Petri Nets Modeling & Analysis for Corporate Energy Invoice and Automation Management Software

Jamshaid Iqbal Janjua
Al-Khawarizmi Institute of Computer Science
University of Engineering & Technology,
Lahore, Pakistan
jamshaid.janjua@kics.edu.pk

Dr. Zubair A. Khan
Dean, Department of Electrical Engineering
University of Engineering & Technology
Lahore, Pakistan
deanee@uet.edu.pk

Dr. Farooq Ahmed
Dean, Faculty of Information Technology
University of Central Punjab
Lahore, Pakistan
dr.farooq@ucp.edu.pk

***Abstract* — Reliability and Conformance with the requirement specifications are the basic points of interest for the verification and validation of developed software in software engineering. Formal methods are a well researched area in verification and validation of software production. We discuss the mathematical approach by using Petri Nets (PN), to model the dynamic behavior of high level complex software, developed for corporate energy management in the open source development environment. Further, we discussed analysis of PN features like reachibility, deadlock, liveness and reversibility upon the modeled PN of developed software to prove the reliability & conformance of the developed software with software requirements.**

Keywords- Software Verification, Software Validation, Formal Methods, Petri Nets, Dynamic Behaviour Modeling, Open Source Environment, Energy Sector, PPA.

SOFTWARE systems have become the part of every circle of human life. Dependable systems are crucial for efficient working in the modern world. The software running the logical sequences of the algorithms to perform certain tasks, are the underline heart beat of these systems [1]. It deals with our financial management [2], regulates power generation [3], pedals various systems and process security-critical information. Invoices are the integral part of financial software which is used where services or products are provided [4][5]. The complexity of the invoices is pertinent in accordance with the problem domain. The flexibility and complexity of the software systems makes the requirements conformance a challenging task for the developers. This requires a solid modeling and analysis for its operational responsibilities. As a result, the skillful development of reliable software is of every growing importance that is in accordance with the requirements and specifications.

I. INTRODUCTION

GOVERNMENT of Pakistan announced a latest policy for the future electric power production projects in 2002. This policy is endorsed by National Electric Power Regulatory Authority (NEPRA). To make this policy effective, a new standard Power Purchase Agreement (PPA) has been developed by National Transmission & Dispatch Company (NTDC). This resulted in definition of a demand for an Efficient Data Collection, and Invoice Processing Automation System which implements the rules and regulations of the PPA in its best form. The new PPA of 2002 revolves around the basic concept of financial transactions based on the Complex Availability dependant billing. The main interacting bodies are Independent Power Producers (IPPs) and NTDC as Power Purchaser (PP) which operates through NTDC sub companies, Central Power Purchasing Agency (CPPA) & National Power Control Center (NPCC) also called as System Operator (SO). CPPA deals with the financial transactions regarding the sale and purchase of power between power generation and power distribution companies. The NPCC is responsible for secure, safe and reliable operations on the national power grid through proper control and dispatch instruction to IPPs and other power production facilities.

In order to implement demand identified for an automated enterprise grade data collection and invoicing software, we developed a software system named “Invoice Processing & Automation System (IPAS)”[6]. This system assists WAPDA and IPPs to operate in a time efficient manner as compared to the existing manual or semi automated

invoicing processes for IPPs. The system supports the transparency that refers to the vision of a glass tube though which the stakeholders can see the whole status of power production to the distribution facade. Another important feature of system automation is to focus on the paperless paradigm for sale and purchase of power.

Invoice Processing & Automation System (IPAS)

This system can be classified as corporate energy management system. It makes available a platform to energy regulatory bodies to have cleanest and crystal clear conduct of agreed terms and conditions with IPPs in PPA and proficient management of WAPDA's own energy resources. IPP's in accordance with PPA, are required to declare their available production capacity in advance, 12 months, 3 months, 14 days and 24 hours for medium & short term complex availability commitment. Followed by these obligations the IPPs are incessantly providing the changes in available capacity due to much unexpected environmental aspects and unforeseen conditions until the energy is generated and delivered in the particular operating hour. The NPCC watch this available capacity declaration from IPPs, while giving the Dispatch Instruction to an IPP for competent load balancing on the National Power Grid. Keeping in view this both side obligations, the IPP delivers the net electrical output energy at ambient conditions (temperature, pressure, humidity, etc.) under particular complex full or limited loading conditions with dissimilar fuel based energy production units. The complex maintenance also have an impact on the available capacity declaration of IPP, which are covered in certain complex outages allowances for scheduled and non scheduled outages for an agreement year.

These factors have an effect on the calculations of the monetary instruments for IPPs, primarily such as Capacity Payments for administrative and operational costs, Energy Payments for fuel utilization for energy production and Liquidated Damages intended for the result of IPPs not satisfying the both side complex availability commitment or not operating up to the required performance levels as per the PPA.

We have developed this system in full with all these preliminary outputs and supporting information. The outputs of the developed software are persevered in the central repository classified as technical and financial data disjointedly. The developed software system is engineered and finished using the open source technologies & tools environment (JSF, J2EE, MySql, etc.). We have processed several technical and financial information from different fuel based IPPs for more than twelve months and efficiently produced the results as per PPA indicating the issues and point of differences of the manual system well thought-out as human error or manual handing difficulty of the PPA due to its half hourly based calculations [6].

Petri Nets (PN) are mathematical as well as graphical tool for dynamic modeling and performance evaluation of information processing systems, which are characterized as being concurrent, asynchronous, distributed, parallel or stochastic [7]. Its powerful features not only assists in understanding of the dynamic behavior of the system but can also determine the different performance constraints that helps in providing orientation for system architecture and the options of parameters for the modeled system [15]. As a graphical tool, PN can be used as a visual communication aid similar to flow charts, block diagram, networks etc [8]. As a mathematical tool, linear programming, linear algebraic and graph theoretic techniques are applicable for verification of the modeled system. PN can be used for both theoreticians and practitioners. The paper proposes the PN oriented method for the evaluation of software dependability and conformance through the analysis of the PN model of a developed system. This method can assist in order to verify and validate the developed system effectively as per the requirements, rules and regulations of PPA in this case, which further reduces the complication of describing and analyzing the software reliability.

The paper is organized as follows. In Section II, we briefly discuss the basic concepts and definitions of the Petri Nets used in modeling of the developed corporate energy invoice and automation management software. In Section III, we discuss the proposed Petri Nets model for the said software. Our approach for application of Petri Nets features analysis to prove the reliability and conformance of the software is summarized in Section IV. Finally, the paper is concluded in Section V, followed by details of the modeled Petri Net in Section VI.

II. BASIC CONCEPTS & DEFINATIONS

MAIN power of Petri nets is their assistance for analysis of specified properties in the understudy software like reachibility, liveness, reversibility, safeness and boundedness, etc. Determining whether software shows a specific functional behavior or not, is a cardinal issue in verification and validation of the complex software & even in the modeling of the software. This contributes effectively in an early finding of whether or not a system modeled with Petri nets shows all desirable properties, as specified in the requirements & functional specifications.

The overall behavior of software can be described in terms of global states of software and their transformations [9]. In order to simulate the dynamic behavior of a system, a marking in PN is modified according to the enabling and firing rules of transitions which actually simulates the software code path execution pattern. Starting with initial marking, transition firings yield new markings that in turn give rise to further transition occurrences.

Some basic definitions of Petri Nets and their properties are discussed related to the modeled software. The associated terminology & information are mostly taken from [7] & [10].

Definition 1: (Petri net)[10]. A Petri net *PN* is 5 tuple: set of finite places P, set of finite transitions T, input function "I" and output function "O" and initial marking M_o : *PN* = (P,T,I,O,M_o). The input function 'I' is mapping from transition 'tj' to a collection of places I(tj), known as input places of 'tj'. The output function 'O' maps a transition 'tj' to a collection of places O (tj) i.e. I: $T \rightarrow P$, and O: $T \rightarrow P$. The directed arcs from places to transitions are shown by mapping I: $T \rightarrow P$ and the directed arcs from transitions to places are shown by mapping O: $T \rightarrow P$ such that $P \cup T = \emptyset$ and $P \cap T = \emptyset$. M: $T \rightarrow P$ defines a marking function in PN, where as initial marking is denoted by M_o. PN structure without M_o is defined as 4 tuple 'N' where N = (P, T, I, O). PN with a known initial marking is represented by (N, M_o).

Suppose I(tj) shows the set of input places of a transition tj, where $tj \in T$ & $pi \in P$, then input function is a mapping from $T \rightarrow P$ if pi is an input place of a transition tj and $pi \in I(tj)$. Further, O (tj) shows the set of output places of a transition tj, where $tj \in T$ & $pi \in P$, then output function is also a mapping from $T \rightarrow P$, if pi is an output place of a transition tj and $pi \in O(tj)$.

Definition 2: (Reachibility) [7]. A marking M_n is considered to be reachable from a marking M_0 if there exists a sequence of firings σ that transforms the markings from M_0 to M_n. If M_n is reachable from M_0 by σ, then we write $M_0 [\sigma > M_n$.

The *set of all reachable markings* from M_0 is denoted by $R(M_0)$ which is a distinct set of markings of PN such that, if $M_k \in R(M_0)$ and firing of transition $tj \in T : M_k [tj > M_k'$, then also $M_k' \in R(M_0)$.

Definition 3: (Deadlock)[11]. A transition tj is considered to be a dead transition at marking $M_k \in R(M_0)$ if there does not exist any reachable marking that enables a transition tj. A marking $M_k \in R(M_0)$ is considered to be in a deadlock. A PN is said to be *deadlock-free* if and only if there is no deadlock.

Definition 4: (Liveness)[7]. A transition $tj \in T$ is said to be *live* if for all $M_k \in R(M_0)$ there is marking reachable M_k' from M_k such that M_k' enables tj and PN (N, M_0) is *live* if all $tj \in T$: tj is live.

Definition 5: (Reversibility)[12]. Any marking M_k of PN is considered reversible only if for each marking M_k' which is reachable from M_k there exist a transition firing sequence 'σ' that regenerates the marking M_k from M_k'. A PN is considered reversible only if M_0 is reversible.

III. Proposed Model

SOFTWARE is designed initially with different diagrams. These diagrams assist in understanding of the basic architecture at different levels of the software [13]. The developed system IPAS, which is under study here is complex technical and financial data treatment software. It has twelve developed modules which processes the information at different levels to achieve the useful desirable functions as the end result of the software. These modules can be classified in four groups such as technical input, financial input, data filtration and output groups. The developed modules can be classified as:

1) *Technical Input Group*
   - *a)* Chronological Data Management *(CDM)*
   - *b)* Plant Availability Management *(PAM)*
   - *c)* Data Collection Management *(DCM)*
   - *d)* Metering Data Management *(MDM)*
   - *e)* Technical Limits Management *(TLM)*
   - *f)* Force-majeure Events Management *(FEM)*
2) *Financial Input Group*
   - *a)* Tariff Indexation Management *(TIM)*
   - *b)* Banked Energy Management *(BEM)*
3) *Data Filtration Group*
   - *a)* Complex Outages Management*(COM)*
4) *Output Group*
   - *a)* Capacity Invoice Management *(CPM)*
   - *b)* Energy Invoice Management *(EPM)*
   - *c)* Liquidated Damages Management *(LDM)*

The overall major high level module interaction can be depicted by the module interaction diagram as shown in Figure 1 and the high level detail of the input parameters is shown in Table 1. The major information exchange between the modules is shown. The data flow is identified from the input information to the ultimate processed output. As the software is designed and developed on hourly calculations as per the basic requirement of the PPA, so the ultimate objective is to generate the Capacity Payment, Energy Payment and Liquidated Damages Payment, firstly for any hour than for a day and finally for a month.

Input ID	Input Name	Input ID	Input Name
CDM: 1	PPA information with COD	TIM:5	WPIadjust
PAM: 1	Declaration Complete Versions with Intimation/Ack times	TIM: 6	Fuel Cost
PAM: 2	Dispatch Complete Versions with Intimation/Ack times	TIM:7	LIBOR
MDM : 1	DNEO from all Meters	TIM:8	KIBOR
MDM: 2	Ambient Temp from Weather Station	TIM:9	Supplementary Tariff
DCM : 1	DNEO for an Hour	TIM:10	CPP
DCM: 2	Ambient Temp for an Hour	TIM:11	EPP
TLM: 1	Complex Event Type	OUT: 1	Final Declaration for an Hour
FEM: 1	FM Event Start Date	OUT: 2	Final Dispatch for an hour
FEM: 2	FM Event End Date	OUT: 3	DNEO for Hour
FEM:3	FM Event Type	OUT: 4	Ambient Temp for an hour
TIM :1	Reference Parameters	LDM:2	Declaration based LDs
TIM :2	Reference Prices	LDM:1	Shortfall based LDs
TIM :3	Fxadjust	BEM :1	Banked Energy Deposit
TIM : 4	CPIadjust	BEM:2	Banked Energy Withdrawal

Table 1. Modules Input Information

Each part of basic module input parameter is injected in the system either by data entry or any equivalent form. A state is possible if and only if the state composition requirements are fulfilled such as if all the set of input parameters are available for a specific state only then the state is said to be possible. To show whether a state is possible or not a token representation is used in Petri nets. Absence of a token means that the state is not active in PN or it is not yet reached by other places in the dynamic system environment. The initial marking in this case may be defined as the availability of all the input parameters required for the data input places.

The module interaction diagram in Figure 1 can further be used to convert the software into equivalent PN model into the form of parallel process net (PPN) [10] with some exceptions. The states of the software become the places whereas the major functions become the transitions in PN. Input set related with every module can be decomposed in the set of places such that $P_{M1} = \{p1, p2, \ldots, pn\}$ becomes a finite set of places. By taking union of the input sets of all modules we can generate set of all places in PN such as $P = \{ P_{M1} \cup P_{M2} \cup P_{M3} \ldots \cup P_{MN} \}$ and $|P| > 0$. Similarly, all the major module functions are taken as finite set of transitions such that $T = \{t1, t2, \ldots, tn\}$ for the PN.

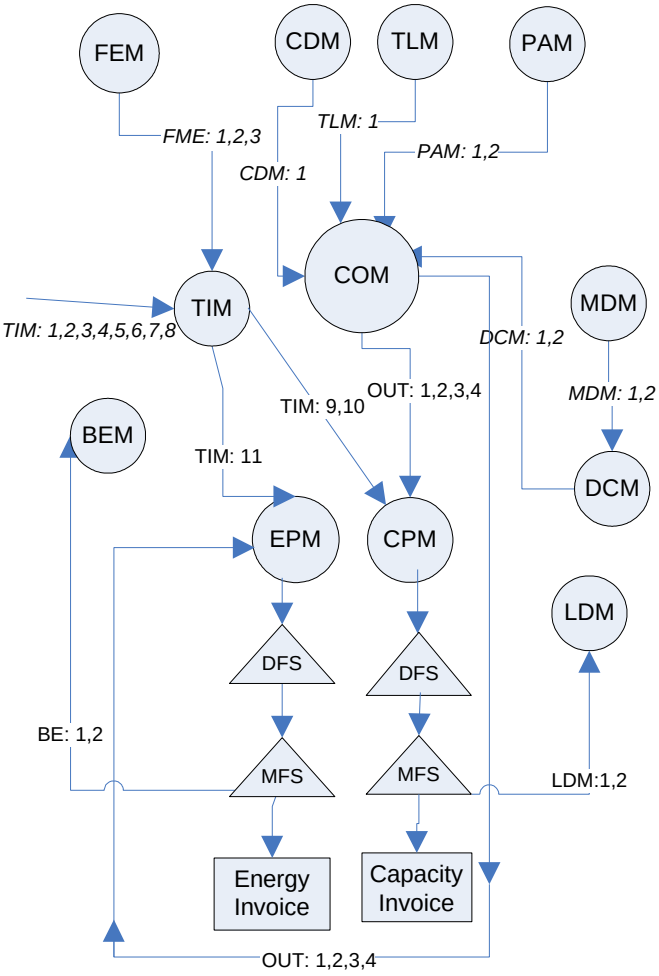

Figure 1. Module Interaction Diagram

When the enabled transition is fired it changes the token placement according to the transition firing rule. The sequence generated from firing of transitions produces the sequence of marking. Once a transition tj is fired, it consumes tokens from the I(tj) and add tokens to O (tj). The change in the state of the software can be simulated by the new positions of markings of tokens in the PN. Further this can be extended to fire next enabled transitions in the PN until the final outputs of the software are achieved.

The modeled petri net PN for IPAS is shown in Figure 2. As recalled from above, the markings represent the states of the software execution and transition tj can only be enabled if its set of input places $I(tj) \subset P$ have tokens. This consumes tokens from I(tj) and introduce new tokens in O(tj). This is also called the enabling rule for transitions. This activity results in a new state of the software referred as $M_k th$ marking.

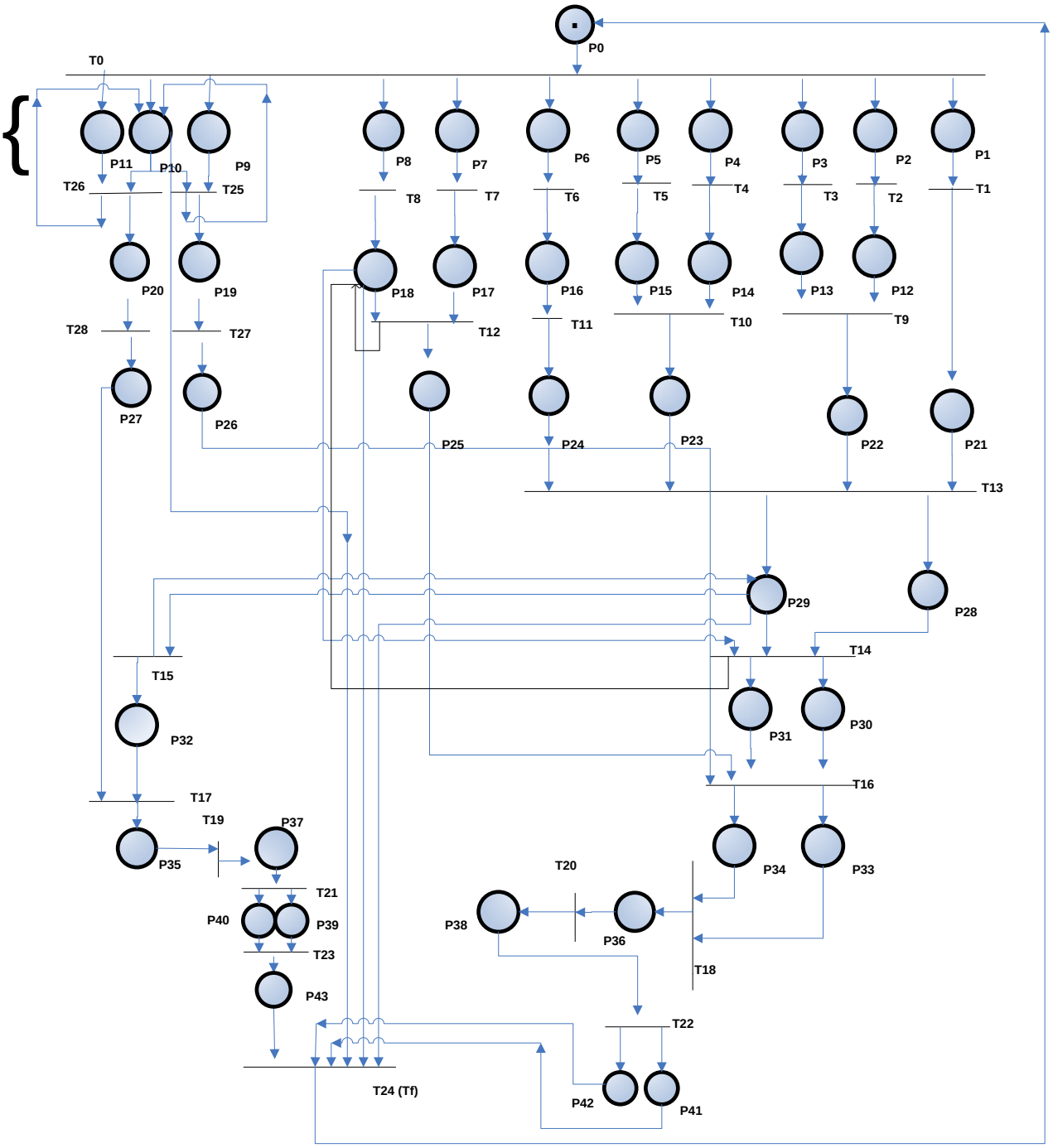

Figure 2. Modeled Petri Net (PN) for Developed Software

The initial marking in the modeled PN is the M_0 results in the productions of the next markings depending upon the transition enabling rules. Any marking M_k is followed by M_k' which contributes a sequence of firing sequence σ (sigma). This can be referred as the order of the firing transitions in the PN of the system, which in fact can be used to detect the flow of the code path coverage in the developed software. The final marking M_f is the final state of the software execution, in this case the generation of Capacity Invoice, Energy Invoice and Liquidated Damages Invoice.

In order to make the PN, first requirement is to generate the set of places P and set of transitions T, we have a set of 45 places and 25 transitions. The detail of each is shown in the Table 3 and Table 4 in Section V of the paper. The initial state of the software PN is $M_0 = (p_0)$, hence the initial token is in p_0 place, this enables the transition t_0 as the pre condition places such that all places in the set I (t_0) = {p_0} have tokens and the output places belonging to set O (t_0) = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11} donot have tokens. When t_0 is fired a new set of markings M_1 = (p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11) is generated. This will enable other transitions in the PN. Table 2 shows the production of all the markings from the software modeled PN. The above mentioned fired transition t_0, contributes as a first transition in the firing sequence $\sigma = t_0$.

Transition	**Modeled PN Generated Markings M_k** **Modeled PN Generated Firing Sequence σ**
	$M_0=(p_0)$
t_0	$M_1=(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11})$
	$\sigma = t_0$
t_1	$M_2=(p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{21})$
	$\sigma = t_0t_1$
t_2	$M_3=(p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12},p_{21})$
	$\sigma = t_0t_1\ t_2$
t_3	$M_4=(p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12},p_{13},\ p_{21})$
	$\sigma = t_0t_1\ t_2\ t_3$
t_9	$M_5=(p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},\ p_{21},p_{22})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9$
t_4	$M_6=(p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},\ p_{14},p_{21},p_{22})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4$
t_5	$M_7=(p_6,p_7,p_8,p_9,p_{10},p_{11},p_{14},p_{15},p_{21},p_{22})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5$
t_{10}	$M_8=(p_6,p_7,p_8,p_9,p_{10},p_{11},\ p_{21},p_{22},p_{23})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}$
t_6	$M_9=(p_7,p_8,p_9,p_{10},p_{11},p_{21},p_{22},p_{23},p_{16})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6$
t_{11}	$M_{10}=(p_7,p_8,p_9,p_{10},p_{11},p_{21},\ p_{22},p_{23},p_{24})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}$
t_{13}	$M_{11}=(p_7,p_8,p_9,p_{10},p_{11},p_{28},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}$
t_7	$M_{12}=(p_8,p_9,p_{10},p_{11},\ p_{17},p_{28},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7$
t_8	$M_{13}=(p_9,p_{10},p_{11},\ p_{17},p_{18},p_{28},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8$
t_{14}	$M_{14}=(p_9,p_{10},p_{11},p_{17},p_{18},p_{29},p_{30},p_{31})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}$
t_{12}	$M_{15}=(p_9,p_{10},p_{11},\ p_{18},p_{25},\ p_{29},p_{30},p_{31})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}$
t_{25}	$M_{16}=(p_{10},p_{11},p_{19},p_{18},p_{25},p_{29},p_{30},p_{31})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}$
t_{27}	$M_{17}=(p_{10},p_{11},p_{18},p_{25},p_{29},p_{26},p_{30},p_{31})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}$
t_{16}	$M_{18}=(p_{10},p_{18},p_{11},p_{33},p_{34},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}$
t_{18}	$M_{19}=(p_{18},p_{11},p_{10},p_{36},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}$
t_{20}	$M_{20}=(p_{11},p_{18},p_{10},p_{38},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}$
t_{22}	$M_{21}=(p_{11},p_{18},\ p_{10},p_{41},p_{42},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}$
t_{26}	$M_{22}=(p_{18},p_{10},p_{20},p_{41},p_{42},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}$
t_{28}	$M_{23}=(p_{10},p_{18},p_{27},p_{41},p_{42},p_{29})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}$
t_{15}	$M_{24}=(p_{10},p_{18},p_{27},p_{29},p_{32},p_{41},p_{42})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}$
t_{17}	$M_{25}=(p_{10},p_{18},p_{29},p_{35},p_{41},p_{42})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}$
t_{19}	$M_{26}=(p_{10},p_{18},p_{29},p_{37},p_{41},p_{42})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}\ t_{19}$
t_{21}	$M_{27}=(p_{10},p_{18},p_{29},p_{39},p_{40},p_{41},p_{42})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}\ t_{19}\ t_{21}$
t_{23}	$M_{28}=(p_{10},p_{18},p_{29},p_{43},p_{41},p_{42})$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}\ t_{19}\ t_{21}\ t_{23}$

t_f	$M_{29}=(p_0)$
	$\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}\ t_{19}\ t_{21}\ t_{23}\ t_f$

Table 2. Modeled Petri Net (PN) Generated Markings & Firing Sequences

IV. Analysis

MODELED petri net can now be verified in reference with the various properties as discussed in the definitions in a previous section.

Firstly, reachibility property of PN is used to simulate achievable states of the software. Let, we want to verify whether or not a desirable marking $M_{16} = (p_{10},p_{11},p_{19},p_{18},p_{25},p_{29},p_{30},p_{31})$ is reachable from a current system marking $M_3= (p_3,p_4,p_5,p_6,p_7,p_8,p_9,p_{10},p_{11},p_{12},p_{21})$ while considering from the possible markings from the Table 2, such that M_3 is M_k marking and M_{16} is M_k' then by applying the firing sequence $\sigma = t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}$, we can reach from M_k to M_k' and vice versa is true from firing sequence $\sigma = t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}\ t_{19}\ t_{21}\ t_{23}\ t_f\ t_0t_1t_2$. Hence, $M_3\ [\sigma > M_{16}$ is true and the reachibility property is proved resulting that we can reach any desirable state from any possible desired state in the software.

Secondly, deadlock detection property is used to determine whether or not the modeled PN have any deadlock existence [14]. Considering the possible markings in the Table 2, we can observe from PN iterations that there does not exist any transition tj that remains disabled by any reachable marking of the system or every transition is fired and is not permanently disabled in the modeled PN. Hence, the deadlock is not detected in the modeled software.

Thirdly, the deadlock freeness of the modeled PN can be proved due to the absence of the deadlock in the system. This means that every I (tj) and O (tj) work under transition firing rules for the PN. Hence, the modeled software is live in nature.

Lastly, the reversibility property is used to simulate that the system is reverserable to a certain desirable state by a sequence of firing sequences σ which defines the sequence of firing transitions. This helps to understand the code path execution analysis of the modeled system. The possible markings in Table 2 shows that the overall modeled PN is starting with the initial marking $M_0= (p_0)$, can be reversed to its initial marking $M_0= (p_0)$ by executing the firing sequence $\sigma = t_0t_1\ t_2\ t_3\ t_9\ t_4\ t_5\ t_{10}\ t_6\ t_{11}\ t_{13}\ t_7\ t_8\ t_{14}\ t_{12}\ t_{25}\ t_{27}\ t_{16}\ t_{18}\ t_{20}\ t_{22}\ t_{26}\ t_{28}\ t_{15}\ t_{17}\ t_{19}\ t_{21}\ t_{23}\ t_f$. Hence, the modeled software is also reversible to its initial state.

Therefore, above analysis converges to a point that the developed system that is modeled through PN is reachable, deadlock free and reversible in nature. This means that the software is in the light of its design and information exchange pattern is workable, stable, dependable and its architecture supports the dynamic code execution. Furthermore, it also identifies the firing sequences of the transitions with respect to major functional activity in the software and can assist in proving the conformance to the requirement specification for the developed software.

V. Conclusion

DYNAMIC behavior modeling & analysis is an important activity for the software verification and validation that leads to the overall software reliability and conformance to requirement specifications. The formal approaches are computational intensive but that can be easily utilized in evaluation of different developed software at high levels. In this paper, we discussed a corporate energy invoice and automation software modeling in a class of nets called Petri nets and then their feature analysis in order to simulate and determine whether or not the developed system is deadlock free, reachable and reversible, which assists in the identifying the developed software as workable, stable, dependable for performing functional activity that obey the rules in the requirement specifications.

VI. Appendix

Places Table

Table 3. Modeled Petri Net (PN) Places Table

Place	Module	Place Name
p_0		Initial State (Master state)
p_1	CDM	PPA Information
p_2	PAM	All versions of Complex Availability Declaration are available for an IPP
p_3	PAM	All versions of Dispatch Instructions are available for an IPP
p_4	MDM	Ambient Temperature is available
p_5	MDM	All Meter Readings are available
p_6	TLM	Data available for TLM
p_7	FME	FME data is available to the system
p_8	TIM	TIM data is available to the system required for supplementary tariff
p_9	TIM	Unique indexations parameters of TIM
p_{10}	TIM	Attributes of TIM common to CPP&EPP
p_{11}	TIM	Unique indexations parameters of TIM to be used in EPP
p_{12}	PAM	Valid Complex Availability Declaration available to the system

p_{13}	PAM	Valid Dispatch Instruction available to the system
p_{14}	DCM	Value of ambient temperature is available to the system after rounding off.
p_{15}	DCM	Commulative Dneo is available to the system.
p_{16}	FME	TLM event is available to the system.
p_{17}	TIM	FME data (Start date, End date & Type) is available.
p_{18}	TIM	TIM data is available to the System for the calculation of Supplementary Tariff
p_{19}	CPM	CPP is available
p_{20}	EPM	EPP is available
p_{21}	CDM	PPA Information available to the System
p_{22}	PAM	Final Declaration & Dispatch instruction is available.
p_{23}	DCM	Commulative Dneo and rounded value of ambient temperature is available.
p_{24}	TLM	Hourly TLM event is available.
p_{25}	TIM	Supplementary Tariff is prepared.
p_{26}	CPM	Indexed CPP is available to the system.
p_{27}	EPM	Indexed EPP is available to the system.
p_{28}	PAM	SO & FO Allowances
p_{29}	PAM	Combined Technical data on the basis of an hour
p_{30}	LDM	LDs generated
p_{31}	CPM	Qualifying Capacity
p_{32}	EPM	Qualifying Energy
p_{33}	CPM	LDs payment
p_{34}	CPM	Capacity Payment
p_{35}	EPM	Daily Fact Sheet for Energy
p_{36}	CPM	Daily Fact Sheet for Capacity
p_{37}	EPM	Monthly Fact Sheet for Energy
p_{38}	CPM	Monthly Fact Sheet for Capacity
p_{39}	EPM	Qualifying Energy
p_{40}	BEM	Data from Banked Energy
p_{41}	CPM	Final Invoice for Capacity
p_{42}	LDM	Final Bill for Liquidated Damages
p_{43}	EPM	Final Invoice for Energy

Transition Table

Table 4. Modeled Petri Net (PN) Transitions Table

Transition	Module	Transaction Name
t_0		Master Reset
t_1	CDM	Fetch PPA information from Chronological Data Management.
t_2	PAM	Provide valid Declaration instructions to the system
t_3	PAM	Provide valid Dispatch instructions to the system
t_4	DCM	Ambient Temperature Rounding off
t_5	DCM	Preparation of Commulative Dneo
t_6	TLM	Provision of TLM event to the system
t_7	FME	Provision of FME data to the system
t_8	TIM	Provision of TIM data(other than FME data) to the system for the preparation of Supplementary Tariff
t_9	PAM	Provides the latest valid Declaration/Dispatch pair for an IPP
t_{10}	DCM	Provision of Dneo & Ambient Temperature to the system
t_{11}	TLM	Provision of TLM event to the system for an hour
t_{12}	TIM	Preparation of Supplementary Tariff
t_{13}	PAM	Acquisition of Technical Data
t_{14}	CPM	Data acquisition for CPP
t_{15}	EPM	Data Processing for the calculation of EPI
t_{16}	CPM	Data Processing for the calculation of CPI
t_{17}	EPM	Preparation of DFS of Energy
t_{18}	CPM	Preparation of DFS of capacity
t_{19}	EPM	Preparation of Monthly Fact Sheet of Energy
t_{20}	CPM	Preparation of Monthly Fact Sheet of Capacity
t_{21}	BEM	Processing with data from Banked Energy
t_{22}	CPM	Preparation of CPI and LDBs
t_{23}	EPM	Preparation of Energy Invoice
t_{24}		Loop back to the Initial state
t_{25}	CPM	Calculation of CPP
t_{26}	EPM	Calculation of EPP
t_{27}	CPM	Provision of indexed CPP
t_{28}	EPM	Provision of indexed EPP

VI. ACKNOWLEDGMENT

WE are grateful to Al-Khawarizmi Institute of Computer Science, University of Engineering & technology, Lahore, Pakistan, for providing us the platform to work on the Invoice Processing & Automation System (IPAS). We also acknowledge WAPDA, Pakistan for project funding that helped us to complete the initiative.

VII. REFERENCES

[1] Roberto Pietrantuono, Stefano Russo, Kishor S. Trivedi, "Software Reliability and Testing Time Allocation: An Architecture-Based Approach," IEEE Transactions on Software Engineering, vol. 36, no. 3, pp. 323-337, May/June, 2010.

[2] Hunter P., Focusing on finance [financial software], Institution of Engineering and Technology, Stevenage, ROYAUME-UNI, 2010, vol. 5, no7, pp. 52-55, ISSN 1750-9645

[3] Di Paolo, I.F.; Cordeiro, T.D.; Sena, J.A.S.; Fonseca,

M.C.P.; Barra, W.; Barreiros, J.A.L.; Silva, O.F.; , "Creational Object-Oriented Design Pattern Applied to the Development of Software Tools for Electric Power Systems Dynamic Simulations," Latin America Transactions, IEEE, vol.8, no.3, pp.287-295, June 2010

[4] Eduardo B. Fernandez, Xiaohong Yuan, An Analysis Pattern for Invoice Processing, 16th Conference on Pattern Languages of Programs, August 28-30, 2009, Chicago.

[5] WANG Quan-bin, Design and Implementation of Invoicing Management System of Storage Enterprises, Journal of Tonghua Normal University, April 2009.

[6] Jamshaid I. Janjua & Dr. Zubair A. Khan, "Real-time, Efficient E-Infrastructure Development Framework for Corporate Energy Sector", IEEE International Conference on Intelligence and Information Technology (ICIIT-2010), Lahore, Pakistan, 28th-30th October 2010, IEEE Catalog Number: CFP1071K-PRT, ISBN: 978-1-4244-8138-5.

[7] Murata, T., "Petri nets: Properties, Analysis and Application", In Proceedings of IEEE, Vol. 77, No. 4, pp. 541-580, 1989.

[8] Reisig, W. Petri Net: An Introduction, Springer-Verlag, 1985.

[9] A. Karatkevich, Dynamic Analysis of Petri Net-Based Discrete Systems, LNCIS 356, Springer-Verlag, Berlin Heidelberg, 2007.

[10] Peterson, J. L., Petri Net theory and the Modeling of Systems, Prentice-Hall, 1981.

[11] Z. Li, and M. C. Zhou, On siphon computation for deadlock control in a class of Petri nets, IEEE
Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(2008): 667-679.

[12]Reveliotis, S., A Necessary and Sufficient Condition for the Liveness and Reversibility of Process-Resource Nets With Acyclic, Quasi-live, Serializable, and Reversible Process Subnets, Automation Science and Engineering, IEEE Transactions, pp 462-468 , Volume: III, Oct. 2006

[13] Li Xiaoli,Long Xiang,Bao Xiaolu,Li Hu, Formal definition and characteristic analysis of UML sequence diagram Journal of Beijing University of Aeronautics and Astronautics [J. Beijing Univ. Aeronaut. Astronaut.]. Vol. 36, no. 3, pp. 350-352, 362, Mar 2010.

[14] F. Ahmad, H. Huang, X. L. Wang, Petri net modeling and deadlock analysis of parallel manufacturing processes with shared-resources, Journal of Systems and Software, vol. 83, pp. 675-688, 2010 .

[15] Sun, T.H., Cheng, C.W., Fu, L.C., A Petri net based approach to modeling and scheduling for an FMS and a case Study. IEEE transactions on Industrial Electronics 41 (6), 593–601, 1994.