	FC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA
Yasper	FC	AV	AV	AV	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	AV	AV	AV	NA
YAWL123	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA
PNetLab	FC	AV	AV	NA	NA	NA	NA	AV	NA	NA	NA	AV	AV	AV	NA	NA	AV	NA
PNML Framework	FC	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
PNSim	FC	AV	AV	NA	NA	NA	NA	NA	NA	NA	NA	AV	NA	NA	NA	NA	NA	NA

Table 4.7: Part 3 of filling data:PN tool functionalities

4.3 Statistical Analysis of PN software tools

In this section, a statistical study to evaluate the PN software feature availability, we aim to highlight the common supported features for these tools and the little-known one as well as. To establish this analysis, we use the SPSS platform that helps us to obtain statistical results for to following requests:

- List the available tools for every petri nets extensions.
- List the supported operating systems offered by these tools.
- Summarize the available petri nets tool licence.
- Highlighted the available functionalities provided by these tools.
- List the available tool that provide:
  - Code generation.
  - Model checking.
  - Net reduction.

4.3.1 PN Extensions and supported tools frequencies

Popular PN extension frequencies

Request1 : List the available tools for popular petri nets extensions.

We use analyse>frequencies option from SPSS to request the percent of tool cases for each extension. Table 4.8 gives for every Petri nets extension the percent of tools that support this extension. We choose to generate a bar chart to effectively illustrate this result (see Figure 4.3).

Other Petri Classes frequencies

Request 2: List the available tools for unpopular petri nets extensions

Table 4.8: PN extension frequencies

		Responses	
		N	Percent
PNClasses	Place/Transition Nets	59	32.4%
	High-level Petri Nets	49	26.9%
	Petri Nets with Time	41	22.5%
	Stochastic Petri Nets	25	13.7%
	Object-oriented PNs	5	2.7%
	Stochastic Symmetric Nets	1	0.5%
	Hybrid Petri Nets	2	1.1%
Total		182	100.0%

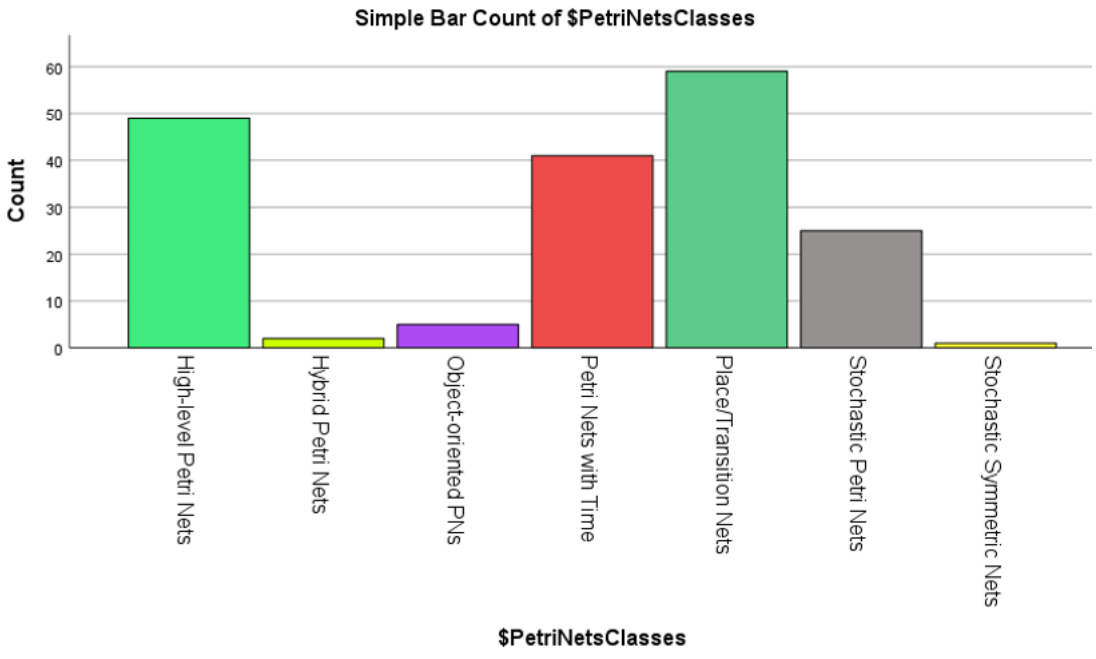

Figure 4.3: Simple Bare graph: Popular PN extension frequencies

Table 4.9: Other PN frequencies

				Other Classes
Tool	ITS Tools	1		PNML
		Total	N	1
	Nevod	1		Inhibitor Petri nets
		Total	N	1
	P3	1		Upgraded Petri Nets
		Total	N	1
	Petri Net Toolbox	1		Generalized Stochastic Petri Nets
		Total	N	1
	Petri-lld	1		Elementary Nets
		Total	N	1
	Petri-Net Kernel	1		DAWN-Nets
		Total	N	1
	Petruchio	1		Transfer Petri Nets
		Total	N	1
	PNML Framework	1		PNML Core Model
		Total	N	1
	Predator	1		hierarchical Petri Nets
		Total	N	1
	QPME	1		(Hierarchical) Queueing Petri Nets
		Total	N	1
	SIPN-Editor	1		Interpreted Petri Nets
		Total	N	1
	SNAKES	1		PBC and M-nets family
		Total	N	1
	Snoopy	1		Continuous Petri Nets
		Total	N	1
	Visual Object Net ++	1		Hybrid Dynamic Nets and Hybrid Object Nets
		Total	N	1
	WebSPN	1		Non Markovian Stochastic Petri Nets
		Total	N	1
	WoPeD	1		Workflow Nets
		Total	N	1
	XRL/flower	1		WF Nets
		Total	N	1
	Total	N		17

Some petri net extensions are supported by only one tool; to obtain the related tools that support this PN extension, we use Case Summaries SPSS option that offers a list of filtered PN cases with the corresponding tool. In total, 17 petri tools are selected as a result of this request; Table 4.9 shows the filter result.

Object Petri Nets tools Report

Request 3: list tools for OPN extensions.

To get a report about a particular Petri net Class tools (here we select OPN), We use execute case summaries from analyze reports, after that we filter cases using select option to envisage only ones where OPN is present. Table 4.10

Table 4.10: Object Petri Nets tools Report

		Tool	Free/Commercial
1		Artifex	C&AD
2		CoopnBuilder	FC
3		JFern	FC
4		JSARP	FC
5		Renew	FC
Total	N	5	5

Table 4.11: Supported environment Frequencies

		Responses	
		N	Percent
$SupportedEnviroment	Windows	44	25.7%
	SunOS	23	13.5%
	Linux	36	21.1%
	Macintosh, Mac OS X	15	8.8%
	HP, HP-UX	12	7.0%
	Digital, UNIX 4.0	1	0.6%
	Silicon Graphics, IRIX	7	4.1%
	Java	32	18.7%
	Solaris	1	0.6%
Total		171	100.0%

shows the available tools that support OPN extension.

4.3.2 The tools Frequencies of supported environments

Request 4: Supported Environment Frequencies.

To answer to the following question: The available tools support multiple operating systems? we use frequencies option from SPSS to figure the percent of occurrence for every selected environment in case studies(see Table 4.11). This result is depicted graphically using Pie chart graph from in SPSS, see Figure 4.4

4.3.3 Tools licence

Request: Summarize the available petri nets tool licence.

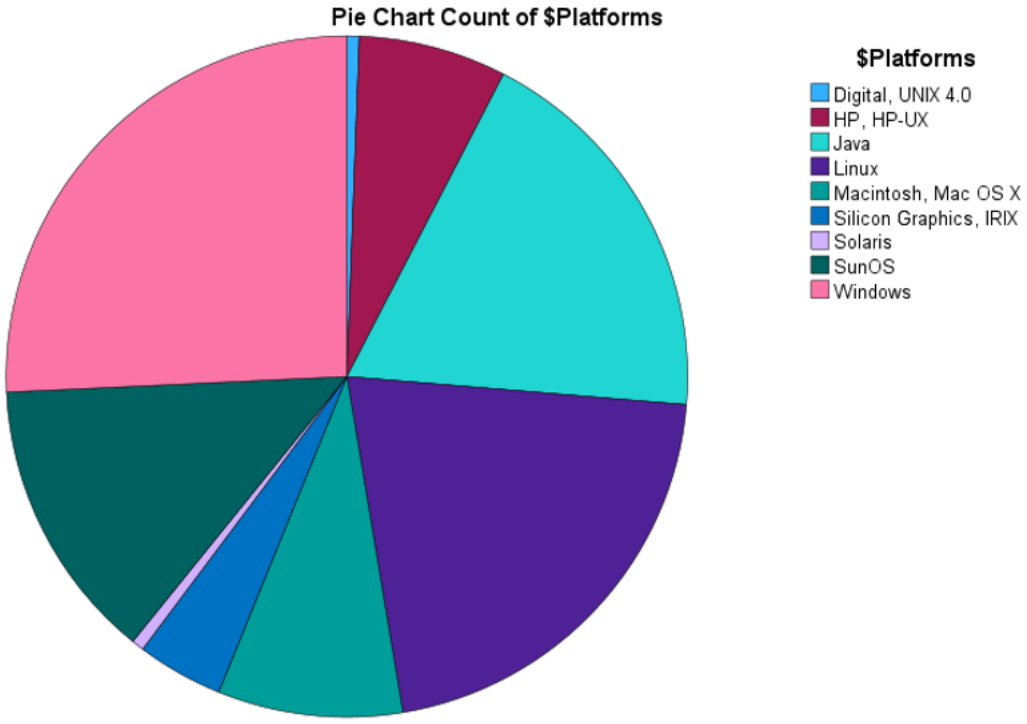

Figure 4.4: Pie Chart environment Frequencies

Table 4.12: Tools licence

		Frequency	Percent
Valid	C	5	5.6%
	C&AD	11	12.2%
	C&AF	8	8.9%
	FC	66	73.3%
	Total	90	100.0%

To obtain an overview of tool licence of the existence PN software solution, we have request frequency of the licence distribution. The result is ordered starting from the commercial to the free charge ones. The Table 4.12 listed the frequency of each licence per all cases. Figure 4.5 depicted this result in Pie chart graphs.

4.3.4 Overview of The available functionalities

Request Envisage available Functionalities

The available tool provides sufficient options to support designers' requirements? To answer this question we execute frequencies option of available functionalities offered by the filled tools, Table 4.13 shows the result of this

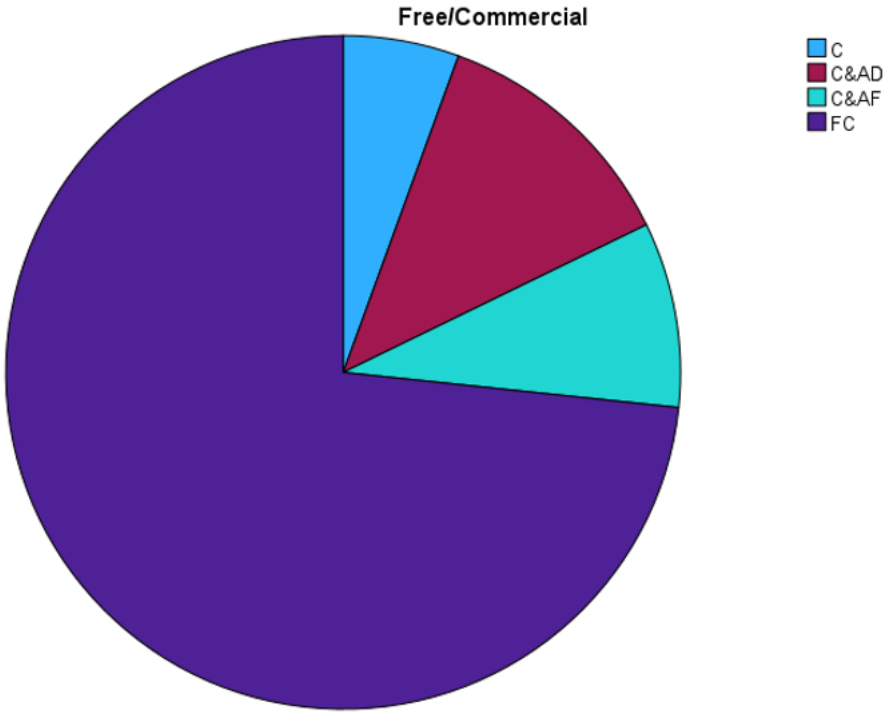

Figure 4.5: Tools licence Pie chart graph

request. A bare graph is used to clearly envisage the achieved result see Figure 4.6

4.3.5 The support of important features

Finding the most required functionalities in a target tool effect the selection of this tool by designers. Based on the filling PN cases, we expose the listed of available tool that offers: net reductions, model checking, code generation and interchangeable file.
- Table listed petri net tools that provides a mechanism of code generation; see Figure 4.15.
- Table listed petri net tools that offers a Model checking functionality ; see Figure 4.14.
- Table listed petri net tools that allows users to reduce a net by more simple one; see Figure 4.16.
- Table listed petri net tools that support the interchgeable file format to deployed petri nets models ; see Figure 4.17.

Table 4.13: Tools Functionalities

		Responses	
		N	Percent
$Functionalities	Graphical Editor	69	17.8%
	Token Game Animation	46	11.9%
	Fast Simulation	41	10.6%
	Code generation	5	1.3%
	Condensed State Spaces	18	4.7%
	State Spaces	35	9.0%
	Interchange File Format	33	8.5%
	Rapid Prototyping	8	2.1%
	Rare Event Simulation	1	0.3%
	Model checker	6	1.6%
	Structural Analysis	22	5.7%
	Place Invariants	20	5.2%
	Transition Invariants	19	4.9%
	Workflow Management System	13	3.4%
	Net Reductions	7	1.8%
	Simple Performance Analysis	26	6.7%
	Advanced Performance Analysis	18	4.7%
Total		387	100.0%

Table 4.14: PN model cheking tools list

		Tool	Free/Commercial
1		ePNK	Free
2		Maria	Free
3		PEP	Free
4		PROD	Free
5		ProM framework	Free
6		The Model-checking Kit	Free
Total	N	6	6

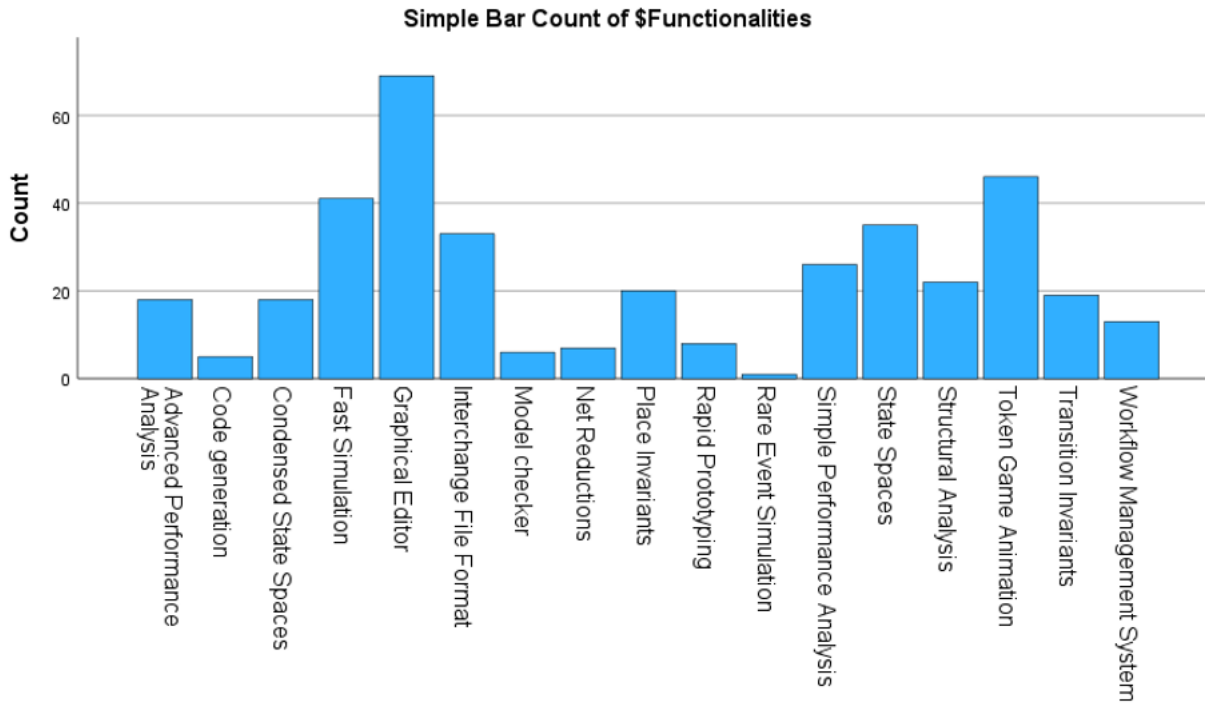

Figure 4.6: Functionalities Simple Bar Graph

Table 4.15: PN code generation supported tools list

	Tool	Free/Commercial
1	ALPHA/Sim	Commercial
2	SYROCO	Commercial academic Free
3	CoopnBuilder	Free
4	PEP	Free
5	SIPN-Editor	Free
Total N	5	5

Table 4.16: PN net reduction supported tools list

	Tool	Free/Commercial
1	Opera	Commercial
2	PACE	Commercial academic discounts
3	Helena	Free
4	INA	Free
5	mist	Free
6	mist2	Free
7	Yasper	Free
Total N	7	7

Table 4.17: Interchangeable file supported tools list

	Tool	Free/Commercial
1	Opera	Commercial
2	Petri Net Toolbox	Commercial academic discounts
3	Netlab	Commercial academic Free
4	SYROCO	Commercial academic Free
5	AlPiNA	Free
6	Cosmos	Free
7	CPN-AMI	Free
8	CPN Tools	Free
9	Disc Software Platform	Free
10	HiQPN-Tool	Free
11	INA	Free
12	IOPT-Tool	Free
13	JARP	Free
14	JFern	Free
15	Mercury	Free
16	P3	Free
17	PEP	Free
18	Petri Net Kernel	Free
19	Petruchio	Free
20	Platform Independent Petri Net Editor 2	Free
21	PNEditor	Free
22	PNetLab	Free
23	PNML Framework	Free
24	Predator	Free
25	QPME	Free
26	Renew	Free
27	SNAKES	Free
28	StpnPlay	Free
29	TAPAAL	Free
30	Wolfgang	Free
31	WoPeD	Free
32	Yasper	Free
33	YAWL	Free
Total N	33	33

4.4 Discussion

We appreciate the effort that has been made in the development of the numerous petri nets software solutions mentioned in this chapter. Usually, designer confuse in the selection of a PN modeling product. As it is showed in the analysis result, the majority of these tools support the low petri nets extension (32.4% of tools support Place/Transition Nets), few of them support high level extension (e.g., 2.7% of tools support Object PN). Other factors that effect the choose of these tools are licence distribution and the support of operating systems; we find in this research that 73.3% of tools are free charge and only 5.6% need to be purchase; where the windows take lion's share of the support environment with 25% .

Whatever the tool, the richness of tools by functionalities is another criteria to select the appropriate modeling product (how far that these tools assist designers in the design and analyse stage of a PN application)? When designers want to select a petri net modeling tool, they are usually interested in ones that have capabilities of simplification, verifications, implementation and interchangeable file functionalities.

The totally of tools that offers the code generation feature are only 5 (*ALPHA/Sim*, *SYROCO*, *CoopnBuilder*, *PEP*, *SIPN-Editor*); where 3 of theme are free charge application and one is a commercial product and other one is commercial solution with a free academic use possibility.

- The net reduction feature is offers by only 7 tools (*Opera*, *PACE*, *Helena*, *INA*, *mist*, *mist*, *mist2* and *Yasper*); where 5 of them are free charge application and one (Opera) is a commercial product and other one(*PACE*) is a commercial solution with a free academic use possibility.

- Concerned the model checking feature, *ePNK*, *Maria*, *PEP*, *PROD*, *ProM framework* and *The Model-checking Kit* offers this functionality.

- The tools that offer the capability to provide models with an interchangeable feature are in total 17 solutions; where 14 of them are a free charge product, one is a commercial and one have an academic discount and two are free for academic use.

We are interested in this thesis to the tools that support object oriented and multi agents systems. Unfortunately, there is no implemented tool for agent petri nets tool, designer can use OPN to design MAS (as it well known in many researches, OPN is used also to design multi agent systems). We find only five tools allow designers to specify Object Petri Net models (*Artifex*, *CoopnBuilder*, *JFern*, *JSARP* and *Renew*), all these tools are distributed in a Free charge licence except *Artifex*.

4.5 Conclusion

Having a suitable modeling solution, that master the complexity of modeling systems is required in the selection of appropriate products. Tools that are rich by functionalities encourage user to adopt them in the development process of their applications. The results showing in this work provide a helpful reference about the PN implemented software solutions; we emphasis several points in this research:

- We have exposed the available implemented solution for the selected PN extension.
- We have highlighted the Operating systems that support these tools, and the commercial distribution of each one.
- We have exposed the functionalities offered by these tools and we have made a focus on ones that provide brilliant functionalities, as code generation, net reduction, and interchangeable file and model checking.

The proposition of a new extension still a challenging issue for designers when no software support is developed. The possibility to extend a PN tool in the future by new functionalities will affect also the productivity of developments. Selecting a development methodology that promotes the productivity in development is required, MDE approaches seem more suitable for this mission.

Chapter 5

MDE Workbenches Evaluation

Contribution:Towards a Simplified Evaluation of Graphical DSL Workbenches [47]

5.1 Introduction

Simplifying the intricacy of systems through abstracting mechanism is one of the core mechanism in the field of computer science. By employing abstraction, the reuse of computer sciences artifact is heightened, thereby implies subsequent in tool productivity. Developing an application from scratch would be an overload for developers. MDSE assists programmers in the development of their software products by providing a panorama of modeling software solutions where abstraction and automatic transition to implementation are the main pillar foundation of these solutions [7].

Nevertheless, these solutions vary in their potential to guarantee a complete or partial transition from an abstract representation to a corresponded implementation (wasting efforts if designers select no convenient MDE solutions in the concretization of their researches).

MDE also aims that non engineering expert could use this technology to develop their own applications by only focusing on modeling stage [48, 49].

In this chapter, we present a comparative study that makes the focus on capabilities of three picked eclipse modeling solutions in the developments

of modeling products; examining how far these tools could really aid domain experts without software engineering backgrounds in the production of their applications will give them a great helping. In section 2, the three brilliant MDSE solutions are presented as well as the methodology adopted in this study is illustrated. In section 3, the comparative study is showcased, and in section 4, the result of this comparison is highlighted and finally we conclude this chapter.

5.2 Picked tools Overview

It is well known that the Eclipse [**?**] platform has a large users community, we have picked in this research some important modeling projects released by Eclipse foundation: Sirius [50, 51], Cinco [52, 33] and, Graphical modeling framework [53]. In the following, a short description of these solutions will be presented:

The graphical modeling framework (GMF) is an Ecore-based solution to generate graphical syntaxes for a new domain model; this solution is based on top of two the modeling framework: the EMF (Eclipse modeling framework) [**?**] and the drawing framework called GEF (graphical editing framework) [54]. To use this solution under the eclipse ecosystem, the user should use the option "install new software" and select the website that includes the GMF modeling project then, follow a dashboard that assists designers to get an end GMF editor after defining a series of GMF models. Figure 5.1 represents the GMF product development process. We should note here that for each editor development process, we have classified its steps according to MDA layers.

Sirius: is an eclipse project that assists designers to build a graphical tool for domain specific system. The first stable release was been in 2013, many releases are available now; two distributions have been deployed to develop desktop applications and also web ones [55]. The main objective of this modeling solution is to give a helping hand for designers in managing the complexity of systems; it allows them to define a meta-model for their applications and provides for them many possibilities to represent the models in different formats [51]. The first help allows the designers to tailor a specific domain model for their applications and the second help promotes a reliable communication between project team through various representations. Figure 5.2 represents the Sirius product development process.

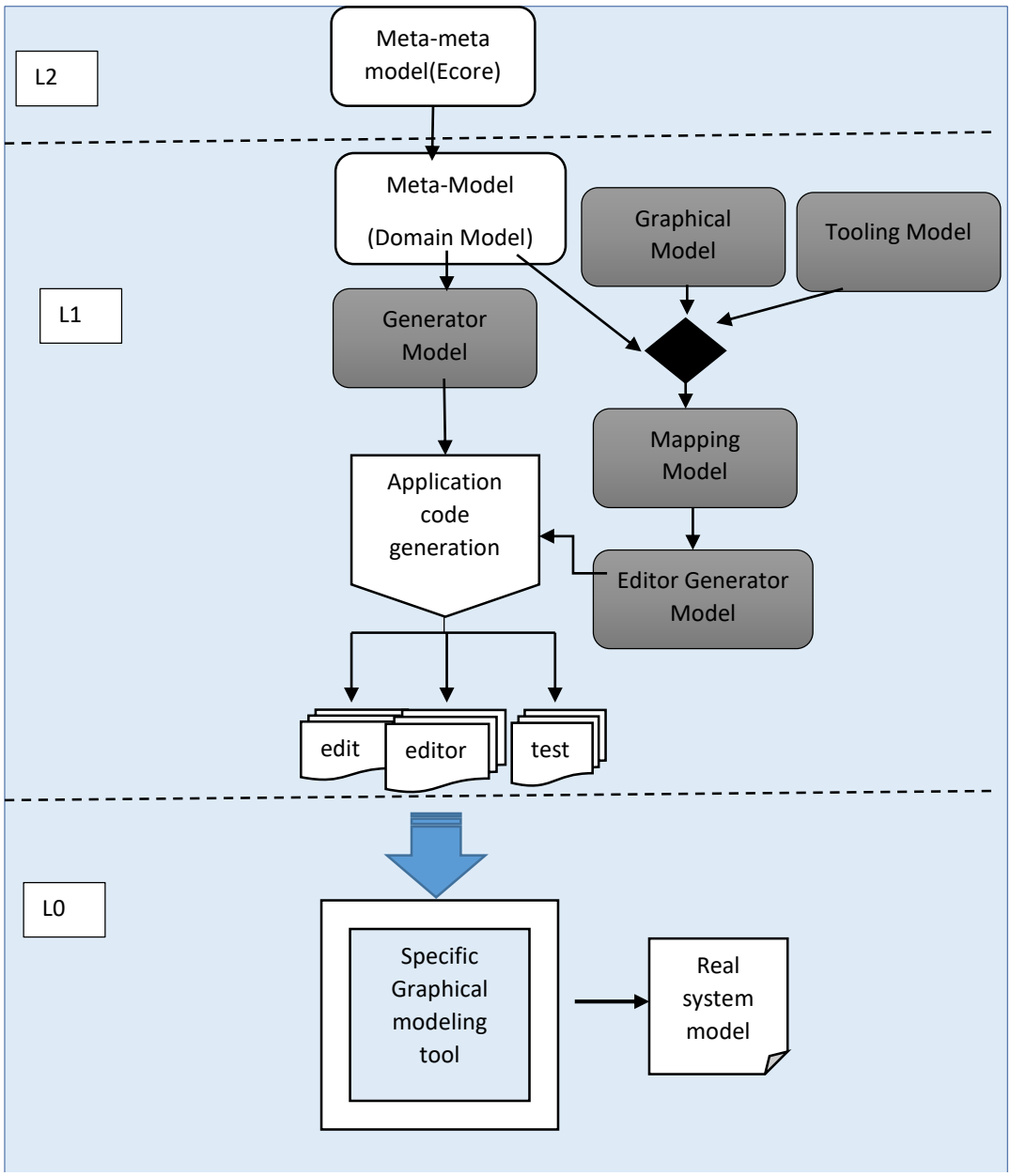

Figure 5.1: GMF product development process

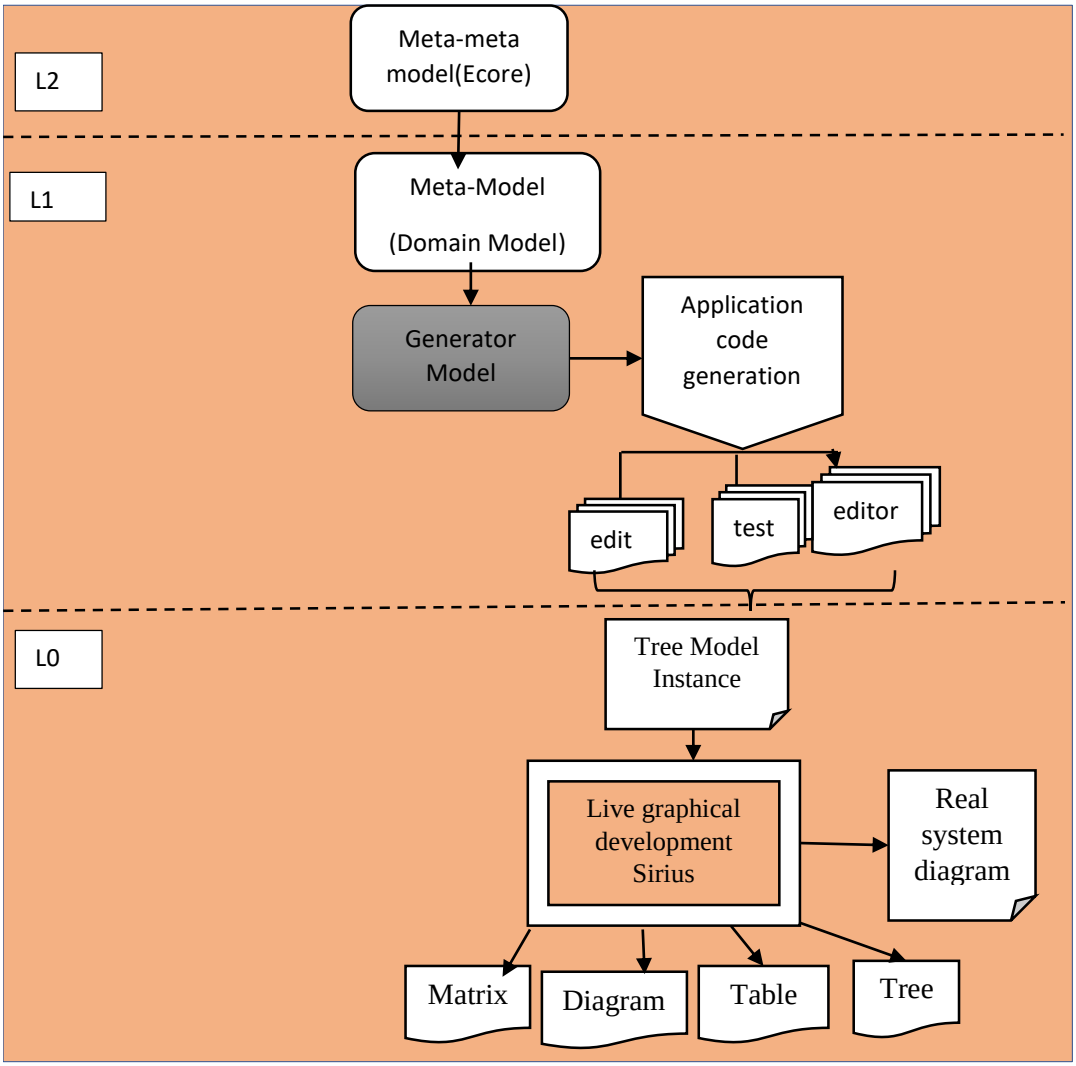

Figure 5.2: The Sirius product development process

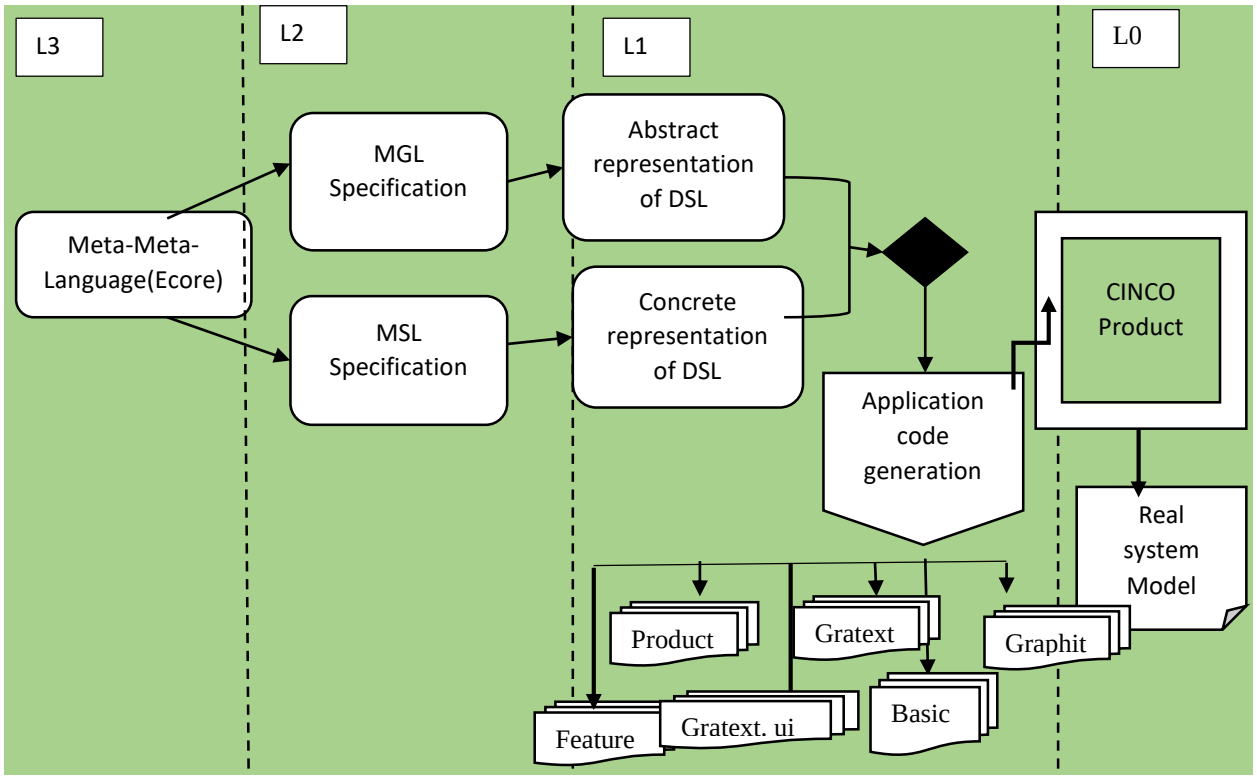

Figure 5.3: The Cinco product development process

The Cinco meta tooling suite [52]: is a modeling and generation solution based on top of two sub frameworks, the EMF and the Graphiti framework [56]. The main purpose of this approach is to streamline as possible the development of application by minimizing the complexities related to EMF and graphiti [52, 33, 48, 57]. Figure 5.3 represents the Cinco product development process.

We aim in this chapter to propose an analysis of the above solutions based on specific criteria. A short description of these criteria in the following points:

1 Abstraction Level: we aim here to evaluate the modeling approach adopted by each tools, and how far that these approaches assist in the increase of the abstraction level of domain specific application.

2 Specialised capacities: our aim here to expose tool's capabilities to tailor a specific user application.

3 Simplicity supported by tools: To what extent that the experts without a software engineering backgrounds could get easily a final modeling prod-

uct for their purpose.

4 Productivity: The strength of the support of an automatic code generation (a partial guarantee or a holistic one).

5.3 Workbenches evaluation

The evaluation of these tools necessities the selection of a common meta-model; we select APN as a common case study in our evaluation process. As we mentioned in the first chapter that this extension does not have a software support to assist domain experts in designing, simulating and verifying their APN models. Statistic and semantic aspects corresponding to the domain specific language should be covered by the evaluation of the picked tools. The comparison is established by creating three editors for the same formal specification (here we have selected APN); we called them respectively: *AgentPetriGMFEditor*, *APNSiriusModel*, *CincoAgentPetriEditor*.

As an example, we take a simple one of print job illustrated in the contribution of[6] and we edit the model with our DSLs Editors. CPU and IMP are two agents participating in the Print-Job service. The movement of Agents in the net is controlled by conditions-functions related to transition T1 and T2[47].

The evaluation is established regarding to the criteria proposed in the above section.

5.3.1 Abstraction level

To evaluate this criterion, we expose the main modelling approach adopted in the development of the three APN editors. As we select agent petri nets formalism as a common use case, the proposition of an abstract representation for this formal modeling language should respect the supported modeling languages supported by each modeling workbench.
The development under GMF necessities the definition of a series of GMF models; figure 5.1 depicted the required models see level L2. We start by specifying a meta-model using Ecore format, this specification is depicted the proposed APN-Ecore concepts and the relation between these concepts (see figure 5.4 the model depicted in tree based view).

Concerned Sirius, the tool also adopts Ecore as a meta-language to

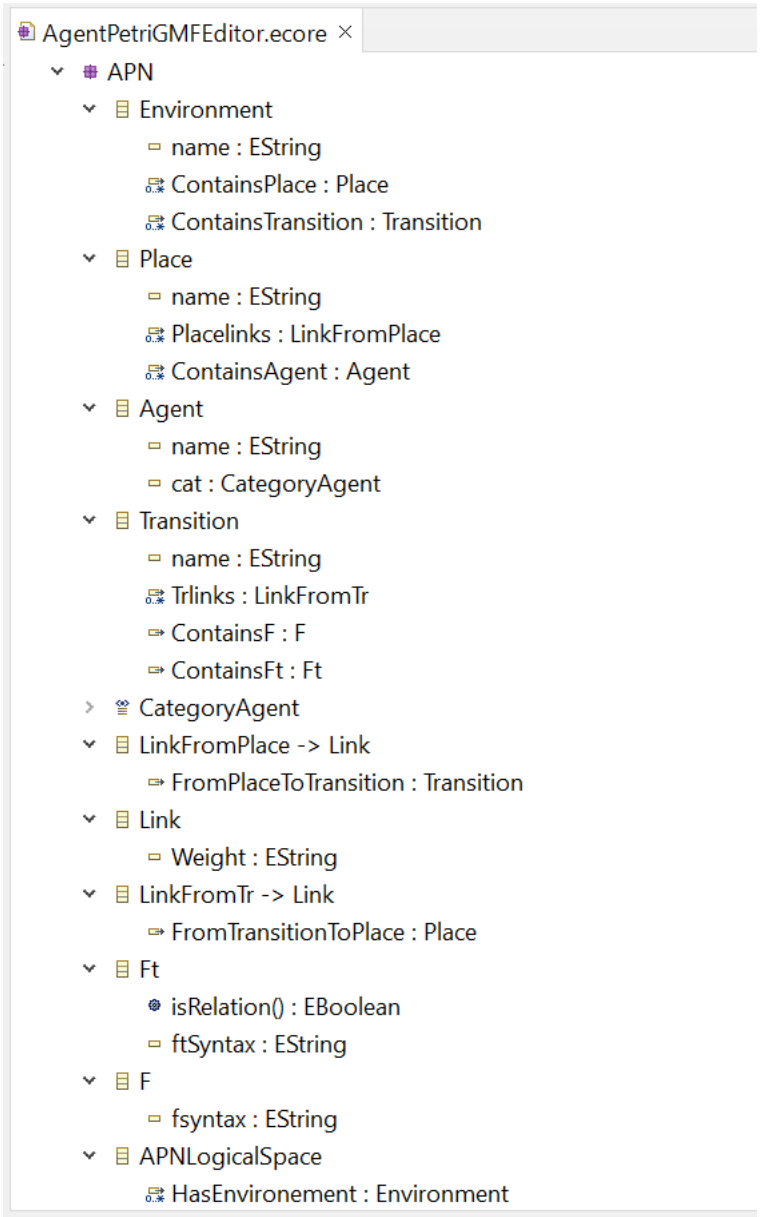

Figure 5.4: The APN meta model in Ecore

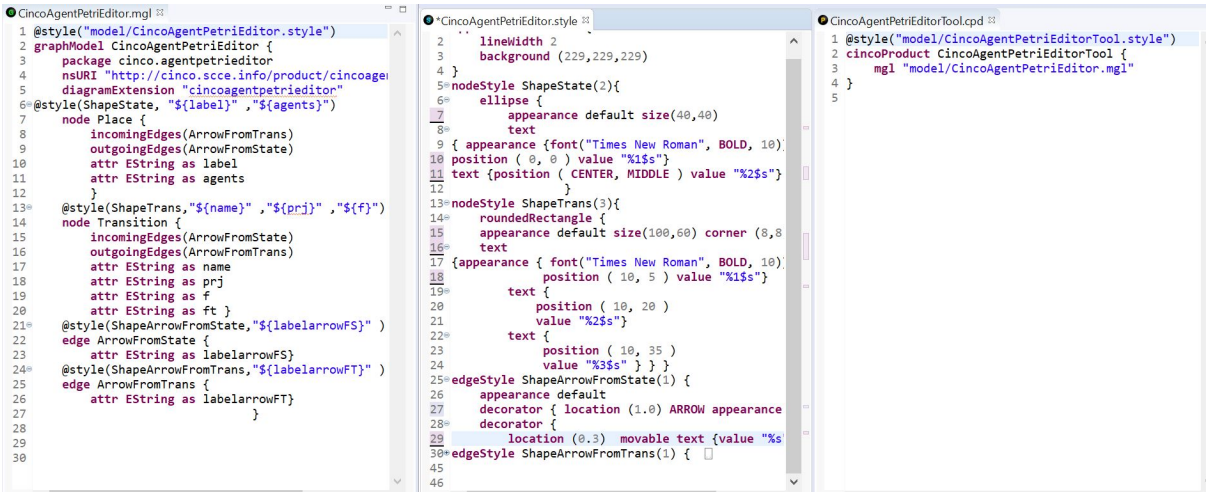

Figure 5.5: Cinco APN meta models

specify meta-models, so we have taken the same meta model for the second Editor. In our contribution [58] the development of an editor, for APN using GMF is widely detailed (the steps required to develop this plugin are explained in details).
Cinco differs from its counterpart by supporting textual meta-language; to specify meta-models for any Cinco product the meta graph language is recommended to be used [33](see figure 5.3 , layer one).
Figure 5.5 left side, shows the proposed model for APN using the MGL notation (the meta language and the APN meta-model specified in Cinco will be thoroughly discussed in the upcoming chapter).

5.3.2 Specialised capacities

we aim here to expose tools capabilities to tailor a specific user application; practically how far these solutions provide expressiveness notation for each abstract representation.
The GMF solution provides a tree based graphical model to enable designers to specify shapes for each model element and allows them to generate a tooling model corresponding to the meta-model elements. Two models in this step are needed to be defined: the graphical (dedicates to the specification of graphical characteristic as shape, label, size) and the tooling (devoted to allow users to instantiate model elements graphically, these tools are arranged in a generated pallet). See figure 5.6 the six GMF meta-models are depicted.

Sirius from its part encourages designers to specify the graphical representation of model elements using odesign specification. Before designers could specify graphical representation, they should generate the basic implementa-

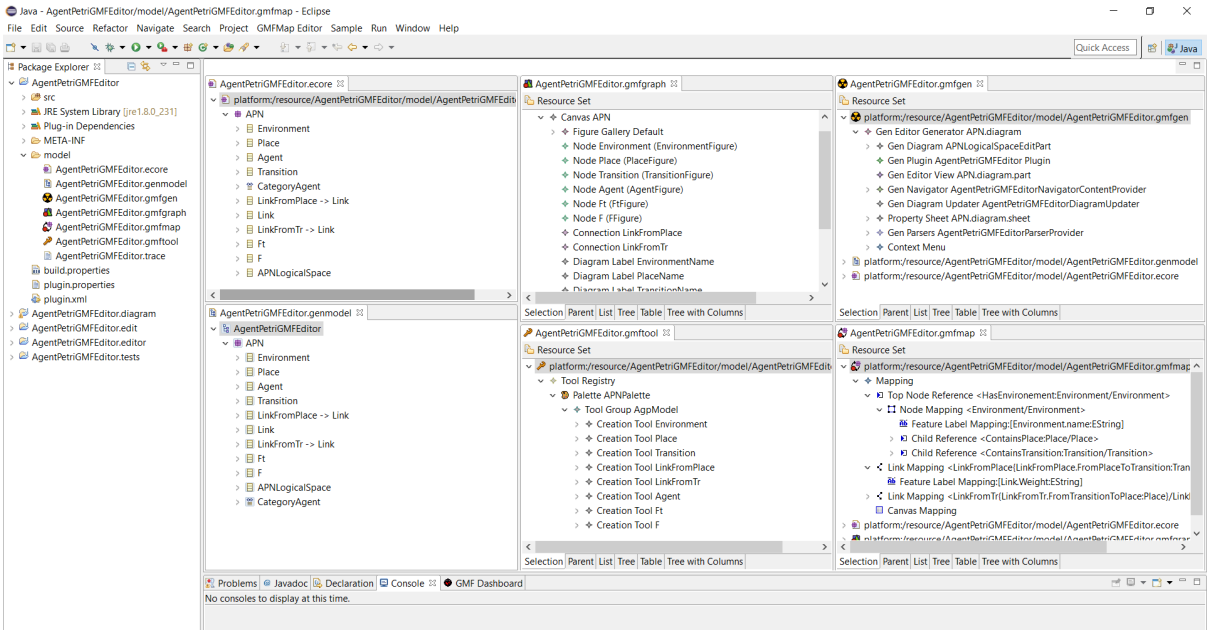

Figure 5.6: APN GMF meta-models
[47]

tion corresponding to the defined meta-model, then create an instance model using a tree based representation. Designers specify the graphical and the corresponding tolling for each model element just in development time where a model instance is required for this mission; this feature allows designers to make changes in the graphical representation in any time. Sirius also enables the designers to visualise graphical model using diagram, metrics, table, tree based editor. See figure 5.8 the Sirius final user product (APNSiriusModel). Having a suitable graphical notation for model elements affects the choice of the more privileged workbench, Cinco form its part enables designers to precise graphical representation using MSL language and, provides a full generation of functionalities related to this graphical appearance for each model element. Figure 5.9 shows the final APN Cinco product(CincoAgentPetriEditor).

5.3.3 Simplicity supported by tools

To what extent that the domain experts without a software engineering backgrounds could get easily a final modeling product for their purpose with little efforts. During the development of our APN Editor using GMF and Sirius, we have noted that for designers with a little experience with these technologies the result could not be achieved easily. A node mapping conflict is usually caused a challenging issue for designers; the generated mapping model often has a disaccording between the model element its tooling, its graphical representation and its associated label, designer could waste time fixing these

errors. In addition, defining six meta-model to get a graphical editor (see figure 5.6), not considered an easiest way for non-engineers designers. In contrast, the newest technology adopted by Cinco promotes the declaration of this mapping textually witch really guarantees a correct according between the meta model elements and its appearance. Domain expert should only master the terminology to edit textually the meta model and its related adequate graphical appearance. Details related to the mapping is automatically taking on consideration by the platform.

5.3.4 Productivity

This criterion is related principally to the code generation mechanism and the reuse of library. We aim to evaluate if domain expert designers can benefice form a holistic code generation possibility or a partial one (designer should add extra code or the tool guarantees an automatic transformation from an abstract representation to an adequate implementation) and how far that the generated code could be easily used by other application. In Cinco, the specification of many meta models for the same product is supported and also the reuse of the same graphical model by many other meta models is enabled; this feature has been applied in The research of[59]. The both GMF and Sirius support a partial code generation mechanism, designers should add extra code in order to get a sophisticated graphical editor (rich by desired designers functionalities like the implementation of button events or menu actions). In contrast Cinco support this feature automatically (more details will be presented in the next chapter). In addition, the implementation of semantic, it is completely a designer task for GMF and Sirius designers. On the other hand, Cinco assists designers by offering an API that streamlines the incorporation of model semantics (generates methods that help designers to browse graphically model elements and retrieve information related to incoming the outcoming nodes, and enable them to make modification); this API support is an advanced feature in this platform.

5.4 Analyzing results

A real experiment with these modeling solutions, helps us to examine the announced functionalities supported by these tools, all these solutions are targeted the creation of DSL software tools. Figures: 5.7 ,5.8 and 5.9 represent

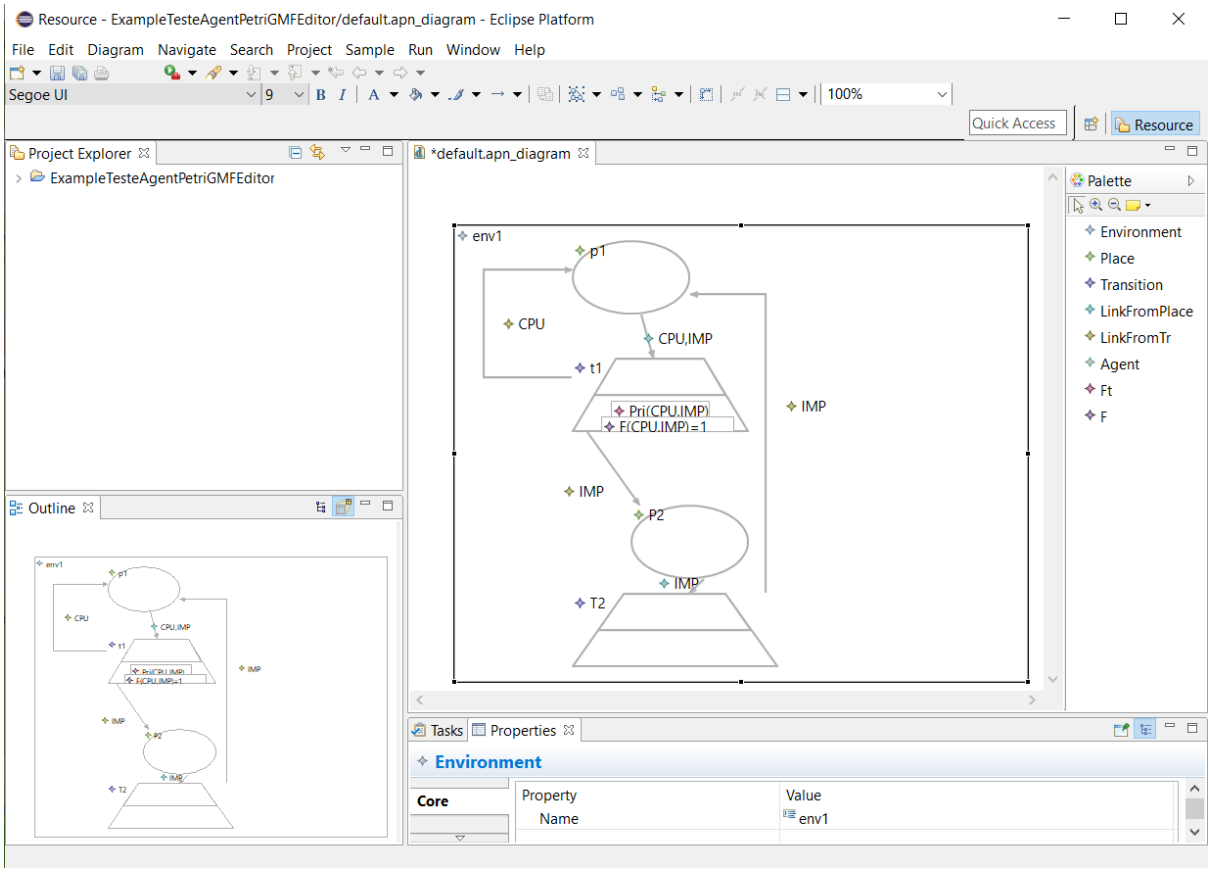

Figure 5.7: The GMF final user product (AgentPetriGMFEditor) [47]

respectively the final users products for each solutions.

Table5.1 gives a summary of the evaluation outcomes.

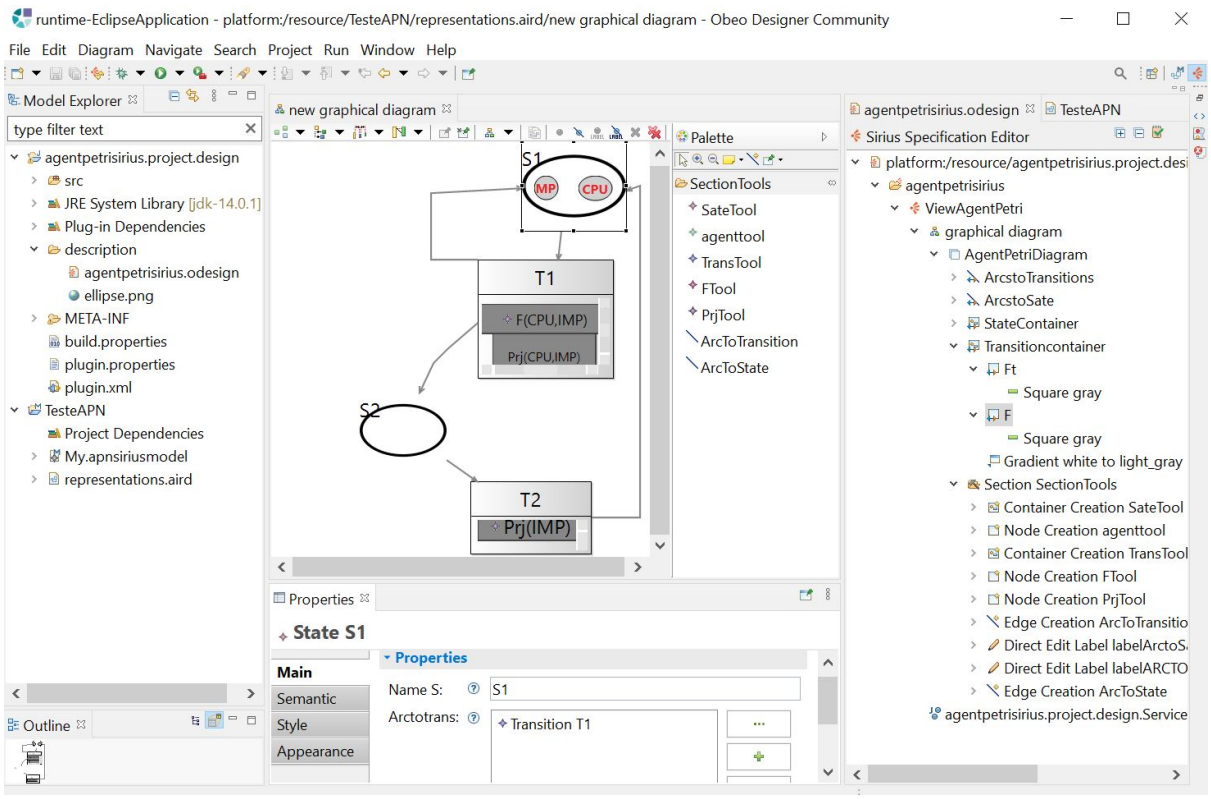

Figure 5.8: The Sirius final user product (APNSiriusModel) [47]

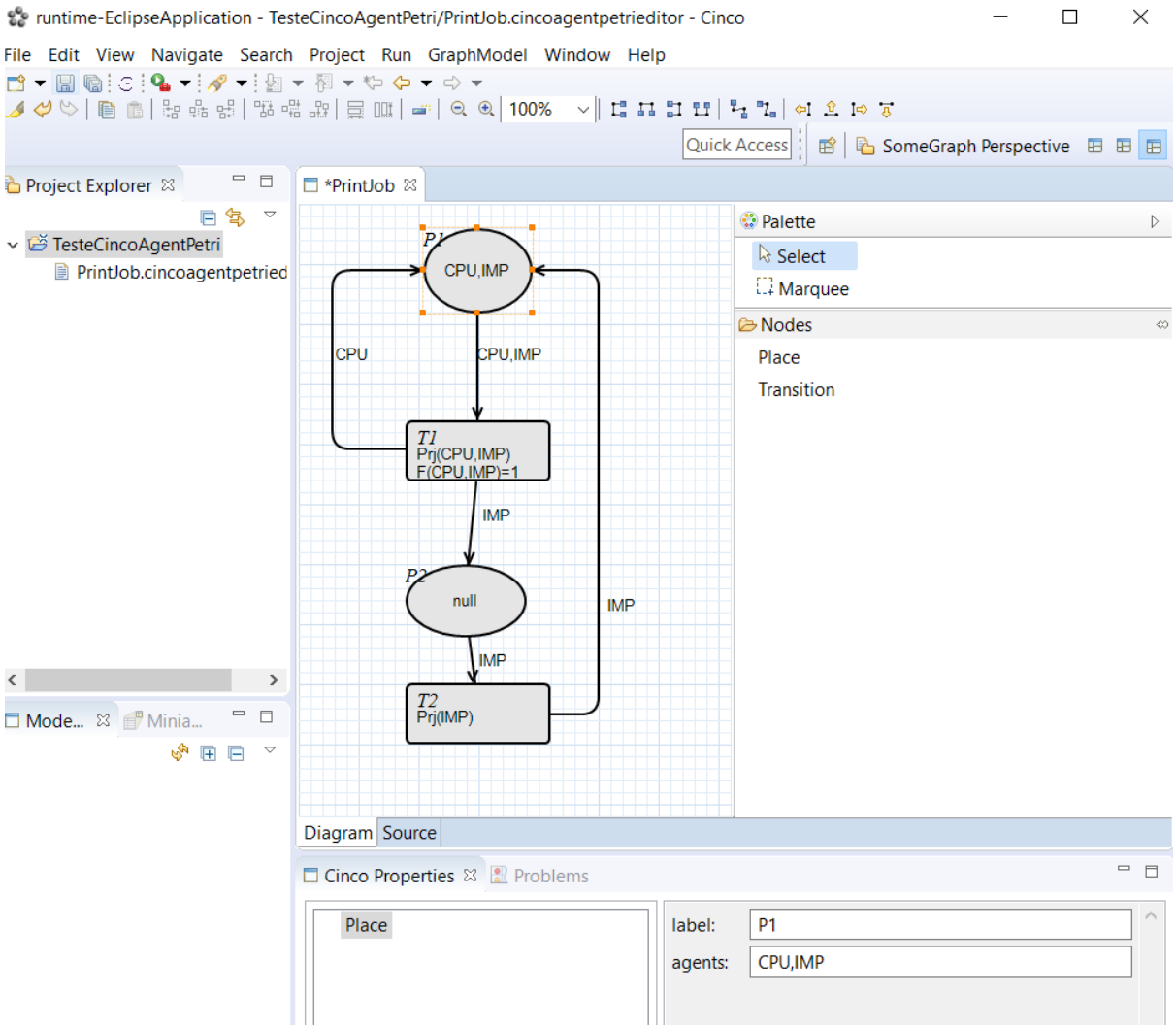

Figure 5.9: APN Cinco product(CincoAgentPetriEditor). [47]

Table 5.1: Evaluation outcomes

	GMF	Sirius	Cinco
Modeling challenge	Usually a mapping conflict happens	Required manually adjust mapping model	Requires just a correct assignment
Abstraction level	- Tree based model specification - Many details should be specified - Six meta-models	- Tree based model specification - Many details should be specified - Two meta-models	- Textual specification support - Little details should be expressed - two meta-models
Graphical notation& node connexion	Simple graphical notation support Required efforts to specify a sophisticated notation Example: Curvature of connexion line is not supported.		Sophisticated graphical notation. Example: a good appearance of shapes, a bendability in the connexion line
Modeling Simplicity	Average (even designer mastering modeling process he needs effort to adapt disaccording)		Strong support (after mastering the textual meta language, a little effort is needed to get a sophisticated tool)
Code-generation	Partial support (structural model implementation)	Partial support (structural model implementation)	- More holistic approach - Sufficient API for sophisticated - experience with model elements - API support for model semantic implementation
live development feature	Unsupported	The specification and the modification of graphical models still always activate in runtime	Unsupported

We should noted here that we make our experimentation with the same use case in the three final editors (the print job example specified in the contribution of [6] has been used)

5.5 Conclusion

In this chapter, we have presented three MDE modeling solution: Cinco, Sirius and GMF. These solutions are compared based on some selection criteria(Abstraction, Simplicity, specialisation and productivity), using a same meta-model called APN (a Petri nets extension). The study helps designers to make a quick preview about each solution and makes focus on strong characteristic supported by each solution. Cinco seems more sophisticated regarding to the result achieved by this evaluation model elements other.

Chapter 6

Cinco Modeling And Simulation Approach For APN

Contribution:Cinco-Based Approach for Agent Petri Net Models [57]

6.1 Introduction

Designing a rigour model necessities the selecting of the adequate modeling language, Agent petri net(APN) is a Petri net kind recommended to design complex systems where agents are the proposed solution for mastering this complexity [6]. Getting an early experience with these modeled systems is a suitable requirement for designers, simulation is a mean that help them to experiment quickly their proposed models [60]. Having a supported solution where designers could specify APN models and execute them is a challenging issue where no software application is released for this extension [58]. Coding application from scratch will be a surcharge for programmer; abstraction still the fundamental pillars in the revolution happening in computer science. Embracing a solution that heavily incorporates abstraction in producing new software will have a huge positives impact in development process. MDE solutions contribute in this revolution by offering modeling languages and tools that promote the application of abstraction through models [61]. It provides mechanisms to make this abstraction a reality by supporting its translation to corresponding code that could be reused in other extended applications.

Merging rigorous methods with model driven solutions allows designers to benefice form the correctness of formal methods and the automatic support to implementation of MDE.

In this chapter, we present an APNSimulator approach to combine APN with Cinco. This research has been the subject of our published contribution [57]. In section 2, we present the tools that we embrace in the realisation of our proposed approach, both APN and Cinco are presented. In section 3, we illustrate our approach and its realisation. In section 4, a real life example are employed to experiment our design-simulator plugin and we conclude this work in the last section.

6.2 Backgrounds

6.2.1 Agents Petri Nets

We have already present the main purpose of this formalism and its applications in Chapter 2; the formal definition is presented here. An APN is described by:
$APN = (S, E, Med, Pst, Prj, Ft, f, Envj)$ [6] where

- S and E are two finite and non empty sets of places and transitions, $S \cap E = \emptyset$.
- Pst is the input function and Med is the output function,
  * Pst :S×E→ N ,
  * Med :E×S→ N
- Prj: is the firing-condition function.
- F: agent relation function (uses to describe the existence of a relation between agents)
- Ft: agent function (this function is used to represent exchange data between agents).

- Env_j: represents the space of agents engagement.

In the contribution of [6]; the formal definitions for each function is fully described and; in our contributions [58, 57] we resume the definition of main functions. Having a PN extension where tokens interpreted as agents is the main purpose behind the proposition of this formalism.

6.2.2 Cinco Modeling generation solution

We have already presented this tooling in chapter 5, where we compare this one with other MDE workbenches. Based on the practical experience, Cinco prevail them regarding to its embraced approach that fully supports an implementation mechanism and emphasizing simplicity and specialisation. We shortly resume these points here:

- Simplicity:
  - The generation of editor solely relied only one the definitions of two models.
  - Adding semantic using the support of the implemented API simplifies the product development process; developers reuse implemented functions that assist them to define model execution. Cinco provides enough functions to enable linking element of applications (model, view and controller); enable the activation of the execution semantic code by graphical components (this will be taken automatically by the tool once a specification for this linking is declared before the concerned nodes).
- Specialisation: the tool helps designers to tailor a satisfactory user application using MGL and MSL specification
- Code generation functionalities: The tool supports the fully generation of editor functionalities to manipulate and validate model instances; APIs that help developers to satisfy the final product(quality of graphical representation, the guarantee of structural validation, ability to export the model in different formats).

6.3 APN Cinco based approach

6.3.1 The approach

We present here an MDE approach to integrate APN in Cinco. In the main purpose to provide an MDE solution where users could build a rigour model and execute quickly their representation in a runtime environment. The APN as a petri extension, plays an important role to provide a correct model and Cinco as an MDE solution leads the transition from abstract specifications to the implementation of user applications.

The figure 6.1 represents our approach. We firstly start by building the graphical editor then we add the implementation of execution (simulation scenario) and we regenerate the Cinco application code again.

6.3.2 APN Simulator Models

The MGL specification is used to describe the structural representations of model components; we note that in the last release there are some additional declarations in this MGL specification. Listing 6.1 represents our proposed meta-model definition for our APN Simulator named *APNSimulatorModel*. Instructions from line 1 to 7 specify the name of the model, its extension and the main compartment where nodes and arrows will be created.
Five principal elements are declared: container,nodes and edges. The container AgentEnvironment is used to specify APN agent environment, (see Listing 6.1 instructions block from line 13 to line 18). The APNSate node is used to specify APN places (see Listing 6.1 instructions block from line 8 to line 12) and the APNTransition node is used to specify APN transition (see Listing 6.1 instructions block from line 20 to line 31). Two edges are declared to specify connexions between APNSate node and APNTransition node. These arrows are named respectively: S2TArrow and T2SArrow; the first one is used to design an arc coming from a APNSate and going to a APNTransition (see instruction from line 32 to line 34) and the second one is used to design an arc coming from APNTransition and going to a APNSate (see instruction from line 35 to line 37).

For each node and arrow, we need to specify some attributes and relations as labels, incomingedges and outgoingedges.

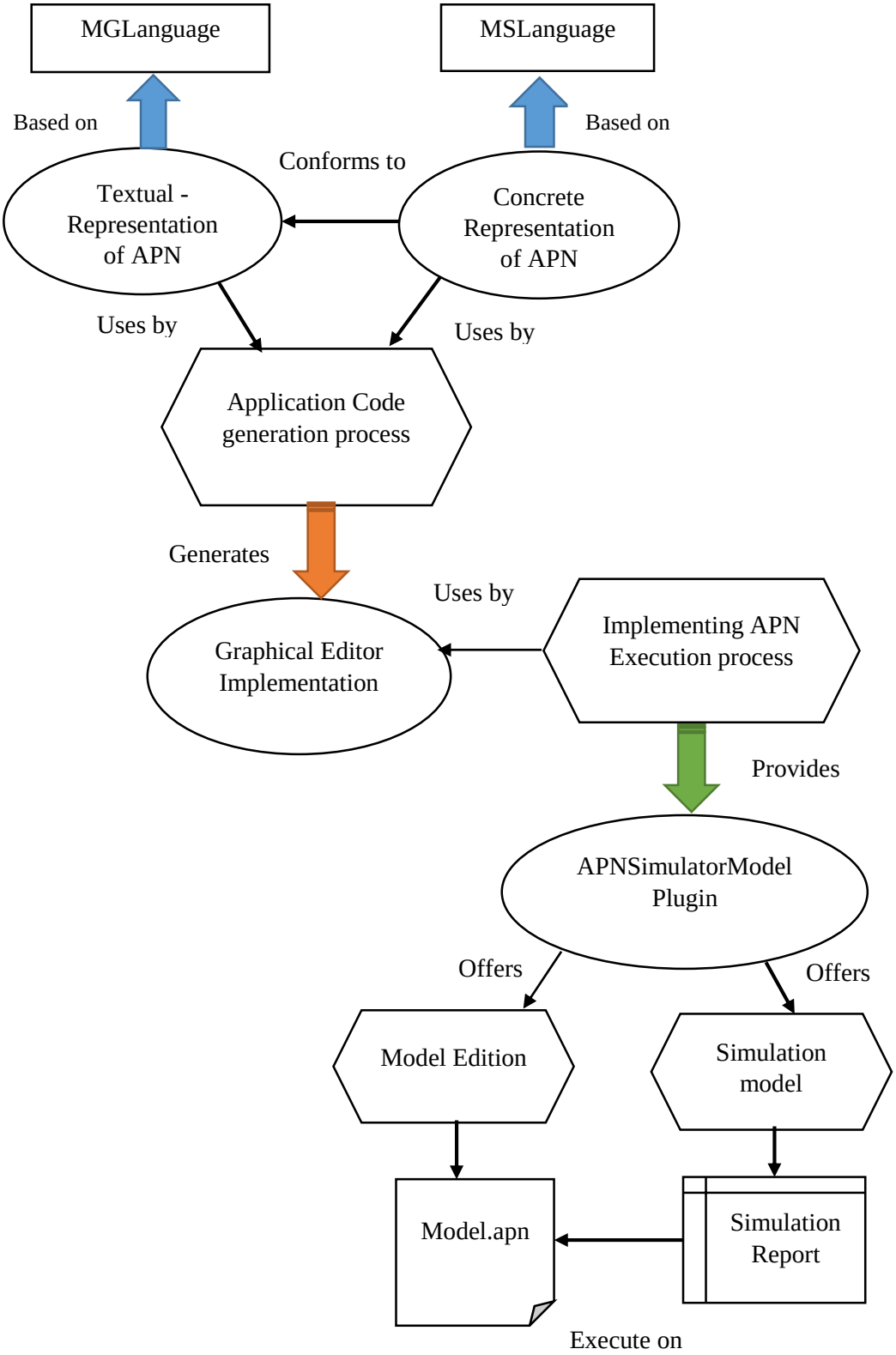

Figure 6.1: APN Cinco based approach

We propose the APNSimulatorModel.style to depict the graphical appearances of model elements (APNSimulatorModel.mgl) using MSL specification. See listing 6.2, the graphical definitions are proposed. The following graphical appearances are specified for model elements:

- The node style named GraphicalPlaceShape is specified to depict APNSate node (see 6.2, form line 17 to line 22).

- The node style named GraphicalTransitionShape is specified to depict APNTransition node (see Listing 6.2, line 23 to line 30).

- The edgeStyle named GraphicalShapeArrowsS2T and edgeStyle named GraphicalShapeArrowsT2S are used to specify respectively S2TArrow edge and T2SArrow edge (see Listing 6.2, form line 31 to line 34).

- The key words @style is used to link the corresponding graphical shape with its GraphModel element (see Listing 6.1 lines: 9, 14, 23 and 33).

Cinco product model is used by the Cinco generator engine to identify the based meta-model for the generated product; Listing 6.3 shows the Cinco implementation for this model.

```
  id info.scce.cinco.product.apnsimulatormodel.mglid
  stylePath "model/APNSimulatorModel.style"

  graphModel APNSimulatorModelGraphModel {
    diagramExtension "apn"
    containableElements(AgentEnvironment,APNState,APNTransition)
    attr EString as modelName }
  container AgentEnvironment
  {   style GraphicalenviShape("${enviromentsname}")
    containableElements(APNState,APNTransition)
    unique attr EString as enviromentsname
  }
  node APNState {
    style GraphicalPlaceShape("${placename}" ,"${agentslist}")
    unique attr EString as placename
    attr EString as agentslist
    incomingEdges(T2SArrow)
    outgoingEdges(S2TArrow)   }

  @contextMenuAction("events.ExecutionScenario")
  @doubleClickAction("events.ExecutionScenario")
  node APNTransition {
    style GraphicalTransitionShape("${Transitionname}" ,"${
prj_function}"
    ,"${ft_function}")
    unique attr EString as Transitionname
    attr EString as prj_function
    attr EString as ft_function
    attr EString as F_function
    incomingEdges(S2TArrow)
    outgoingEdges(T2SArrow)
  }
  edge S2TArrow {
    style GraphicalShapeArrowsS2T("${labelS2T}" )
    attr EString as labelS2T}
  edge T2SArrow {
    style GraphicalShapeArrowsT2S ("${labelT2S}" )
    attr EString as labelT2S}


```

Listing 6.1: Meta-model “APNSimulatorModel.mgl”

```
    appearance default {
      lineWidth 2
      background (229,229,229)
    }
    appearance Env extends default {
      background (255,255,255)
    }
    nodeStyle GraphicalenviShape(1){
      roundedRectangle {
        appearance  Env
        size(100,100)
        corner (8,8)
        text {appearance { font("Times New Roman", BOLD, 10)}
          position ( 10, 5 )
          value "%1$s"}
    }}
    nodeStyle GraphicalPlaceShape(2){
      ellipse {
        appearance default
        size(40,40)
        text { appearance {font("Times New Roman", BOLD, 10)} position
    ( 0, 0 ) value "%1$s"}
             text {position ( CENTER, MIDDLE ) value "%2$s"} } }
    nodeStyle GraphicalTransitionShape(3){
      roundedRectangle {
        appearance default
        size(100,60)
        corner (8,8)
        text {appearance { font("Times New Roman", BOLD, 10)} position
    ( 10, 5 )  value "%1$s"}
        text { position ( 10, 20 ) value "%2$s"}
        text { position ( 10, 35 ) value "%3$s" } } }
    edgeStyle GraphicalShapeArrowsS2T(1) {
      appearance default
      decorator {   location (1.0)  ARROW appearance default }
      decorator {   location (0.3)  movable text {value "%s"}}}

```

Listing 6.2: Meta-Style “APNSimulatorModel.style”

```
cincoProduct APNSimulatorModelTool
{
  mgl
  "model/APNSimulatorModel.mgl"
}

```

Listing 6.3: Meta-Style “APNSimulatorModelTool.cpd”

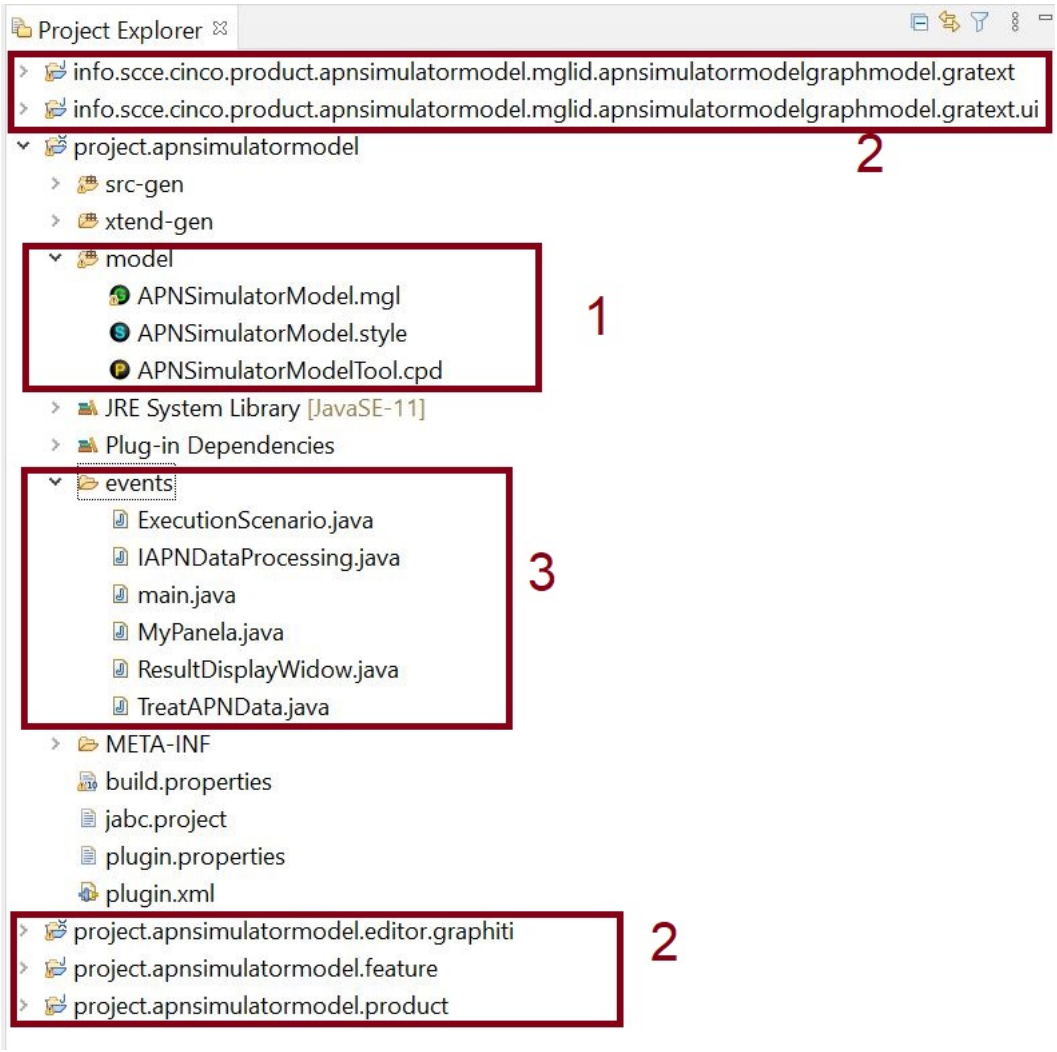

Figure 6.2: Plugin Packages

6.3.3 APN Code generation

To process a code generation of application, Cinco uses the model APNSimulatorModelTool.cpd; This declaration is used to build the plugging. This file is automatically generated once the project is created. The generation of Cinco product will be applied based on this file. The figure 6.2 shows the plugin implementation packages that contain:

1. Cinco models,

2. Generated implementation: Six plugins are generated: the basic application code, the gratex, the gratext.ui, the editor, the feature and the test.

3. The package event contains our implementation for the simulator .

6.3.4 APN Simulator implementation

As it mentioned in the previous chapter that the Cinco tooling supports three approaches to insert the model semantics: building process models, using specification languages method and coding using programming languages as java or Xtend. In the implementation of our simulator, we choose to define the semantic using java method. In the listing 6.4, we propose an abstract vision of the steps of executing an APN transition.

```
Step 1. Read transition information from the APN model (name, prj, f,
  incoming_places,  outgoing_places).
Step 2: Check the validation of prj_fuction(t).
Step 3: Check the validation of f_relation(t).
Step 4: Apply the validation result of  functions:
      if (prj_fuction(t)=true and f_relation(t)= true) then
      {
          Withdraw agents from the incoming_places.
           moving agents to the outgoing_places.
      }
      else
        write ("transition is not available").

```

Listing 6.4: Check firing APNtransition(t) Step

The figure shows 6.2 the package events that we have created to hold the implemented classes for our simulator: InterfaceTreatAPNData, ExecutionScenario, TreatAPNData are the main classes that implement the simulation process.

In the listing 6.5 The InterfaceTreatAPNData contains the declaration of necessary operations to treat the specified user model are declared, we describe in the following the main purpose of each method briefly:

- *extractlistAgenstConditions* method: the method extracts Agents list required in the Condition Prj or/and F from a String edited by users in model instance and stores the result in a vector.

- *extractlistAgentsStates* method extracts existent Agents list in predecessor nodes.

- *compareLACToGLP* method compares agents required in the transition con-

ditions with the global list of existent agents in predecessor nodes of this transition

- *SetChainPred* method returns the update value in a predecessor node
- *SetChainSuccessor* method returns the update value in successor node.

```
1  package events;
2  import java.util.Vector;
3  public interface InterfaceTreatAPNData {
4
5    Vector  extractlistAgenstConditions (String Conditions);
6
7    Vector  extractlistAgentsStates (String ChaineAgents);
8
9    boolean compareLACToGLP (Vector  CV, Vector AGl ) ;
10
11   String    SetChainPred (Vector listAC, Vector PredecessorlistIN) ;
12
13   String    SetChainSuccessor (String ArcAgent, String successorlistON)
       ;
14 }
```

Listing 6.5: Interface TreatAPNData

We implement *InterfaceTreatAPNData* in the class named *TreatAPNData*. *ExecutionScenario* class is the core component of our simulator(see listing6.6), we have extended it from the *CincoCustomAction* class, we have implemented two main methods canExecute and execute (see Listing 6.7 and execute (see Listing 6.10), these Listings are promoted by comment lines for more explanation.

Clicking on the concerned transition in model instance will activate the simulator implementation. canExecute method checks the validation of methods checkrF(see Listing 6.8) and checkPrj(see Listing 6.9). The assignment of this process with transition should be done on the meta-model; see listing 6.1 lines 20 and 21 a declaration of this assignment specified by the keyword @contextMenuAction followed by the name of the class that implements the simulator, in our case: *events.ExecutionScenario*

```
1  package events;
2  import java.util.Vector;
3  import org.eclipse.emf.common.util.EList;
4  import de.jabc.cinco.meta.runtime.action.CincoCustomAction;
5  import info.scce.cinco.product.apnsimulatormodel.mglid.
       apnsimulatormodel.APNState;
6  import info.scce.cinco.product.apnsimulatormodel.mglid.
       apnsimulatormodel.APNTransition;
7  import info.scce.cinco.product.apnsimulatormodel.mglid.
       apnsimulatormodel.S2TArrow;
8  import info.scce.cinco.product.apnsimulatormodel.mglid.
       apnsimulatormodel.T2SArrow;
9  // this class contains the simualtion code
10 public class ExecutionScenario extends
11 CincoCustomAction<APNTransition>{
12   public TreatAPNData TD ;
13   /* canExecute method Checks if the crossing of a transition is
       available  */
14   public boolean canExecute(APNTransition t){/* the code in the
       following listening....*/}
15   /* this method run if canExcute return true, here the simulation
       scenario will be executed*/
16   public void execute(APNTransition t) {/* the code in the following
       listening....*/}
17    public String getName() {return " ExecuteTransition";}
```

Listing 6.6: ExecutionScenario.java

```
public boolean canExecute(APNTransition t) {
  // method that checks if relation F is true;
  boolean checkC1 = checkrF(t);
  //method that checks if Prj is true;
  boolean checkC2 = checkPrj(t);
  if(checkC1 ==false)
  {
    String msgRef="- Agents in transition "+t.getTransitionname() +"
   refuse to establish a connexin  ";
    MyPanela pp3=f.getPanel();
    index=index+1;
    pp3.addpanlecomponent(msgRef, index);
    f.setPanel((pp3));
    f.setVisible(true);
    return false;
  }
  else
  if (checkC2==true)
  return true;
  else
  return false;
}
```

Listing 6.7: CanExecute.java

```
1    public boolean checkrF(APNTransition t) {
2      String msg="";
3      String q=t.getF_function().trim();
4      // the method f_syntaxe(q) checks the validation of the syntaxe
   of the edited relation(F) by user.
5      boolean f_synt=f_syntaxe(q);
6      if (f_synt=true )
7      {int po=q.indexOf("(");
8        int pf=q.indexOf(")");
9        String listagentsrelation=(String) q.subSequence(po+1, pf);
10       char response=q.charAt(pf+2);
11       if (response=='0') {
12         return false;
13       }else {
14         return true; }}
15     else
16     {
17       msg="msg error";
18       return false;}
19
20   }
21
```

Listing 6.8: checkF.java

```
public boolean checkPrj(APNTransition t) {
  String  Conditions=t.getPrj_function().trim();
  String  allAgentlist= "";
  /* TreatAPNData is an instance of an implimented classe
  where we treat the user's inputs in the diagram*/
  TD=new TreatAPNData();
  // the following code extract list agent from the condition list
  TD.listAC=TD.extractlistAgenstConditions(Conditions);
  for (APNState pre : t.getPredecessors()) {
    allAgentlist=allAgentlist+" "+pre.getAgentslist().trim(); }
  TD.listIN =TD.extractlistAgentsStates(allAgentlist);
  /* The following code compare agents in the predecessor nodes
  with required list in the condition of the transition*/
  int x=0;
  if ((TD.compareLACToGLP(TD.listAC, TD.listIN))==true )
  {System.out.println (" the transition is available ");}
  else
  {messagerapport="- User wants  to  process the transition "+t.
   getTransitionname()+" witch is no available ";
    index=index+1;
    MyPanela pp2=f.getPanel();
    pp2.addpanlecomponent(messagerapport, index);
    f.setPanel((pp2));
    f.setVisible(true);
  }
  // return the final result of checking the ability of transition
   execution
  return TD.compareLACToGLP(TD.listAC, TD.listIN);
}

```

Listing 6.9: checkPrj.java

```
public void execute(APNTransition t) {
  // the following msg will be add to generated in  simulation report
  messagerapport="- User wants to  process the transition  "+t.
   getTransitionname()+"  witch is available";
  index=index+1;
  /* the following code adds the initial information to the window
  where we display the result of simulation */
  MyPanela pp=f.getPanel();
  pp.addpanlecomponent(messagerapport, index);
  f.setPanel((pp));
  f.setVisible(true);
  messagerapport="  The Transition  "+t.getTransitionname() +" "+ "is
   running ";
  String agentsmouved="    as a result ";
  //the following code updates the Agents List in predecessor Places of
    the  transition t
  for (APNState pre : t.getPredecessors()) {
    TD.listIN=null;
    pre.setAgentslist(TD.SetChainPred(TD.listAC, TD.
   extractlistAgentsStates(pre.getAgentslist())));
  }
  ///the following code updates Agents List in  Successor states  of
   transition t
  for (APNState succ : t.getSuccessors()) {
    EList ListArcOutgoing=t.getOutgoingT2SArrows();
    EList<T2SArrow> ListArcIncoming=succ.getIncomingT2SArrows();
    for(int e1=0 ;e1<succ.getPredecessors().size();e1++)
    {  if (t.equals(succ.getPredecessors().get(e1)))
      {    String etiquette =ListArcIncoming.get(e1).getLabelT2S();
        agentsmouved=agentsmouved+"   "+etiquette;
        succ.setAgentslist(etiquette+" "+succ.getAgentslist());
  }}}
  // the the follwing code adds the final report of the execution of a
   transition t
  messagerapport=messagerapport+ ""+ agentsmouved + " are mouved ";
  pp=f.getPanel();
  index=index+1;
  pp.addpanlecomponent(messagerapport, index);
  f.setPanel((pp));
  f.setVisible(true);}
```

Listing 6.10: Execute.java

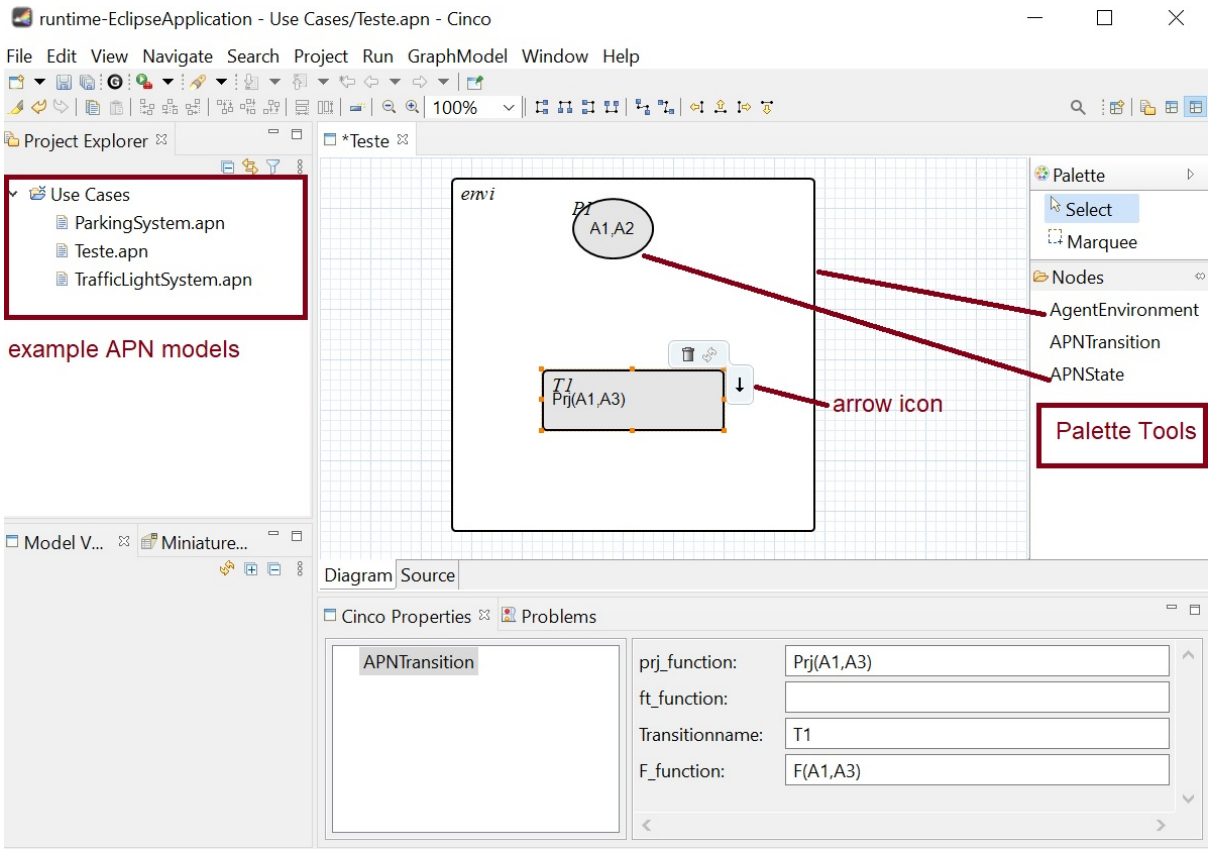

Figure 6.3: APNSimulatorModel plugin Main Window

6.3.5 APNSimulaor Plugin

After implementing these classes, we should process a regeneration of implementation of our Cinco product to ensure that all these classes will be taken in consideration in the runtime environment. By selecting the package project.apnsimulatormodel.editor.graphiti (see figure6.2 zone 2), we can run our Cinco Simulator product. Figure 6.3 shows the running result of our application. To create instances from model elements, user needs to select the corresponding tool(APNTransition or APNState) from the Palette, links between these nodes could be created from an icon attached to a node, to delete a current selection an option appears with it also.

6.4 Use Case

6.4.1 TL Simulation Model

Overview

To ensure a safety mobility of vehicles and pedestrians, the traffic light system (TLS) has been developed [62]. The standard mechanism aims to control the urban traffic, in reality this solution often be itself a source of traffic congestion where pedestrian and vehicles could be waiting for no reason to get a cross permission caused by fixed duration system. The adaptation of this mechanism has been the subject of many researches [62, 63].

The development of adaptive mechanism for TLS as event systems (the execution of this system is leaded based on environmental information) is an efficient solution to overcome the challenges of the standard version. We are interested in the research of [64], where a natural solution based on multi agent system has been proposed, traffic light timing board is updated based on environmental information (weather, arriving of pedestrian or vehicles, roadwork); collecting and exchanging information between agents will make the solution more natural and efficient. In our contribution, we have propose to design and simulate the announced problem using a rigour system, a Petri nets extension that combine the power of formal method and the characteristics of multi agent systems; an APN model for adaptive traffic light system is designed and experiment through a simulation functionalities in our developed tool [57].

TL proposed model

We propose the following agents, states, and transitions to design an adaptive traffic light system using APN:

Agents: Three agents are proposed to play roles of observation, controlling and managing the system.

- The first agent (observer) observes the environment using sensors, accumulates information and sends this result to the agent that is responsible to control the system.

- The second agent plays the role of controller; it receives environment in-

formation from the first agent and sends this information with the current state of the traffic light board to the third agent (manager).

- The third agent responsible to manage the crossroad authorisation between pedestrian and vehicles according to the current state and newly in the environment.

The three agents' names in the Traffic Light adaptive model system are Observer Agent (OTL), Controller Agent (CTL), and manager Agent (MTL).

States:

- P1: Observer Agent ready to execute sending msg1.

- P2: Observer Agent waiting to execute interaction with Controller Agent.

- P3: Controller Agent ready to interact with the Observer Agent.

- P4: Controller Agent preparing msg2.

- P5: MA ready to interact with the agent controller.

- P6: Controller Agent waiting to execute interaction with agent manager.

- P7: MA is preparing msg3.

- P8: Controller Agent updating TL board.

Transitions

- Tr1: Observer Agent sends msg1 to Controller Agent.

- Tr2: Controller Agent receives msg1.

- Tr3: Controller Agent sends msg2 to Manager Agent.

- Tr4: MA receives msg2.

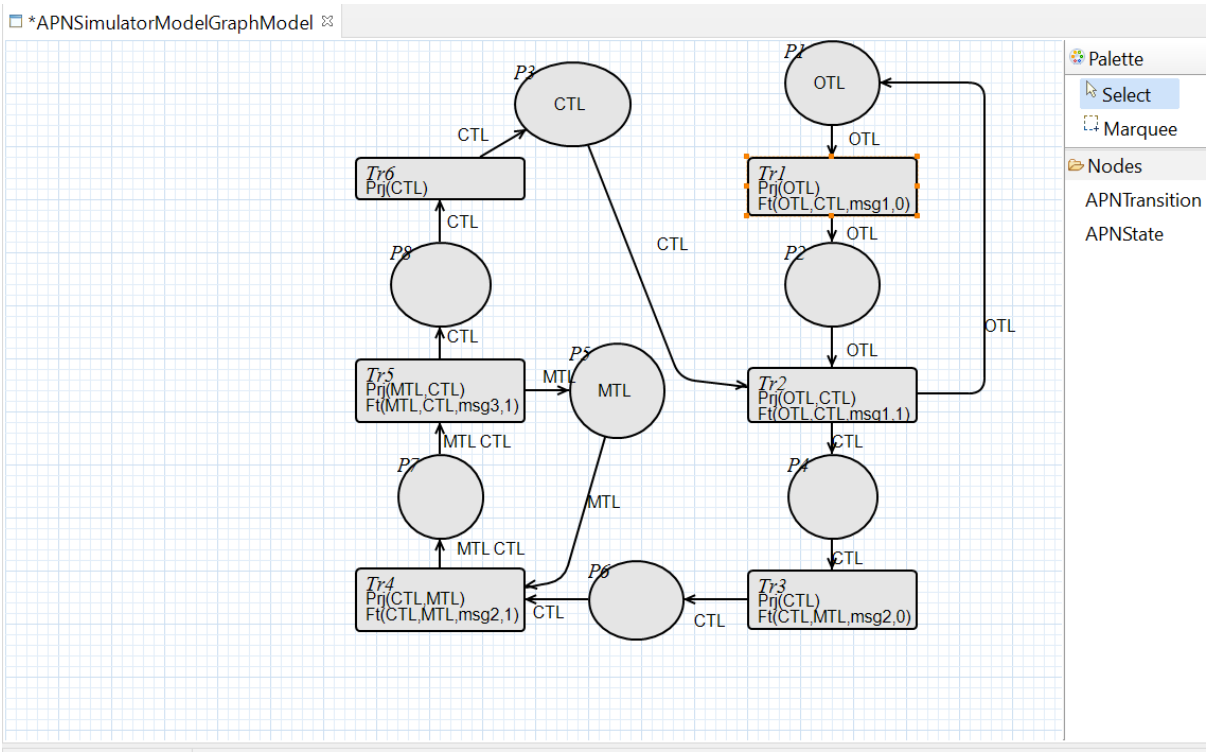

Figure 6.4: TrafficLightSystem APN Model

- Tr5: MA regulate values Of TL board.

- Tr6: Controller Agent updates TL timing.

The proposed model for this system is depicted in the figure6.4.

To execute a transition user needs to click on the rectangle that represents a transition (double click is required); the implemented simulation code will be executed. The corresponding simulation code checks the validation of two agent-conditions: Prj and F; the transition will be enabled if the both conditions are validated. Agents will be moved from incoming places to outgoing ones as a result of the execution of a firing of a selected transition.The validation of this model is shown in the figure 6.5.

We experiment different scenario in the following examples:

Example 1: in the logical sequence of events, the sequence *Tr1Tr2Tr3Tr4Tr5Tr6* will be simulated. To clearly envisage the simulation result of executing this sequence, the screenshots of this result are captured step by step and shown in figures: 6.6, 6.7, 6.8, 6.9, 6.10 and, 6.11.

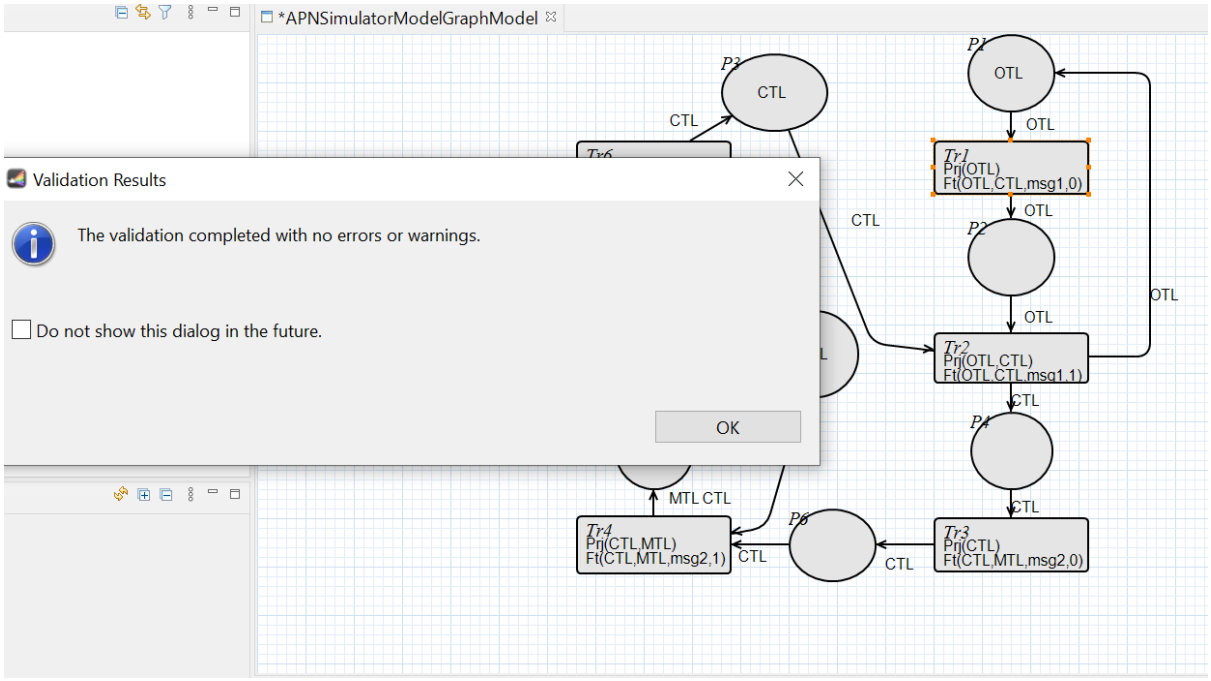

Figure 6.5: TrafficLightSystem APN Model Validation

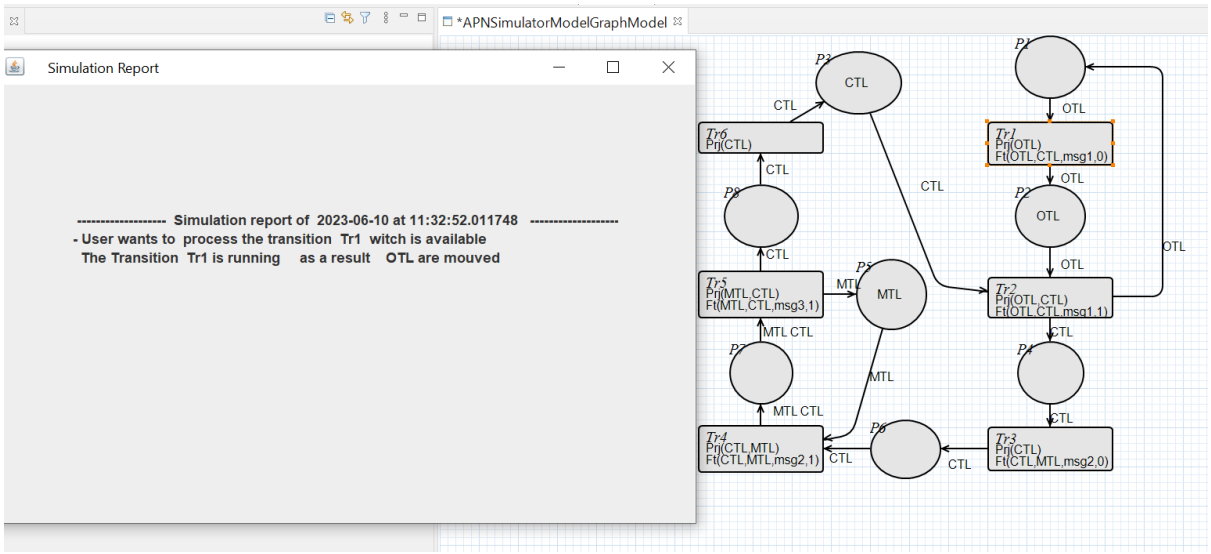

Figure 6.6: Execute Transition Tr1

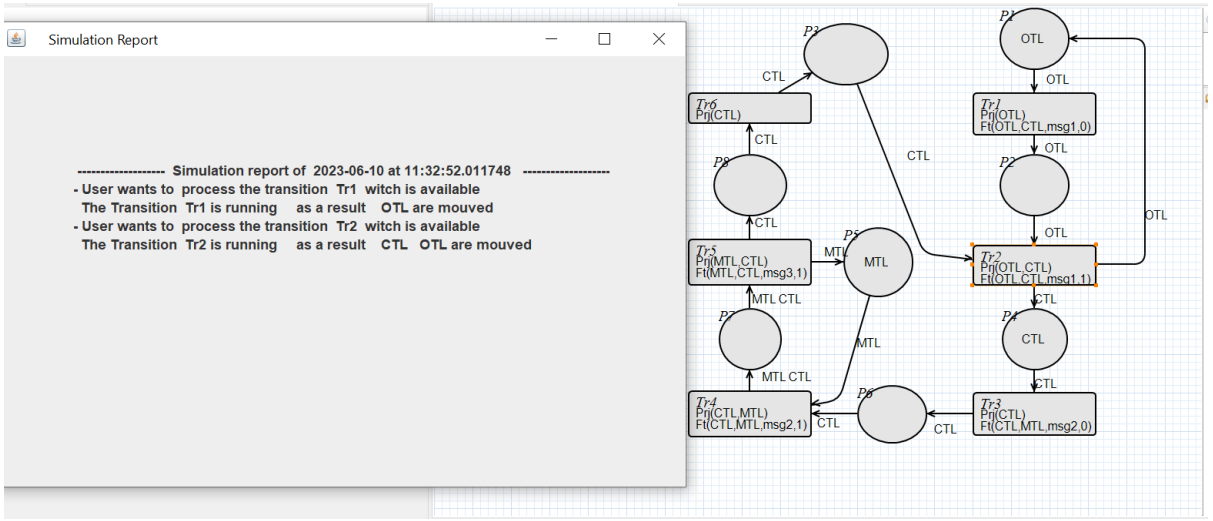

Figure 6.7: Execute Transition Tr2

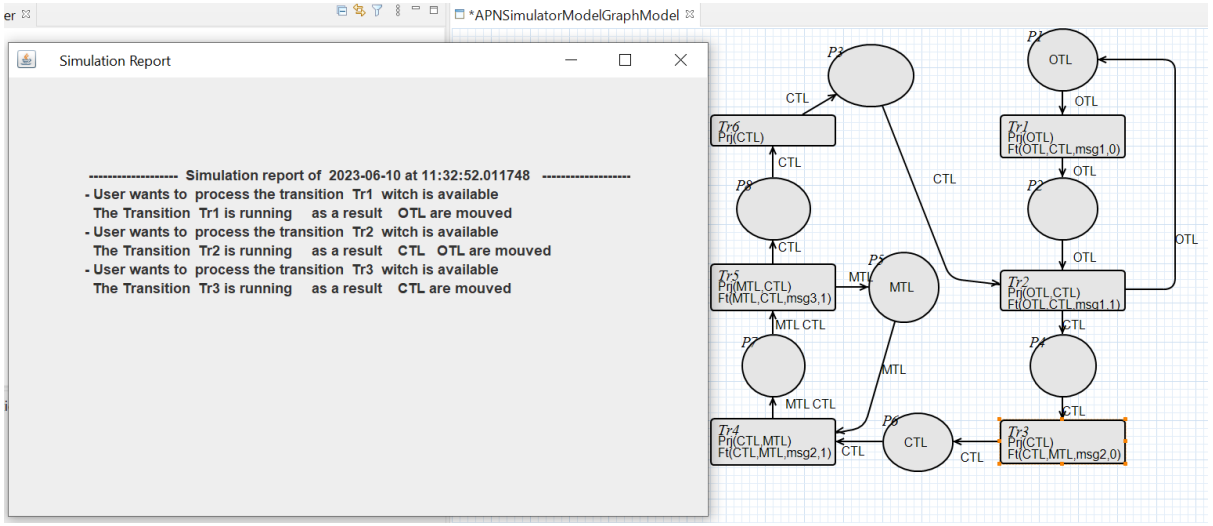

Figure 6.8: Execute Transition Tr3

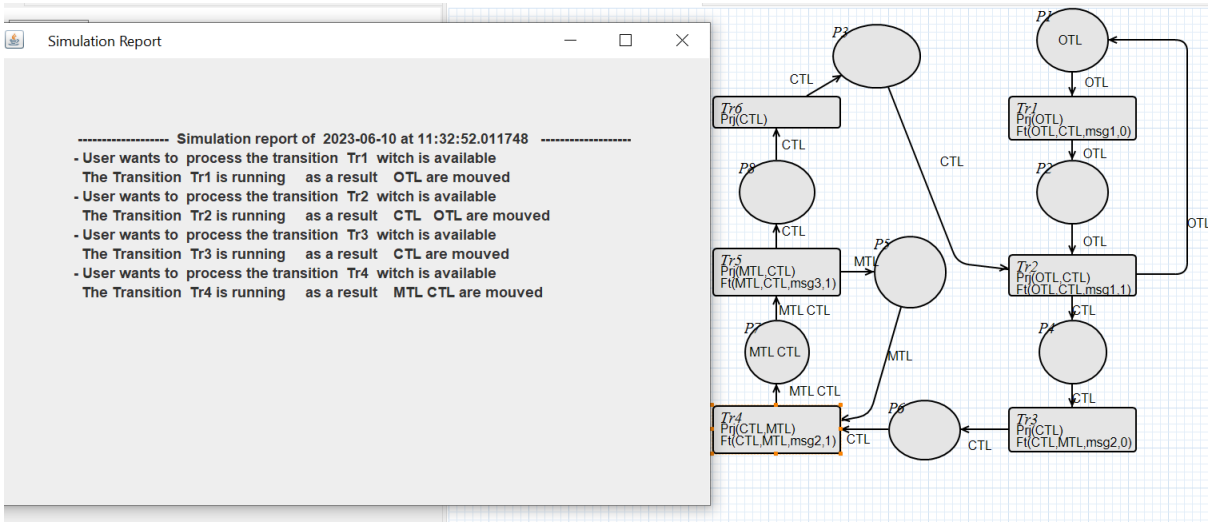

Figure 6.9: Execute Transition Tr4

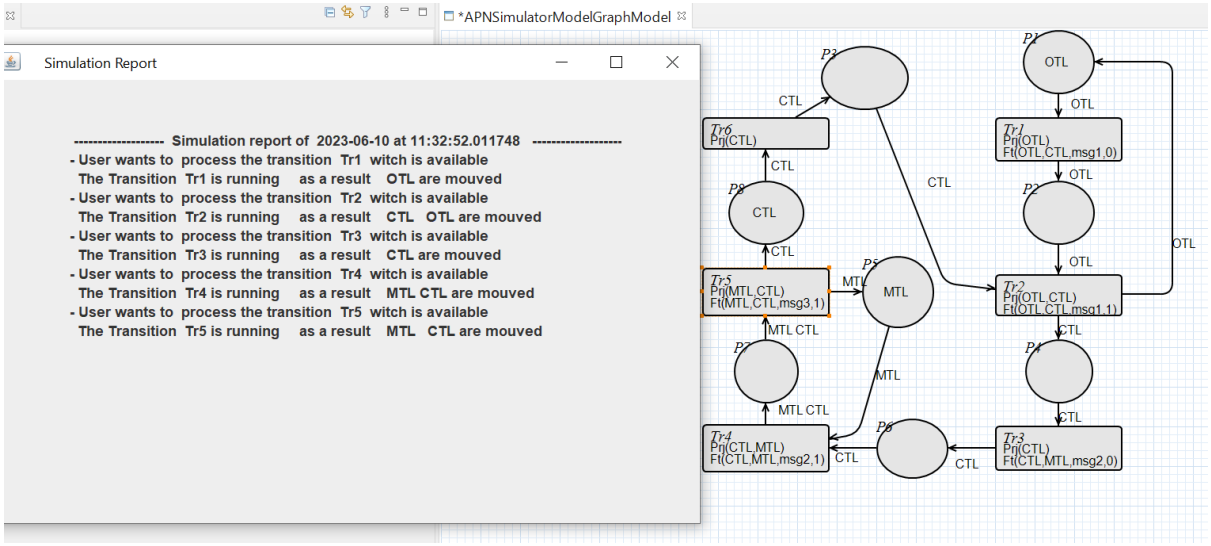

Figure 6.10: Execute Transition Tr5

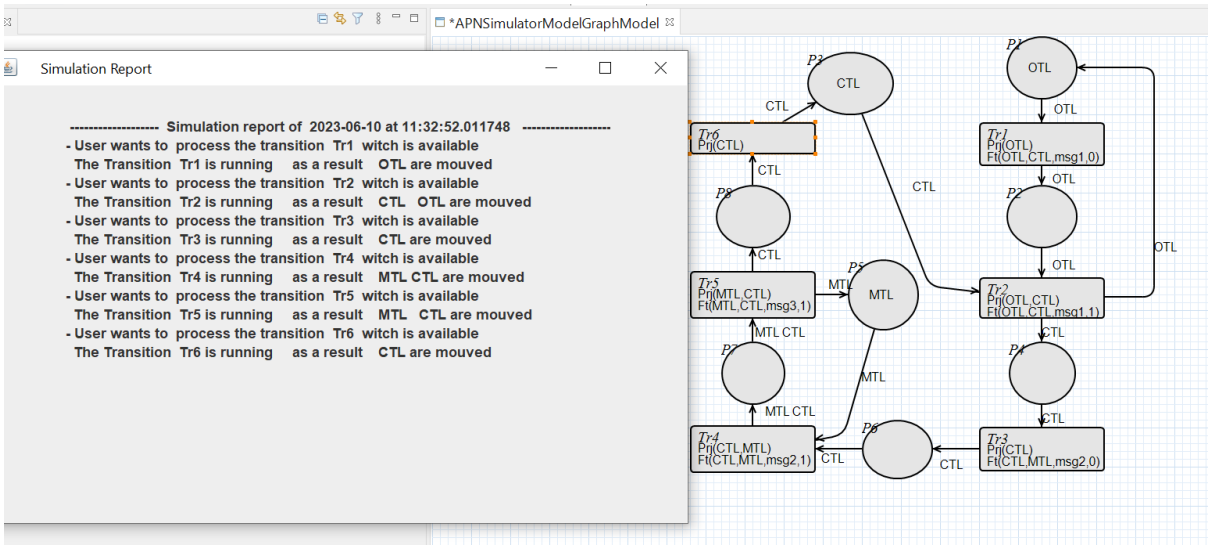

Figure 6.11: Execute Transition Tr6

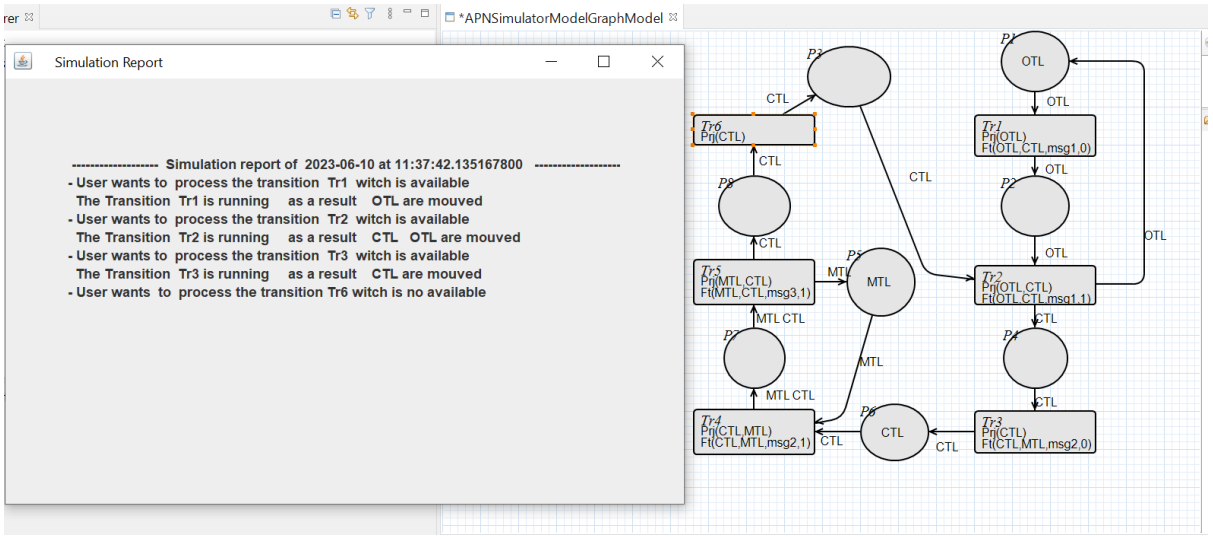

Figure 6.12: Execute Senario: SequenceTr1Tr2Tr3Tr6

Example 2: Figure 6.12 shows the simulation result of executing a transition sequence (*Tr1Tr2Tr3Tr6*) where some transitions are not available for this sequence.

Example 3: Figure 6.13 shows the execution result of the transition sequence *Tr1Tr2Tr3Tr4Tr5Tr6* where a possibility of declining a connection could happen.

6.5 Conclusion

In this chapter, we have presented our MDE approach to incorporate a PN extension that supports the multi-agents approach in Cinco tooling suite. Creating the graphical editor and implementing the simulation has been done using the facilities provided by Cinco. As a result, a software solution executed as a plugin eclipse; this solution assists user to edit and simulate models specified in an apn extension. An adaptive model for traffic light system is presented, modeled using APN extension and, simulated. Some execution scenarios are applied to have an early experience with this system. The simulator helps users to get an opportunity to experiment quickly their APN models. Nevertheless, the requirements to promote this tool by verification functionalities are required.

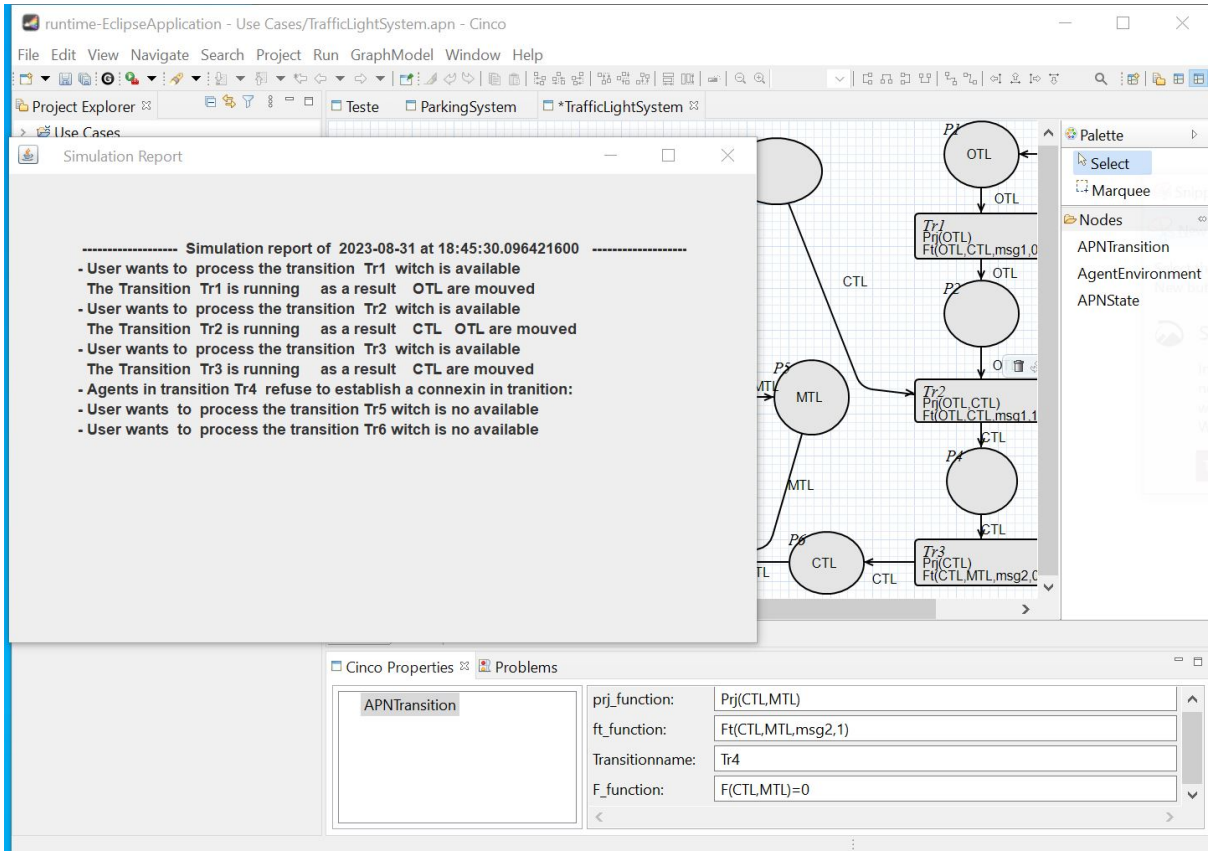

Figure 6.13: Execute Senario:Sequence Agent refuse connection

General Conclusion and Perspectives

Conclusion

The mapping between petri nets and semi formal model have been applied in many research, despite the advantage of this approach, ensure application over all consistence is another challenge. Merging formal method with model driven engineering tools in the same framework gives a hand to designer to leverage the positive aspect of the both approaches. The main aim of our contribution is to provide an integrated solution that combines formal methods with model driven engineering tools, this integration gives a hand to designers in the development of rigour modeling systems.

Exposing worthwhile reference about the state of art of a formal method and model driven have been the subject of the first part of this dissection. Provide an analysis study of available PN software, comparing brilliants MDE workbenches and proposing an integration of APN in MDE platforms has been the subject of our contributions.

The purpose of this dissertation is to communicate our result and analysis with Petri nets and model driven communities.

- In Chapter 2 and 3, we have elaborated a bibliographical synthesis that covers the essential concepts of Petri net modeling languages and model driven engineering. Chapter 2 deals with the presentation of the formal definitions, the modeling capabilities and the important analysis techniques. Chapter 3 is devoted to expose the main concepts of MDE.

- In chapter 4, we have offered a valuable report addressing the PN implemented software solutions, a study is elaborated where we envisage the capabilities of these solutions, we expose a list of the available modeling solutions for numerous PN kinds accompanied with their distribution licences and their supported operating environments. The available functionalities in these solutions are also highlighted with a focus on the most important ones (the ones that are most required by developers); designer could easily identify the most suitable solution for his developed application. Even with the presence of over than 90 PN modeling products, the support of new extension still a challenging issue for PN designer, develop a modeling solution by adopting MDE approaches seems more fitting choice for this challenging.

- In Chapter 5, we have offered a synthesis study where the evaluation of Oboe-designer, GMF and Cinco has been the subject of this research. The analysis aims to expose the level of abstraction of every solution, evaluate the specialization and simplicity of these three solutions in the production of DSL modeling solution, and highlight the capabilities of these solutions to make an automatic transition from this abstraction to the corresponding implementation. Regarding to the result of this comparison effort, the Cinco tooling prevail the others in this challenge due to its adopted approach based on simplicity and specialization that assists engineering designers and also non computer science designers to exploit the benefit of this technology in the creation of their own applications.

- In chapter 6, we have presented our MDE approach to integrate Agent Petri Nets with Cinco tooling suite. APN ascertains the accuracy of the modeled systems and MDE plays a transited bridge from abstracting representation to their corresponding implementation. A simplified APN Simulator approach is presented and implemented in a MDE platform. The simulator assists users in the creation of their APN models and allows them to experiment their APN models through simulation functionality; an assistance report is also provided during the execution of the simulation. This APN modeling solution is realised as an eclipse plugin and could be easily extended to support other user requirements. A Traffic light systems as an example of industrial system is used to experiment this tool.

Perspectives

We highlight in the following points the future perspectives that appear on the horizon of this contribution.

The evaluation of workbenches In this thesis, the comparison is based on some important criteria as abstraction level and development simplicity; however, it will be also interested to extend the scope of this comparison. The evaluation of the potential of these workbenches to define semantics and to implement verification process is very important; examples:

- The support of transformation between models.

- The provider API facilities to integrate model-checking techniques.

The integration of PN in Cinco: Despite that our plugin helps designers to specify their APN, guarantee the structural validation of APN models and enables designers to process an early experience with their models through simulation, the semantic verification still a challenging issue. In future works, we aim to promote our plugin with helpful functionalities as model checking, reduction nets, simple and advanced analysis techniques.

Appendix

List of publications

Dembri, Amel, and Mohammed Redjimi. “Towards a Model Driven Approach for Integrating NWN Models in CINCO.” 2022 2nd International Conference on New Technologies of Information and Communication (NTIC). IEEE, December 21, 2022. https://doi.org/10.1109/ntic55069.2022.10100549.

Dembri, Amel, and Mohammed Redjimi. “Towards a Simplified Evaluation of Graphical DSL Workbenches.” 2022 5th International Symposium on Informatics and Its Applications (ISIA). IEEE, November 29, 2022. https://doi.org/10.1109/isia55826.2022.9993580.

Dembri, Amel, and Mohammed Redjimi. "Cinco-Based Approach for Agent Petri Net Models." International Journal of Organizational and Collective Intelligence (IJOCI) 12, no. 2 (2022): 1-17.

Dembri, Amel, and Mohammed Redjimi. "Towards a meta-modeling and verification approach of multi-agent systems based on the agent petri net formalism." In International Journal of Information Technology and Computer Science, vol. 11, no. 6, pp. 50-62. MECS Publisher, 2019

Bibliography

[1] What Is ISE? — ise.washington.edu. https://ise.washington.edu/about/what-is-ise. [Accessed 09-Jan-2023].

[2] Carl Adam Petri. Kommunikation mit automaten, 1962.

[3] Industrial Use. https://cs.au.dk/cpnets/industrial-use.

[4] Kurt Jensen and Lars M Kristensen. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.

[5] Rüdiger Valk. Object petri nets: Using the nets-within-nets paradigm. *Lectures on Concurrency and Petri Nets: Advances in Petri Nets 4*, pages 819–848, 2004.

[6] Borhen Marzougui, Khaled Hassine, and Kamel Barkaoui. A new formalism for modeling a multi agent systems: Agent petri nets. *Journal of Software Engineering and Applications*, 3(12):1118–1124, 2010.

[7] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. Model-driven software engineering in practice. *Synthesis lectures on software engineering*, 3(1):1–207, 2017.

[8] Maurice H Ter Beek and Dejan Ničković. *Formal Methods for Industrial Critical Systems: 25th International Conference, FMICS 2020, Vienna, Austria, September 2–3, 2020, Proceedings*, volume 12327. Springer Nature, 2020.

[9] Rene David and Hassane Alla. Petri nets for modeling of dynamic systems: A survey. *Automatica*, 30(2):175–202, 1994.

[10] Manuel Silva. 50 years after the phd thesis of carl adam petri: A perspective. *IFAC Proceedings Volumes*, 45(29):13–20, 2012.

[11] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[12] James L Peterson. Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252, 1977.

[13] Md. Saidur Rahman. *Basic Graph Theory*. Springer International Publishing, 2017.

[14] WM Zuberek. Petri net models of process synchronization mechanisms, 1999.

[15] Xudong He. A comprehensive survey of petri net modeling in software engineering. *International Journal of Software Engineering and Knowledge Engineering*, 23(05):589–625, 2013.

[16] Michel Diaz. *Petri nets: fundamental models, verification and applications*. John Wiley & Sons, 2013.

[17] Borhen Marzougui and Kamel Barkaoui. Interaction protocols in multiagent systems based on agent petri nets model. *International Journal of Advanced Computer Science and Applications*, 4(7), 2013.

[18] Wolfgang Reisig. *Petri nets: an introduction*, volume 4. Springer Science & Business Media, 2012.

[19] Rachid Bouyekhf and A El Moudni. On the analysis of some structural properties of petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(6):784–794, 2005.

[20] Chérif Amer-Yahia, Noureddine Zerhouni, Abdellah El Moudni, and Michel Ferney. Some subclasses of petri nets and the analysis of their

structural properties: a new approach. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 29(2):164–172, 1999.

[21] Claude Girault and Rüdiger Valk. *Petri nets for systems engineering: a guide to modeling, verification, and applications*. Springer Science & Business Media, 2013.

[22] rph Sifakis. Structural properties of petri nets. In *Mathematical Foundations of Computer Science 1978: Proceedings, 7th Symposium Zakopane, Poland, September 4–8, 1978 7*, pages 474–483. Springer, 1978.

[23] Monika Trompedeller. Petri Nets Classification — www2.informatik.uni-hamburg.de. https://www2.informatik.uni-hamburg.de/TGI/PetriNets/classification/, 1995. [Accessed 01-Dec-2022].

[24] Luca Bernardinello and Fiorella De Cindio. A survey of basic net models and modular net classes. In *Advances in Petri Nets*, pages 304–351. Springer, 1992.

[25] Fei Liu, Monika Heiner, and Ming Yang. An efficient method for unfolding colored petri nets. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–12. IEEE, 2012.

[26] Gul A Agha. *Concurrent object-oriented programming and Petri Nets: advances in Petri Nets*. Number 2001. Springer Science & Business Media, 2001.

[27] Toshiyuki Miyamoto and Sadatoshi Kumagai. Application of object-oriented petri nets to industrial electronics. In *IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 64–69. IEEE, 2007.

[28] Li-Chih Wang. Object-oriented petri nets for modelling and analysis of automated manufacturing systems. *Computer Integrated Manufacturing Systems*, 9(2):111–125, 1996.

[29] Zhenhua Yu, Fang Guo, Jie Ouyang, and Lijun Zhou. Object-oriented petri nets and -calculus-based modeling and analysis of reconfig-

urable manufacturing systems. *Advances in Mechanical Engineering*, 8(11):168781401667769, November 2016.

[30] Borhan Marzougui, Kamel Barkaoui, Mohamed Amine Makni, et al. Agent petri nets framework for modeling staphylococcus epidermidis biofilm formation. *E-Health Telecommunication Systems and Networks*, 5(01):19, 2016.

[31] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[32] Jeff Kramer. Is abstraction the key to computing? *Communications of the ACM*, 50(4):36–42, 2007.

[33] Stefan Naujokat, Michael Lybecait, Dawid Kopetzki, and Bernhard Steffen. Cinco: a simplicity-driven approach to full generation of domain-specific graphical modeling tools. *International Journal on Software Tools for Technology Transfer*, 20:327–354, 2018.

[34] Miroslaw Staron. Adopting model driven software development in industry–a case study at two companies. In *Model Driven Engineering Languages and Systems: 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006. Proceedings 9*, pages 57–72. Springer, 2006.

[35] Edwin Seidewitz. What models mean. *IEEE software*, 20(5):26–32, 2003.

[36] Bran Selic. The pragmatics of model-driven development. *IEEE software*, 20(5):19–25, 2003.

[37] Thomas Kühne. Matters of (meta-) modeling. *Softw. Syst. Model.*, 5(4):369–385, 2006.

[38] Stephen J Mellor, Tony Clark, and Takao Futagami. Model-driven development: guest editors' introduction. ieee software, 20 (5). pp. 14-18. issn 0740-7459. *IEEE software*, 20(5):14–18, 2003.

[39] Richard Soley et al. Model driven architecture. *OMG white paper*,

308(308):5, 2000.

[40] Model Driven Architecture (MDA) | Object Management Group — omg.org. https://www.omg.org/mda/. [Accessed 05-Jan-2023].

[41] Douglas C Schmidt et al. Model-driven engineering. *Computer-IEEE Computer Society-*, 39(2):25, 2006.

[42] MBSE Initiative — incose.org. https://www.incose.org/incose-member-resources/working-groups/transformational/mbse-initiative. [Accessed 04-Jan-2023].

[43] A Wayne Wymore. *Model-based systems engineering*, volume 3. CRC press, 1st edition, 1993.

[44] David Ameller. Considering non-functional requirements in model-driven engineering. Master's thesis, Universitat Politècnica de Catalunya, 2009.

[45] Freddy Allilaire. The atlas transformation language (atlas transformation language atl) project.

[46] Petri Nets World: Petri Nets Tools Database Quick Overview — informatik.uni-hamburg.de. https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html. [Accessed 06-Jan-2023].

[47] Amel Dembri and Mohammed Redjimi. Towards a simplified evaluation of graphical dsl workbenches. In *2022 5th International Symposium on Informatics and its Applications (ISIA)*, pages 1–6. IEEE, 2022.

[48] Stefan Naujokat, Johannes Neubauer, Tiziana Margaria, and Bernhard Steffen. Meta-level reuse for mastering domain specialization. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II 7*, pages 218–237. Springer, 2016.

[49] Colin Atkinson and Thomas Kuhne. Model-driven development: a metamodeling foundation. *IEEE software*, 20(5):36–41, 2003.

[50] Vladimir Viyović, Mirjam Maksimović, and Branko Perisić. Sirius: A rapid development of dsm graphical editor. In *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pages 233–238. IEEE, 2014.

[51] Sirius | Home — eclipse.org. https://www.eclipse.org/sirius/. [Accessed 07-April-2023].

[52] About — cinco.scce.info. https://cinco.scce.info/about/. [Accessed 08-April-2023].

[53] Graphical Modeling Framework | The Eclipse Foundation — eclipse.org. https://www.eclipse.org/modeling/gmp/. [Accessed 07-April-2023].

[54] Eclipse Graphical Editing Framework™ (GEF) — projects.eclipse.org. https://projects.eclipse.org/projects/tools.gef. [Accessed 10-April-2023].

[55] Obeo products presentation - Obeo — obeosoft.com. https://www.obeosoft.com/en/products. [Accessed 10-April-2023].

[56] Michael Wenz. Graphiti Home | The Eclipse Foundation — eclipse.org. https://www.eclipse.org/graphiti/. [Accessed 11-Jan-2023].

[57] Amel Dembri and Mohammed Redjimi. Cinco-based approach for agent petri net models. *International Journal of Organizational and Collective Intelligence (IJOCI)*, 12(2):1–17, 2022.

[58] Amel Dembri and Mohammed Redjimi. Towards a meta-modeling and verification approach of multi-agent systems based on the agent petri net formalism. In *International Journal of Information Technology and Computer Science*, volume 11, pages 50–62. MECS Publisher, 2019.

[59] Nils Wortmann, Malte Michel, and Stefan Naujokat. A fully model-based approach to software development for industrial centrifuges. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II 7*,

pages 774–783. Springer, 2016.

[60] Sophie Prat. *Intégration de techniques de vérification par simulation dans un processus de conception automatisée de contrôle commande*. PhD thesis, Université de Bretagne Sud, 2017.

[61] Igor Rožanc and Marjan Mernik. The screening phase in systematic reviews: Can we speed up the process? *Advances in Computers*, 123:115–191, 2021.

[62] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence*, 26(5-6):1575–1587, 2013.

[63] Junchen Jin and Xiaoliang Ma. Hierarchical multi-agent control of traffic lights based on collective learning. *Engineering applications of artificial intelligence*, 68:236–248, 2018.

[64] Assia Belbachir, Amal El Fallah-Seghrouchni, Arthur Casals, and Marcia Pasin. Smart mobility using multi-agent system. *Procedia Computer Science*, 151:447–454, 2019.