Abstract—This paper investigates the different uses of Petri net main classes or categories for systems and software engineering. For this work Petri nets are classified into four main categories which are i) elementary nets, ii) general Petri nets, iii) higher order nets and iv) timed Petri nets or Petri nets with time. Each category has a specific use for systems engineering and software engineering. The main differences between these categories are clearly defined. It is shown how Petri Net classes can be made to fit in a semi structured approach used for systems design or requirements engineering. A simple case study of a vending machine is used for illustration.

Keywords— Higher order Petri nets, Petri nets, Requirements engineering, Software modeling.

I. INTRODUCTION

Traditionally Petri nets have been used to model both hardware and software systems. They can be applied at different levels of granularity in the software or hardware engineering process. Petri nets can model very well higher levels of abstraction as is done in Fundamental Modeling Concept Approach [16] down to program level or very low levels of programming logic [9]. Petri nets can describe a high level manufacturing process managed via hardware down to multi layered networking protocols and programming instructions in any language. They are special formalisms that share a dual identity. Petri nets can also be represented formally using formal languages and notations or in graphical form. Compared with other formalisms Petri nets are more adequate for visualization and quick understanding. Petri nets can be verified and simulated. All Petri net classes can be converted to timed Petri nets for performance modeling.

Some main uses of Petri nets being expanded on are i) Software engineering modeling: Petri nets have been used for requirements elicitation, system modeling, supporting UML diagrams for verification and execution [2]-[9]. Most UML diagrams ranging from class diagrams to use cases and activities have been supported via Petri nets. ii) Communication systems modeling: Petri nets are used to describe and model communication protocols [2], different types of networks, etc. iii) Hardware modeling: Interaction of system components, system on a chip, CPU etc. iv) Manufacturing: equipment configuration, setups. Requirements engineering deals with the critical understanding of system issues. Different viewpoints are used. When examining behavior conceptual, logical and physical views are important.

Petri nets are a visual formalism based on sound mathematical principles that can properly explain the ‘physical’ behavior of a system and its components. Petri nets share many common properties with other formalisms.

Petri nets have extensive use in information technology as evidenced in literature. They can be used for many different issues in the system and software engineering processes.

II. PROBLEM FORMULATION

In principle system and software engineering [1], [13]-[14] are related. The success of systems implementation depends on the integration of both.

Petri nets can be classified into four main categories which are i) elementary nets, ii) Petri nets, iii) higher order nets [10]-[11] and iv) timed Petri nets. These categories have further subdivisions that can become quite complex and are not dealt with here. All these classes are suited for some very specific problem modeling.

Unfortunately there is no clear guideline that indicates which Petri net class or classes should be used for systems and software engineering. Each class offers different features from another class. This is evident from the vast literature available. There are overlaps in the types of Petri nets used so these can be generalized for both systems and software engineering which are considered. Sometimes a difficulty occurs when to use a particular Petri net class instead of another. To complicate matters, different classes have similar properties that might make the Petri nets look similar when in reality there are many other attributes included. Another problem is that a Petri net class that is unnecessary complex is used to represent a simple system. This could have easily been shown using a more elementary class. At different levels of requirements elicitation, different models are normally required.

III. PETRI NET CLASSIFICATION

Requirements [1], [12]-[14] have to specify what a system does and how it does it. Systems are composed of a series of interacting components either hardware or software or a combination of both. System processes expressible in graphical notations can be modeled as Petri nets. System processes are derivable from sequence of observations made about system behavior. In this approach or work a simple method is suggested. Complex transformations are avoided.
and the idea is to generate an initial Petri net that will serve to create more complex and higher order nets if necessary. The idea is to formally or informally transform requirements directly into a simple Petri net or elementary net. This is a good starting point. Later more complex Petri nets like higher order nets and timed Petri nets can be derived from the initial net. The information used to create the initial net can be derived from various sources of information, formal, informal or both.

The analogy is to use a “correct-by-construction” approach where the Petri net can be validated at different stages before reaching the final detailed model or models. The final model would serve to obtain the detailed requirements for producing the system. Fundamental flaws or incorrectness would be sorted out initially or as work progresses.

A. Petri Net Categories and Classsification

According to [10]-[11] Petri nets are normally classified into three major categories. In this work we classify Petri nets into four major types or levels. These are i) Elementary Net Classes, ii) Main Petri Net Classes, iii) Higher Order Net Classes and iv) Timed Petri Nets (TPNs) [13]. Refer to table 1 and 2. TPNs are very important for system and software modeling and there are extensive references to these being used directly.

B. Elementary Nets

In the elementary net categories [15] the Petri net structures are rather basic and restricted. Condition event nets (C/E) are structurally similar to elementary nets (EN). These structures are identified as having simple structural qualities in Petri net terminology. In this category places can contain at most one token. Input arcs and Output arcs connecting to a transition remove and output one token. This means that places represent boolean information. Condition event net systems are pure/simple/1-live. There is backward and forward reachability where every event has a chance to occur.

C. General Petri Nets

The second category of Petri nets [10]-[12],[15] are still quite similar to the previous one. There are structural similarities. In fact elementary net structures could be Petri nets with certain restrictions. The Petri nets in this category are still simple in structure but places can contain more than one token and arcs can have multiple values removing more than one token at a go. In this category there are i) Ordinary Place Transition nets, ii) Free choice nets, iii) S-System state machines, iv) T-System marked graphs, etc. It is possible to classify Place transition systems in this category. Tokens are still unstructured but they can represent integer values.

D. Higher Order Nets

The third category as its name implies has significant differences from the previous ones [4]-[6]. Here Petri nets are no longer simple and start to resemble programmable artifacts that are graphical and retain basic structural and operational properties of Petri nets.

Some higher order nets are: Algebraic Petri nets, Predicate

<table>
<thead>
<tr>
<th>Level</th>
<th>Petri Net</th>
<th>Category type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Elementary Nets</td>
<td>i. simple structured nets</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ii. well behaved</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>iii. boolean tokens</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>iv. simple behavior</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>General Petri Nets</td>
<td>i. arcs can have multiple values</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ii. place capacity greater than one</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>iii. unstructured tokens</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>iv. tokens as integer values</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>v. many sub classes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>vi. e.g. P/T nets, S-nets, T-nets etc</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Higher Order Petri Nets</td>
<td>i. highly structured places</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ii. representing records, sets, objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>iii. well formed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>iv. complex data types</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>v. complex transition firing rules</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>vi. arc structures</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>vii. complex structures overall</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Timed Petri Nets</td>
<td>Variants of the above classes with</td>
<td></td>
</tr>
</tbody>
</table>
Transition nets (Prt), Product nets, environmental nets, object oriented Petri nets, colored Petri nets etc [10]-[11]. Higher order structures are characterised mainly by the token types that can represent anything like an object, record, data sets, complex data types, etc. Tokens are well structured or well formed and can represent complex values or structures defined on abstract data types. Transitions can have complex firing rules that match special token data sets. The rules can be programmed using special languages like ML as in the case of CPNs [5]-[7]. Arcs can contain special inscriptions. These Petri nets are suitable for detailed modeling and system behavior. They offer a great deal of flexibility and simulations that are close to the real world or actual environment. This class of Petri nets greatly increases the modeling power of these structures, at the same time these structures become very complex and are no longer simple to create. They require special expertise. To work with these classes programming knowledge is a prerequisite.

E. Timed Petri Nets

The fourth category represents timed Petri nets (TPNs) [12]. More than a category in its own right, TPNs are considered as extensions. However here they are considered as separate structures because they offer a different modeling perspective and the time dimension which is important for system and software structures. This implies that it is possible to convert/transform an elementary net into a timed Petri net. Normally the time values are assigned to transitions. It is possible to assign the time values to arcs and places also. Some common types of time Petri nets are deterministic timed Petri nets (DTPNs), stochastic Petri nets (SPNs), GSPNs [12], Q-nets etc. TPNs are very important for performance modeling, simulation and analysis related to bottleneck problems in systems.

IV. PROBLEM SOLUTION

A. Petri Net Transformation Relationships

The diagrams in fig. 1 illustrates the possible relationship between the four main categories of Petri nets useful for modeling. This diagram illustrates that all net types are directly transformable from one to another. Obviously this transformation does not necessarily imply that the nets will and must look similar to one another.

From a practical point of view transformation from one class or category to another can be done directly. An EN system can be transformed into a TPN by adding time values to transitions. A higher order net can be created from a general Petri net or elementary net by adding token types, firing rules and arc inscriptions. A higher order Petri net can also be transformed into a TPN by adding time values to the transitions [12]. A higher order Petri net can be created from a P/T net or vice-versa. From the TPN structure it is possible to create a higher order Petri net or an EN or TPN etc. For modeling system and software behavior the ideal starting point might be either the EN system or a Place transition net.

Transformation from the P/T net into the EN is quite simple as actually it is a reduction of the structural properties of the P/T net. Different literature exists for formalizing the possible correspondence and transformations. Formalizing the transformations might reduce the actual usefulness of this approach because it might be better to leave it open to the user to develop the models accordingly. Hence the diagram in fig. 1 serves as a reference map or guideline to what is possible.
B. Semi Structured Transformations

For i) Initial system modeling: This is normally done at a high level of abstraction. It is ideal to start off using either main Petri nets or elementary nets. These are suitable to illustrate basic system operations and create working models. ENs and main Petri nets are suitable for modeling very elementary protocols or device handling or basic operational logic. At this stage emphasis is placed on conditions and events as well as understanding the main features. ii) Advanced or detailed system modeling: In this case EN and main Petri net classes might not be suitable as more information and detail needs to be represented or captured in the Petri net structure. Higher Order nets like CPNs, etc. might be more suitable for this. The detailed system model can be constructed using the information of the initial system model. The emphasis here is more on detail and expression handling.

iii) Timed model: The timed model can be constructed directly from the initial model or from the advanced model for experimenting with timing issues.

Different Petri net models can be targeted towards the needs of different system stakeholders e.g. one non technical person is interested in the top level functionality, so the EN are more suitable, but the systems engineer is interested in the low level detailed functionality so the higher order net is more appropriate.

At any step in the process if one is satisfied with the model there is no need to construct further models. If refinement is needed, the model can be modified or more detailed models created.

V. CASE STUDY

A. Simple Vending Machine

A case study of a simple vending machine that operates by coin insertion is considered. In [15] a different view of a vending machine is given. The vending machine’s main steps summarized are: i) coin insertion, ii) select and dispense item and iii) refill item. The initial net to be constructed according to the semi structured transformation in fig. 2 is the elementary net. This is shown in fig. 3. This structure is used as the starting point for constructing the basic Petri net, higher order nets and/ or the timed Petri net. All the models created are executable.

B. Elementary Net

The elementary net in fig. 3 shows the restricted behavior of the vending machine. The elementary net is similar to a condition event net system. The Petri net is very simple. Places can hold only 1 token and arcs remove only one token. It is a 1-safe Petri net. This structure depicts simple behavior and is easily constructed and understood.

C. General Place Transition Petri Net

The detailed Petri net or place transition net is built from the elementary net in 4.1. This is shown in fig. 4. Places can contain more than 1 token. The net is not necessary conflict free and is not simple. This Petri net contains more reasoning about the system. The execution logic is more complex than the EN system. This structure allows for the possible selection of items using conflicting transitions. The refilling of products has been separated from the main structure. A coin counter that allows maximum insertion of 4 coins is introduced. When a coin is inserted it is possible to accept it or reject it. Arc weights can be used to decide which items to dispense. In this example, for product A, 3 coins are necessary, so dispense product A is enabled if there are 3 tokens in coin accepted place. When an item is dispensed an item is removed and a refill takes place automatically so there are always 4 items buffered to be dispensed for each product.
available.

D. Higher Order Petri Net

The higher order net is constructed from the general place transition Petri net in 4.2. This is shown in fig. 5. Token types are based on sets. Some token types are coin, change, amount and items. The operational logic of the colored Petri net is closely imitates the real machine operational logic. It includes many details. Coins are inserted into the vending machine using a random function to obtain the value. Product selection is done using a function called check. Once a product is selected the change is computed and given to the customer. Items have an actual name like A for product A and a quantity value. Thus a value (“A”, 4) means that there are 4 items of product type A. One place can be used to manage all items.

E. Timed Petri Net

The timed Petri net can actually be constructed from the nets described in 4.1, 4.2 or 4.3. The net in 4.2 is converted to a TPN. For constructing the TPN it is not necessary to alter the underlying net structure but add time to transitions, places or arcs. Normally timed transitions are used. This is shown in fig. 6. This timed Petri net looks almost identical to the vending machine general Petri net shown in fig. 4. All that has been changed are the transition types. The immediate transitions in fig. 4, were converted to timed transitions. This is the only change carried out and the underlying structure is unmodified. Two types are used: i) Deterministic or fixed time, ii) time obtained from uniform distribution. The transition insert coin has a time value of [50,100] which could represent the time in seconds required to insert a coin by a customer. If the colored Petri net in fig. 5 is converted or extended to include time, the structure would look similar.

VI. RESULTS

The aim of this work was to identify the different uses of Petri net classes in systems design process and see how they can fit together. Traditionally users look only at a single class of Petri nets and cannot see the full picture of the different classes at a single glance. In this work it has been shown that all these classes are all useful at one stage or another. Petri net structures can practically represent various forms of dynamic behavior for most systems and software artifacts. Normally it is better to progress from simple structures to more complex ones. As fig. 1 indicates transformation from one class to another is possible at any stage. Some transformations imply adding more information, whilst others imply reducing the Petri net structure and removing information. A semi structured non formal approach is presented to simplify the idea and present the working importance of Petri nets to a large group of users.

All the models created are working models which are fully functional, deadlock free, etc. The models were created in the given order and can be developed further. From a modeling perspective the EN and general Petri nets are less complex than the higher order net. The TPN is useful when the underlying structures are correct and timing issues need to be analyzed. This is the case with systems like real time, critical, etc.

Other results like Petri net static and dynamic analysis can be investigated. This is carried out on the models that need to be analyzed. The results would be useful for comprehending the underlying structures, complexities involved and Petri net
properties.

VII. CONCLUSION

This paper has very briefly explained the main Petri net classes and their uses in systems engineering. It has explained how to use these classes in a semi structured approach. The case study presented is quite simple. In reality there are many difficulties with different systems requiring special and dedicated attention. It is possible to combine this work with a lot of research on Petri net theory.

The idea of combining different classes creates many issues which need to be solved appropriately. It has not been explained how to carry out the transformations. This could be done very simply by just creating completely new models and keeping some parts of the initial model, i.e. informally or formally. The transformations have not been formalized or explained in detail. This was done as the aim of this paper is to present the concept of combining Petri net main classes. A lot of work can be done to formalize the transformations. It is possible to find better models of how the classes could contribute between each other. The main classes are subdivided into many other classes e.g. S-nets, T-nets, different levels of safe nets, etc, complicating things. Other issues are: i) different Petri net case tools are needed and proper integration is required, ii) reduction of the model’s structure will normally mean information is lost, iii) the final Petri nets might result with many differences from the initial ones.

REFERENCES


