Supporting Requirements Engineering With Different Petri Net Classes

A. Spiteri Staines

***Abstract*—**This paper considers how Petri net main classes or categories can be used to support systems and software requirements engineering processes. In general Petri nets are classifiable into four main categories which are i) elementary nets , ii) Normal Petri nets , iii) higher order nets and iv) timed Petri nets or Petri nets with time. Apart from some major fundamental differences, each category has a specific use for systems engineering and software engineering and thus they can clearly help with requirements engineering issues. In this work the main differences between these categories are briefly explained. It is also shown how these Petri net classes can be made to fit in a semi structured approach. This is very useful for the analysis and design of a whole range of system types. A simple case study of a vending machine is used for illustrating this work.

***Keywords*—** Higher order Petri nets, Petri nets, Requirements engineering, software modeling,

I. Introduction

PETRI NETS are special graphical formalisms based on sound mathematical principles that can properly explain the 'cyber physical' behavior of systems and their components. They have over three decades of use. Petri nets share many common properties with other formalisms. Petri nets can be verified and simulated. Petri nets can normally be converted to time Petri nets for simulation and performance modeling. Petri nets have a dual identity. They can be represented graphically and non graphically. Compared with other formalisms Petri nets are preferable for visualization and comprehension by different stakeholders.

Petri nets have been used to model i) hardware, ii) software systems, iii) communication systems, iv) manufacturing and v) software modeling. Some examples are: intelligent transport system analysis, analysis and design of flexible manufacturing systems, requirements engineering of embedded applications, supporting UML notations and their transformations, most UML diagrams ranging from class diagrams to use cases and activities have been supported via Petri nets. Other examples are LAN and WAN requirements analysis and performance estimation, computer hardware architecture modeling, architectural specification for distributed systems, system on a chip verification, hybrid systems analysis, wireless network routing, real time systems critical performance estimation, fault diagnosis, workflow analysis, traffic control, e-commerce system modeling, etc [17]-[21]. Many more uses can be found. In general Petri nets have been used for requirements elicitation, system modeling, supporting UML diagrams for verification and execution [2]-[9]. The extensive use of Petri nets in information technology is evidenced in many different sources and books.

Petri nets 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 almost any language.

Modern systems are normally a hybrid of software and hardware in varying amounts. Many research projects challenge traditional development by requiring new methods of integrating system components. Thus more stringent requirements are placed at the analysis and design stages. Today many industries depend heavily on software and hardware integration. New methods of software engineering are required to solve these problems. However it is not always an easy task and involves a lot of time.

Petri nets can be used for many different issues in the system and software engineering processes. System and software engineering processes are intricately complex issues and cater for a variety of systems ranging from simple to complex systems that are very difficult to specify correctly. It is for this reason that many different sub classes or different Petri net types have been formally defined and created. Every one of them has its own particular specialized use which is very interesting to solve and address a particular problem. But having so many different types of Petri nets creates issues as how to choose the best Petri net type or class for a particular problem. Petri nets can be used to test and verify rigorously, system functionality and design implications at the very initial phases of analysis. They can be used to determine critical performance issues related to time and failure. Petri nets can be used to create different viewpoints of a system related to the conceptual, logical or physical design structures.

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. Requirements engineering manages the whole process of requirements elicitation, design and requirements expression. But at the same time the

Manuscript received Jul 15, 2010. Anthony (Tony) Spiteri Staines, is with the Department of Information Systems, Faculty of ICT, University of Malta, (corresponding phone: 00356-21373402,e-mail: toni_staines@yahoo.com)

requirements engineering process is not restrictive as to which model or notation is to be used at a particular stage. These serve as tools for supporting requirements engineering.

Different types of Petri nets have been created for different problem scenarios. This is sometimes done for a one off problem. Petri nets are classifiable under three or four main types.

Finding the right Petri net type for a problem is no easy task. This is because it becomes complicated to select the correct class and type creating a problem. Classifying Petri nets into four major types indicates that it is not easy to select the correct Petri net type. This is further complicated by the fact that what is required initially at the early analysis stage might become obsolete for the design stages. To explain this issue: e.g. if we take a particular system, initially a simple Petri net model of it could suffice for the general system stake holders but as the design stage is entered the system engineers require a complex performance model for clarification, i.e. a timed colored Petri net is needed. This implies that if we have a simple Petri net model, this model is not so useful as more detail is required. On the other hand if there is a simple straightforward problem, modeling it using a complex Petri net is unjustified.

Petri nets four main categories 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. Normally the more detailed or complex classes are suited for 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 regarding 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

TABLE I
PETRI NET MAIN CLASSES

Level	Class/Category type		Description
1	Elementary Nets	i.	simple structured nets
		ii.	well behaved
		iii.	boolean tokems
		iv.	simple behavior
2	General Petri Nets	i.	arcs can have multiple values
		ii.	place capacity greater than one
		iii.	unstructured tokens
		iv.	tokens as integer values
		v.	many sub classes
		vi.	e.g. P/T nets, S-nets, T-nets etc
3	Higher Order Petri Nets	i.	highly structured places
		ii.	representing records, sets, objects
		iii.	well formed
		iv.	complex data types
		v.	complex transition firing rules
		vi.	arc structures
		vii.	complex structures overall
4	Timed Petri Nets	Variants of the above classes with timing elements introduced	

TABLE 2
PETRI NET CLASS USES

Level	Class/ Category type		Description
1	Elementary Nets	i.	basic top level system description
		ii.	simple modelling,
		iii.	abstraction of top level processes
		iv.	initial systems analysis and design
		v.	teaching Petri nets
		vi.	basic network structure models
		vii.	preliminary analysis
2	General Petri Nets	i.	initial systems analysis and design
		ii.	system composition and decomposition
		iii.	teaching Petri nets
		iv.	similar to EN classes
3	Higher Order Petri Nets	i.	more program oriented solutions
		ii.	problems requiring complex modeling rules e.g. real time error handling
		iii.	fine granularity modeling
		iv.	complex modeling as close as possible to the real solution
4	Timed Petri Nets	i.	simulation and behavior analysis
		ii.	performance modeling with time
		iii.	bottleneck identification
		iv.	system failure identification

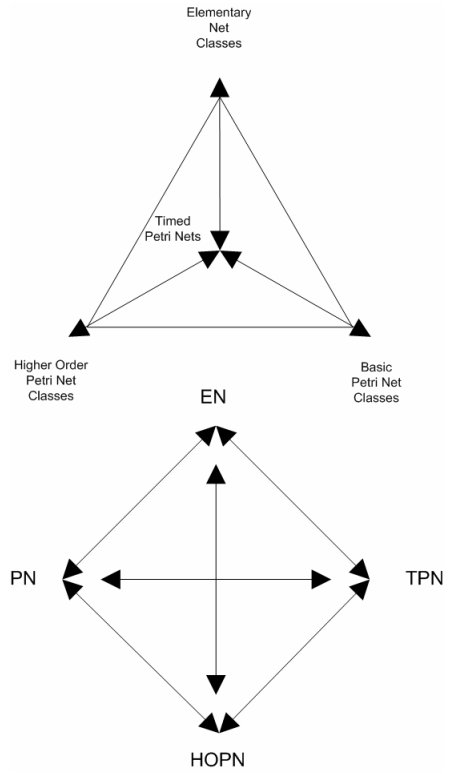

Fig. 1 Petri Net Class Transformation Relationships

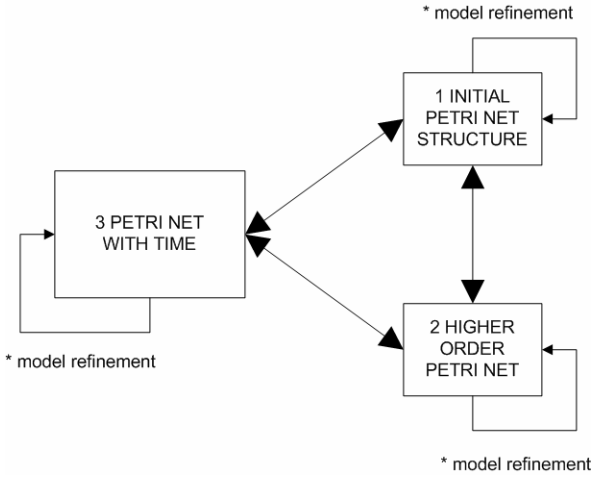

Fig. 2 Semi Structured Petri Net Class Transformations

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 Net Classes

These are a fundamentally simplistic class of Petri nets. Normally in ENs controlled changes take place via events that must be identical in another context. It is possible to obtain EN structures by 'reducing' other types of Petri nets. In this case information is lost or removed. For EN systems the state space is quite small. Behavior is always predictable.

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. It is still quite simple to convert this class into elementary nets via reduction. 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 Net Classes

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

Transition nets (Prt), Product nets, environmental nets, object oriented Petri nets, colored Petri nets, etc [10]-[11]. Higher order structures are characterized mainly by token types that can represent anything like an object, record, data sets, complex data types, etc. Place token types are complex. 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 Net Classes

The fourth category represents timed Petri nets (TPNs) [12]. There are many different sub categories of TPNs, but basically they are one of the three classes previously presented including the time dimension. As explained, more than a category in its own right this class derives from the previous ones. 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 timed Petri nets (TPNs), 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. Some sub classes of TPNs can become quite complex and detailed and require special expertise being directed to special areas.

IV. POSSIBLE 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 necessary 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.

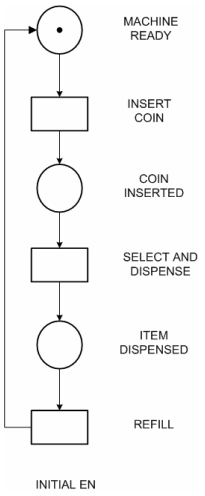

Fig. 3 Vending Machine Elementary Net

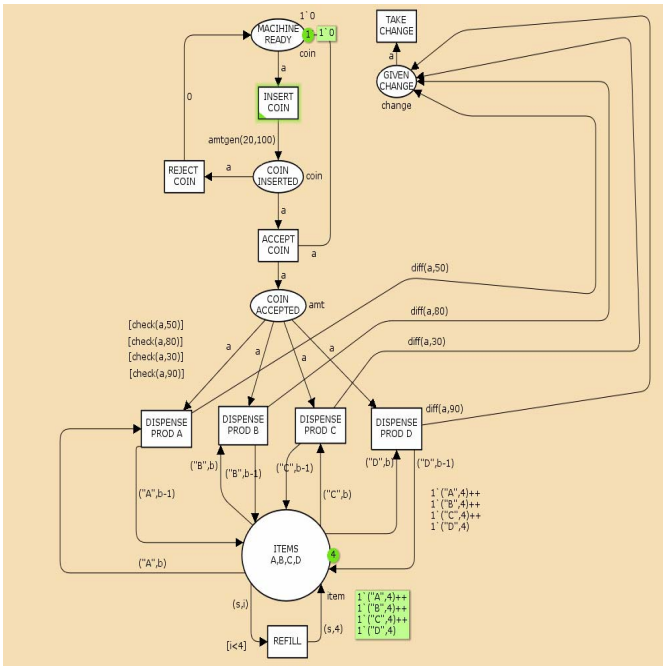

Fig. 5 Vending Machine Colored Petri Net

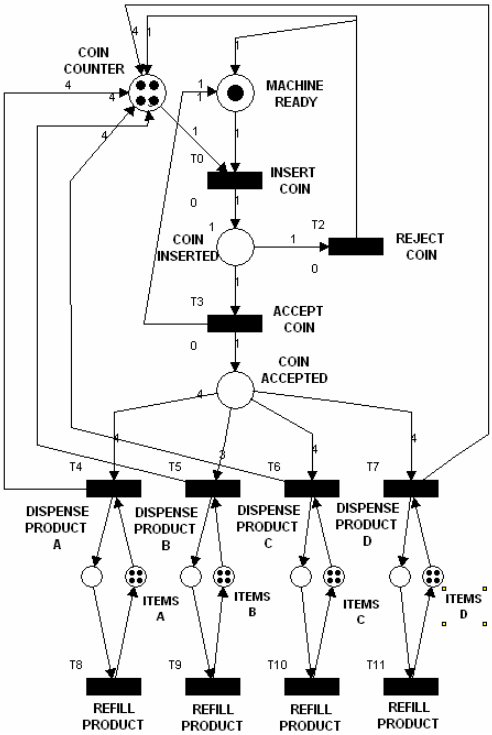

Fig. 4 Vending Machine General Petri Net

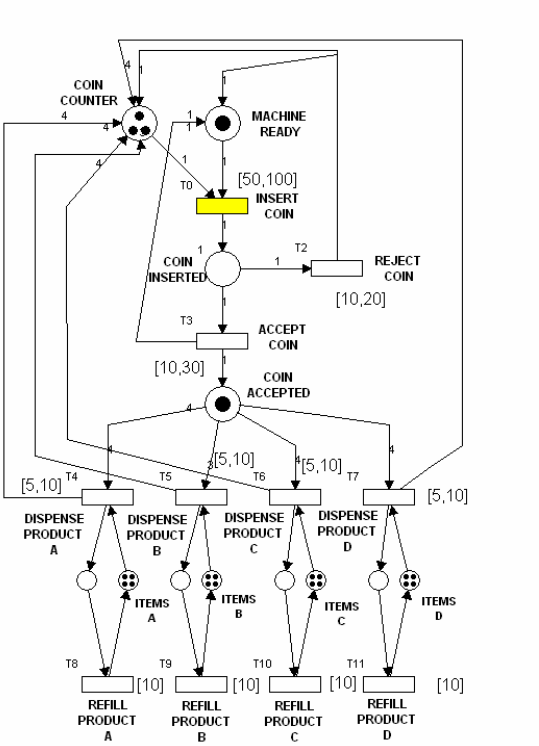

Fig. 6 Vending Machine Timed Petri Net

This structure is used as the starting point for constructing the basic Petri net, higher order nets and/ or the timed Petri net. It serves as the starting point for creating more detailed models for further investigation and analysis. It is possible to go into detail and draw the elementary net aesthetically better. The node and edge layout can be improved.

B. 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.

C. 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.

D. 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. As shown 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. The only changes are the transition types. The immediate transitions in fig. 4. are 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. Table 4 summarizes the use of these Petri net classes. 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. Hence it is easier to analyze EN and general PNs for static properties. It is possible to prove the correctness of these models using place/transition invariants and input/output incidence matrix. Table 3 summarizes the basic properties for the nets in fig. 3 and 4. These are the elementary net and general Petri net. These basic properties can be used to analyze these Petri nets. E.g. It is possible from the reachability analysis to construct a marking graph. Boundedness refers to the places being bounded. Liveness would indicate that the net transitions are enabled for firing in a given sequence. Reversibility indicates that the net can return to its original state or initial marking. In addition to these properties there are things like siphons and traps, etc. All the nets can be analyzed using simulation. 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.

A timed Petri net like the one in fig. 6 is useful for obtaining the cycle time of the system. This can be used to understand if there is a possible bottleneck with timing of different activities. This could then be resolved by changing the timings. Optimization problems begin with crucial understanding of the problem domain and the functioning of a system. Such systems normally are constraint bound related to capacity or time. Timed Petri nets are very useful for investigating this case. Normally optimization problems can

TABLE 3
STRUCTURAL QUALITIES

PN CLASS	Structural Qualities
EN,PN	Self loops Reachability Boundedness and Safeness Conservativeness Liveness Reversibility Home States

be formulated mathematically. Timed CPNs and certain classes of TPNs can be used for investigating certain types of problems. Systems operating at below optimal levels can incur certain penalties.

The Petri nets are suitable for visualizing the systems. It is possible to transform the Petri nets into directed graphs for other forms of analysis. The Petri nets could also be enhanced by using other formal languages. Other forms of analysis are possible like probability estimation of transition firing, discrete event simulation

Unfortunately the graphical complexity of the Petri nets will grow with the system complexity making them more difficult to depict. Validation and verification are not so simple when the Petri nets are complex. The models can be simplified using Petri net reduction methods based on rules for fusion, augmentation or elimination of places and transitions.

TABLE 4
PETRI NET CLASS SUMMARY OF USES

PN CLASS LEVEL	Complexity	Possible Results	System/software Development uses
EN	LOW	limited	initial
PN	MEDIUM	limited	middle
CPN	HIGH	detailed	detailed design
TPN	DEPENDS	detailed	depends

VII. CONCLUSION

This paper has very briefly explained the main Petri net classes and their uses in systems engineering. It has been shown how all the different classes of Petri nets fit together and can be combined for different purposes. It has been 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.

Petri nets are decomposable in a top down approach. This implies that the models especially the more complex ones can be refined in detail. It is possible to extract functioning of the individual system components.

The idea presented is suitable for using different Petri net classes for requirements engineering in general. One can start off with simple models and work up to new advanced and complex models as needed. It has been shown that all Petri net main classes find their use for modeling systems and software artifacts at different stages. Petri nets model dynamic behavior which is essential for proper comprehension.

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. It is possible to formalize the transformations. This requires a lot of new work, definitions and rules. It can become quite complex.

This was done as the aim of this paper is to present the concept of combining Petri net main classes. The fact of having several Petri net main classes and several sub classes indicates that they are quite flexible for use. 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. Many different case tools exist for supporting Petri net construction and simulations.

Some advantages of using Petri net classes in requirements engineering might be: i) improved system/software features, ii) reduced rework at construction stage, iii) final product has fewer defects, iv) better stakeholder negotiation of the product at the initial stages a priori to the design, v) accurate final system performance and timing ,vi) good requirements validation. Disadvantages of using Petri nets are that normally more work is required, special expert knowledge is needed and it is difficult to find proper case tools that integrate all different Petri net classes. The possible benefits from using Petri nets depend on many other factors and issues, like quality ,good project management, expert knowledge, good team support, etc. which are not always easy to find.

REFERENCES

[1] S.R. Schach, *Introduction to Object-Oriented Analysis and Design with UML and the Unified Process*. NY: McGraw-Hill, 2004, ch.1-11.

[2] J. Brusey, D. McFarlane, "Designing Communication Protocols for Holonic Control Devices using Elementary Nets*", Holonic and Multi-Agent Systems for Manufacturing*, 2nd Int. Conf. on Industrial Applications of Holonic and Multi-Agent Systems, Aug 2005, pp.76-78.

[3] L. A. Cortes, P. Eles, Z. Peng , "A Petri Net Based Model for Heterogenous Embedded Systems*", Proc. Norchip Conf*., Oslo, Norway, Nov 1999, pp. 248-255.

[4]] K. Jensen, G. Rosenberg, *High-Level Petri Nets: Theory and Application* , Springer – Verlag, Berlin, 1991.

[5] L.M. Kristensen, S. Christensen, K. Jensen, "The Practioner's Guide to Coloured Petri Nets*", International Journal On Software Tools for Tech. Transfer (STTT)*, Vol. 2, Springer-Verlag,1998, pp. 98-132.

[6] K. Jensen, L. M. Kristensen, L. Wells, "Colored Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems", *International Journal On Software Tools for Tech. Transfer (STTT),* Springer- Verlag, Vol. 9, Springer-Verlag, 2007, pp. 213-254.

[7] CPNTools, CPN Group, Department of Computer Science, University of Aarhus, Denmark. http://www.daimi.au.dk/CPnets/

[8] T. Spiteri Staines, " Using a Timed Petri Net (TPN) to Model a Bank ATM ", *Proc. of 13th IEEE Int. conf. on Engineering of Computer Based Systems,*Potsdam Germany,2006, pp. 151-158.

[9] A. Spiteri Staines, "Modeling UML Software Design Patterns Using Fundamental Modeling Concepts (FMC)", *Proceedings of the 2nd WSEAS European Computing Conference*, Malta, pp. 192-197, Sep 2008.

[10] *A Classification of Petri Nets,* http://www.informatik.uni-hamburg.de/TGI/PetriNets/Classification

[11] L. Bernardinello, F. De Cindio, A Survey of Basic Petri Net Models and Modular Net Classes, *LNCS Vol. 609*, Springer-Verlag, 1992.

[12] M. Zhou, K. Venkatesh, *Modeling, Simulation And Control Of Flexible Manufacturing Systems A Petri Net Approach*, World Scientific, 1999, isbn981023029X.

[13] I. Graham, *Object-Oriented Methods Principles & Practice*. ED: Pearson Education, 2001, ch.6.

[14] S. Goldsmith, *Real-Time Systems Development*, Pretence Hall, 1993, ch. 1.

[15] J. Desel, W. Reisig, Place/Transition Petri Nets, *LNCS*, Vol 1491\1998, Springer-Verlag, 1998.

[16] A. Knöpfel, B. Gröne, P. Tabeling, *Fundamental Modeling Concepts*, Wiley UK ,2006.

[17] P. Strbac, M. Tuba, D. Simian, " Hierarchial model of a systolic array for solving differential equations implemented as an upgraded Petri net", *WSEAS Transactions on Systems*, Vol. 8. Issue 1, pp.12-21, Jan 2009.

[18] K. Mun Ng, Z.A. Haron, " Visual Microcontroller Programming using Extended S-System Petri Nets ", *WSEAS Transactions on Computers*, Vol 9. Issue 6, pp. 573-582, Jul 2010.

[19] H. Apaydin Ozkar, A. Aybar, " A reversibility enforcement approach for Petri nets using invariants", *WSEAS Transactions on Systems*, Vol. 7. Issue 6, pp. 672-681, Jun 2008.

[20] M.A. Drighiciu, A. Petrisor, M. Popescu, "A Petri Nets approach for hybrid system modeling", *NAUN Int. Journal of Circuits, Systems and Signal Processing*, Issue 2 vol 3, 2009.

[21] A. Spiteri Staines, " A Compact CPN Representation for Embedded and Control Systems Fault Diagnosis and Recovery", *Proc. of the 8th Wseas Int. Conf. on SEPADS*, Cambridge, pp. 78-83, Feb 2009.

A Petri net, also known as a place/transition (PT) net, is one of several mathematical modeling languages for the description of distributed systems. It is a class of discrete event dynamic system. A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, represented by bars) and places (i.e. conditions, represented by circles). The directed arcs describe which places are pre- and/or postconditions for which transitions (signified by arrows). Some Supporting Requirements Engineering with Different Petri Net Classes. Jan 2010. A Staines. Staines, A., 2010 Supporting Requirements Engineering with Different Petri Net Classes" International journal of computers, vol.4 (4). Internet Host Reliability Modeling with Time Petri Nets. Article. Jun 2012.Â In this paper we consider a class of Time Petri nets defined by structural restrictions. Each Time Petri net which belongs to this class has the property that their liveness behaviour does not depend on the time. Therefore, the Time Petri net is live when its skeleton is live. View. Show abstract.