Towards a Unified View of Modeling and Design with GHENeSys

Arianna Olivera Salmon, arianna@usp.br
José Armando San Pedro Miralles, jaspm2004@usp.com
Dept. of Mechatronics Engineering, University of São Paulo, Brazil
Pedro Manuel Gonzalez del Foyo, pedrogdelfoyo@gmail.com
Dept. of Mechanical Eng., Federal University of Pernambuco, Brazil
José Reinaldo Silva, reinaldo@usp.br
Dept. of Mechatronics Engineering, University of São Paulo, Brazil

Abstract. *It is almost consensual that modeling and design should converge to a distinguish phase in the project after specifications are well defined and a step further in the visualization of the artifact should irrupt in the scenario. For automated distributed discrete systems it is mandatory that this phase should describe how parts would harmonically contribute to the final goals.*
However, representing, documenting and also combining all these expectations with a pragmatic description that is also prepared to face verification is very challenging. Schemas are very well suited to do that, besides the fact that, due to its high generality, they are not particularly handful as a design approach. The good properties of schemas, that is, their lack of direct semantic is interesting to deal with incomplete knowledge for one side, but is also a problem to the need to preserve the knowledge acquired during requirements elicitation and analysis.
Object oriented Petri Nets was pointed as a good answer to that dilemma but was abandoned because there was not a broadly accepted formalism to objects that could be combined with the net system. This problem is revisited in this work using a different approach to nets that is absolutely inserted in the recent standard of IEC 15909, while using a model driven approach to object-oriented design. A classical case study is discussed concerning the problem of cats and mouse's in an automated environment where invariant analysis is considered.

Keywords: *schema, Petri nets, requirements analysis, formal verification*

1. INTRODUCTION

The last twenty years of last century was marked by the change from years of structured refinement approach to software and systems to schemas based on modularity. Quickly the most used approach evolve to be model driven, with the appearance of the model-view-control approach, used in Smaltalk. However, the strong separation between data transformers (models), its interface of activation (view) and a solver dispatcher (controller), even if very useful in new interactive software applications did not show the same appeal to system design, specially because it did not reinforce structure and refinement.

Thus, it is quite expectable that the end of the last century meet a new different approach: the Object Management Architect, created by OMG (Object Management Group) in 1990. This approach is basically object-oriented, meaning that it reinforces more encapsulation and the roles of data modifier and controller. It also dispense a different representation for interface connections, composing a very regular and neat approach. That was the base for representations such as UML, a great success in software development. However, in what concerns systems design the refinement oriented development was missed, and a generalization of UML to the design of any system would demand a revision in that subject.

Model Driven Architecture (MDA) is a new approach and candidate to substitute OMA in this years. It is based in four different layers: a meta meta model, a meta model, the model of the artifact and a instantiation of this model(Bezivin and Gerbe, 2001). UML itself could be located at the level of meta model, together with other model representations such as CWM (Common Werehouse Metadata), Wfl (Workflow) or even OCL (Object Constraint Language). It is unnecessary to stress how useful those parallel representation could be to systems design, specially to automated systems design. However, for a stepwise refinement of large (distributed) systems a good validation cycle is important, and even following MDA, models must be validated by different users and stakeholders before going further in the development.

This is an important point, which also brings a question on what should be a good model representation to validate (or verify formally, is possible) a system and be also a good reference and documentation to the development itself. Petri Nets are a good candidate for that and in this work we will try to justify the convenience of having a schema that could reproduce the same semantic of a Conceptual Graph for one side and dispose of graph formalism that allow verification and a capacity to simulate state-transition evolution. It could also be useful to represent systems with time constraints, even time slice or dense time evolution.

Besides, since such representation is today a subject for a standardization project, launched to support Place/Transition nets, colored nets and now to cover all extensions, it could be also important that have a Petri Net environment that could also capture extensions and time constraints, while maintain a complementary representation of classic and high level nets. That would mean changing from classic to high level or extended representation to fit design needs.

In what follows it will be described a general environment called GHENeSys (General Hierarchical Enhanced Net System) with all the described properties, that is, that was constructed to capture the features of extensions, without loosing the formal approach and property analysis and that also fit the new Petri Net Standard.

2. PETRI NETS STANDARD

Since the beginning of this century two major tendencies in the state-of-art of Petri Nets emerged: one is the confirmation of Petri Nets as a sound formalism to model several discrete systems in different areas of knowledge, since biological and chemical systems up to the control systems applied to manufacturing and automated process in general; another tendency is the introduction of a unified approach to Petri Nets, giving another perspective to the profusion of particular approaches and extensions. Even high-level nets are now envisaged as a general approach that subsumes elementary nets through the Basic High-Level Nets.

The PN standard can be seen as the greater effort of the PN community to unify the current status and the future of the research and development efforts in the area. The principal aim of the PN standard is to define the characteristics that allows a net proposal to be considered a Petri Net. This standardization effort converge into the ISO/IEC 15909 project and which has been designed in three independent parts in order to enable flexibility of the process.

In the next section 3,a proposal for an enhanced PN called GHENeSys will be presented. This proposal is evolving to meet the PN standard specifications, including extensions of the classic and high level model, which are being discussed now in the committee of the 15909 project. Before that, the next sub-sections will be used to point out briefly the steps taken in this project, to stress the ones that are still under discussion.

2.1 Part I: 15909-1

The first part of the standard was published as an International Standard (IS) in December 2004. It provides a comprehensive documentation of the terminology, the semantical model and the graphical notations for High-level Petri nets. It also describes different conformance levels. Finally, a tutorial example given in annex illustrates the different concepts in the standard. Technically, part 1 provides the mathematical definitions of High-level Petri Nets, called the semantic model, the graphical form of the technique, known as High-level Petri Net Graphs (HLPNGs), and its mapping to the semantic model. Part 1 also introduces some common notational conventions for HLPNGs (ISO/IEC, 2002).

In Miralles and Silva (2010) the semantical model and the graphical notations of GHENeSys where revisited to attend the specifications of the 1st part of the standard. GHENeSys definition is now being modified to allow the representation of HLPNs.

2.2 Part II: 15909-2

Originally, the Petri Net Markup Language(PNML) was introduced as an interchange format for all kinds of Petri nets (Bastide *et al.*, 2000; Billington *et al.*, 2003; Kindler and Weber, 2001). Some major concepts of PNML were driven by this objective. Part 2 of this international standard, that was published as an IS in February 2011, defines a transfer format in order to support the exchange of High-level Petri Nets among different tools (ISO/IEC, 2005). Technically, this part of the standard defines a transfer syntax for High-level Petri Net Graphs and those subclasses of Petri nets only that have been conceptually and mathematically defined in the first part of the standard, for capturing the essence of all kinds of coloured and high-level Petri nets.

In order to deal with all kinds of Petri nets, the transfer format needs to be flexible and extensible. In order to obtain this flexibility, a Petri net is considered to be a labeled directed graph, where all type specific information of the net is represented in labels. A label may be associated with a node, an arc, or the net itself. This basic structure of a PNML Document is defined in the PNML Core Model using a UML class diagram. Based on this PNML core model, part 2 also defines the transfer syntax for the three versions of Petri Nets that are defined in part 1 of this IS: Place/Transition Nets, High-level Petri Nets and Symmetric Nets.

Technically, the PNML Core Model is a UML package, and there are additional UML packages for different Petri net types that extend this PNML Core Model package. ISO/IEC 15909-2 defines a package for Place/Transition Nets, a package for Symmetric Nets, and a package for High-level Petri Net Graphs, where the package for High-level Petri Net Graphs extends the package for Symmetric Nets. Therefore, every Symmetric Net is also a High-level Petri Net Graph. Figure 1 gives an overview of the different packages defined and their dependencies, (Hillah *et al.*, 2010).

To make the this part of standard applicable and provide a reference implementation to Petri net tools developers, was developed the PNML Framework. PNML Framework is a generated set of comprehensive and easy to use tailored application programming interface (API) to import and export Petri net models designed according to the standard specifications. It is intended to be used as a library, therefore it can be easily integrated into Petri net tools. Tools developers are considered as the primary users of PNML Framework. Thanks to this framework, tools developers would rather focus on their applications core development instead of coping with how to stay up-to-date and compliant with the standard.

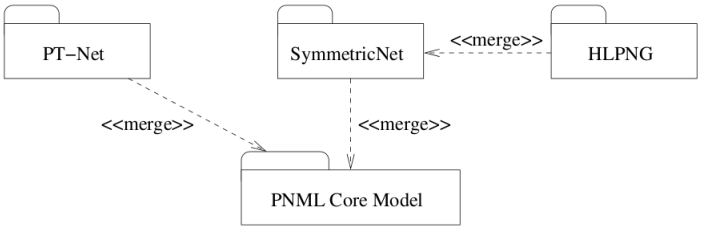


Figure 1. Overview of the UML packages of PNML, (Hillah *et al.*, 2010)

Furthermore, they would not have to deal with ensuring continuous compatibility with other tools and many Petri net types and variants.

The application tool based on GHENeSys proposal is in the list of tools that uses this framework to support the PNML, this list have less than 10 members, (Pnml.org, 2011).

2.3 Part III: 15909-3

Part 3 is devoted to the standardization of Petri nets extensions, including hierarchies, time and stochastic features. These extensions will be built upon extensions of the core model. They require a stable version of the core model to be available. According to the standardization schedule provided in Petrucci (2007), the working draft of the 3rd part should be ready in 2009, but there are not any available reference of this document. In Hillah *et al.* (2006) it was suggested that to make a solid work on the 3rd part, a stable version of the standard *core model*, part 1 and 2, must be available. Since part 2 was only published as an IS a few months ago, the authors believe that the main work concerning part 3 must be starting right now.

In the next section a proposal of an enhanced PN will be presented. This proposal it is called GHENeSys and capture several extensions that are within the scope of this part of the standard. Due to the fact that the 3rd part is not ready yet, the authors considers that this extensions could be analyzed and included at least partially in this part of the standard, since, as they were defined this extensions preserve the algebraic representation of the net (Miralles and Silva, 2010). Therefore, algorithms can be implemented to verify structural and behavioral properties of net models made with GHENeSys.

Concluding, the standardization process is quite long and relatively complex. A standard must be built in order to be stable enough to be used by the people involved. It is developed within a schedule which should not exceed three years and it is subject to revision every five years (more information on the rules can be found in ISO (2011)). Nevertheless, the advantages that brings this standardization process are more than welcome by the PN community and worth all the effort, once, as a result of this process, all research and development efforts will soon converge to an unified and strong path. Part 1 of this IS already establishes what needs a net proposal must satisfy to be considered as a PN; part 2 provides an unique, portable and easy extensible transfer format allowing the interchange between different PN based modeling tools; and last but not least, part 3 will unify all the existing extensions and will state formal mechanisms to generate new extensions in the future. Thus, more people from/outside the PN community will increase or start to use PN within the design process of systems in multiple knowledge domains.

3. GHENESYS: AN ENHANCED PETRI NET FACILITY

The net environment GHENeSys was designed and developed in DLab starting from ideas presented in Miyagi (1988); Silva and Miyagi (1996), and finally formalized in del Foyo (2001). Initially, it was conceived as an extended net with concepts of object-orientation, a mechanism of abstraction - through the definition of hierarchy -, and synthesis, based on a structured approach. GHENeSys has a great potential to become a tool capable to represent, in a unified way, classic PN and all their extensions, as well as a high level PN. From the definition of GHENeSys four meta modes could be derived, which cover a wide range of systems: centralized, hierarchical, hierarchical modified and distributed or multi-agent, and timed control systems.

Basic elements are:

- **Pseudoboxes:** this elements allows to model the transfer of information between different parts of the system, providing a mechanism to obtain a modular structure that improve the process of net synthesis. These elements also allows to represent the interaction with the surroundings of the system, since there is no isolated systems in nature, del Foyo (2001). Finally, they are also used to represent information of observable, predictable but not controllable events (Lin and Wonham, 1990; Ramos and Silva, 1998).

- **Hierarchy:** elements called *macro* are abstractions in GHENeSys and can encapsulate sub-nets. The *macro* elements definition allows to preserve (by construction) structural and behavioral properties of the “global” net,

making possible the verification of global properties guarantying that the behavior of sub-nets will have no influence on these properties (except by adding or multiplying by a constant). Thus, the net can be constructed mixing simple and *macro* elements within different levels of abstraction. This aggregational advantage is aligned with the evolution of modeling and design processes, where elements defined in abstract level get alive with others defined in more detail, and can eventually be reused in other projects.

- **Object-oriented approach:** in del Foyo (2001) net elements are defined as "active" objects, and tokens are defined as "passive" objects (without methods). Therefore, a net is formed by the aggregation of component objects, and it is also an object. Sub-nets, represented using *macro* elements, with widely known structures are frequently used in models (like buffers FIFO, LIFO) and can be replaced by methods. Thus, it is possible to reuse such structures in others models. This allows to face the problem of state explosion, a common issue when dealing with real life systems modeling. In the current development stage, GHENeSys definition is ready and capable to deal with HLPN models in similarly to CPN Tools, but this feature is not fully implemented yet in GHENeSys graphic editor. Like in CPN Tools (Jensen *et al.*, 2007), GHENeSys modeling tool uses Standard ML (SML) (Harper, 2010), to make declarations of token types and attributes; and also arc's inscriptions and transition's filters.

- **Temporal approach:** in previous versions of *GHENeSys* evolution del Foyo (2001), treated time slice as in Ramchandani (1974), allowing to represent processes with deterministic durations only. Thus, processes duration time was represented as a waiting time before the firing of transitions. Taking into consideration that most processes are not deterministic, as in real-time systems, such approach was modified in del Foyo (2010). The current temporal approach in *GHENeSys* admit the representation of non deterministic durations (Merlin and Faber, 1976). In Merlin's approach, there are a set of time intervals with minimum and maximum limits for the fire of every transition. This approach maintain the capabilities of the net to have multiple behaviors making some assumptions that proves its equivalence:
  - Assuming a time interval like $[tMin, tMax]$ when $tMin$ = minimum time limit and $tMax$ = maximum time limit for the firing of every transitions.
  - To model a timed system using Ramchandani's approach, just make $tMin = tMax$
  - To model a non-timed system, just make $[tMin, tMax] = [0, \infty]$

All this features are integrated in GHENeSys environment, which is now in its final steps to have a version that will be distributed under a GNU General Public License. This version includes a set of analysis features that will help users to extract useful informations from their models. These features are:

- **Incidence Matrix generation:** this feature allows to generate the Incidence Matrix behind the net model. When dealing whit hierarchical models, the algorithm take this into account and is able to generate the "global" net matrix and also the sub-nets matrices separately. This feature constitutes the foundation that supports the rest of the analysis features.

- **Invariants analysis:** in Clarisó *et al.* (2005) and many others works it is shown that invariants can be used to prove safety properties of the Petri Net, like boundedness and deadlock freedom. Thus, it is a great advantage to have algorithms that extract place and transition invariants from PN models. GHENeSys is capable to generate the set of place and transition invariants from PN models as will be shown in more detail in the next section. The user will be able to see and save the set of invariants from PN models.

- **Models Simulation:** this feature allows the simulation of all PN models to validate behaviors. This is possible because GHENeSys definition use an unified approach within the modeling process. The algorithms implemented for the simulator were based on the generation of the state class (del Foyo, 2010) which is the first step to make time analysis. Even if it is not in the scope of the present work we claim that this algorithm is equivalent to the on going generation of states from the state equation. This it is possible since the set of assumptions exposed in above in this section are respected .

However, since the environment can always generate the incidence matrix, it is also possible to find the state equation, as a step to analyze properties of the nets. The invariant analysis is described in the following.

4. INVARIANT ANALYSIS IN PETRI NET AND GHENESYS

Invariants are fundamental algebraic characteristics of Petri nets and are used in various situations such as checking (necessary condition of) liveness, boundedness, the presence of loops and so on (Murata, 1989). Generally, invariants represent repetitive and conservative components of a net, that is, sets of places and transitions which behavior do not

change during execution. The identification and interpretation of each of these sets is important because they reflect certain properties of the net that may be of interest to the modeling process.

Place and transition invariants are important means for analyzing Petri nets since they allow for the net's structure to be investigated independently of any dynamic process (Lautenbach, 1987). Another advantage of the invariants is that its analysis can be performed on local subnets without considering the whole system. Invariants are also used to model validation and verification, what can be performed while analyzing other properties of the net.

4.1 Place invariants

Place invariants are sets of places whose token count remains always constant.These invariants represent the conservative component of the net. They are represented by an n-column vector x, where n is the number of places of the Petri net. The non-zero entries correspond to the places that belong to the particular place invariant and the zeros to the remaining places. Place invariants are a positive integer solutions of the homogeneous equation:

$$A.x = 0 \tag{1}$$

Considering the state equation of a net system, that means to have an integer vector solution x that satisfies the equation:

$$M^T.x = M_0^T.x \tag{2}$$

where M_0 is an initial marking and M belongs to $R(M_0)$.

Equation (2) means that the possibly weighted sum of the tokens in the places of the invariant remains constant in all markings and this sum is determined by the initial marking of the Petri net.

4.2 Transition invariants

Transition invariants denote a sequence of transitions which firing can reproduce the initial marking in the sequence. These invariants represent the repetitive components of the net and are represented by an m-column vector *y* that contains integers in the positions corresponding to the transitions belonging to the transition invariant and zeros everywhere else. The integers denote how many times the corresponding transition must fire in order go back to the initial marking. They can be derived from the state equation as:

$$A^T.y = 0 \tag{3}$$

As with place invariants, any linear combination of transition invariants is also a transition invariant for the Petri net. The existence of transition invariants in the Petri net denotes a cyclic behavior.

4.3 On the structural analysis of Petri nets

Two formal presentations have extensively been used for structural analysis: *graph theory* and *linear algebra*. Clearly, the main goal is to develop techniques that can be easily implemented on a computer. This has motivated the development of methods and techniques to optimize the analysis of the structural properties of petri nets. Techniques that have been considered efficient are those based on linear-algebra, because of their simplicity to obtaining invariants as a initial step for studying the structural properties of Petri nets.

For instance, many works have been developed to analyze properties of Petri nets using Linear Programing techniques. One of the main advantages of using this method is that the computational complexity of linear programming problems is polynomial. Usually, the solution is based on the simplex algorithm, which among other advantages, performs most often in linear time even though its worst case which is exponential (Silva and Colom, 1988), (Colom *et al.*, 1990), (Silva *et al.*, 1998). Silva *et al.* (1998) provides an analysis of important safety properties of net systems (boundedness, mutual exclusion, deadlock-freeness, etc.) using the state equation. This method is at the same time more efficient and accurate than the classical invariant method, which on the other hand is easier to understand. A limitation of the state equation method is that it is best suited to analyze safety properties, i.e., the existence of markings (and firing vectors). Owing to this, the analysis allows only to decide partially about the corresponding properties, i.e., to find only necessary or sufficient conditions.

There are other works, such as (Bouyekhf and el Moudni, 2005), that considers some structural aspects of general Petri nets and tries to improve the link between Petri nets and linear algebra techniques. On the other hand, using linear programming techniques do not guarantee that real invariants can be obtained. That is why many researchers have combined methods (WAKUDA *et al.*, 1999), such as the combination of the Fourier-Motzkin method, which is based on linear programming, with algorithms for obtaining minimal siphon (FDMS) (Tanimoto *et al.*, 1996).

In this workf invariants were used for the analysis of some structural properties of Petri nets, as well as for the verification of system requirements. The invariant analysis is very similar to the use of invariants in program verification: the designer must find a set of equations that denotes the desired properties and test if they hold in any reachable state. This method can be inserted in the GHENeSys environment as part of the modeling and design approach to discrete systems.

4.4 Calculation of the invariants in GHENeSys

The invariant computing can be reduced to the resolution of the homogeneous system 1 and 3 . In this work we propose a Gauss-Jordan Elimination with pivoting method (Peters and Wilkinson, 1975),(Yalamov, 1991) to transform the coefficient matrix into a diagonal matrix and then obtain the invariants. For the matrix m x n, the method has a complexity of $O(mn^2)$. After obtaining the diagonal matrix, we propose a method to obtain the set of vectors that represent the basic solution. This means that any linear combination of these vectors is also a solution of the homogeneous system.

Steps to calculate invariants
Transition invariants

- Step 1: Calculate the incidence matrix (already a feature of the environment).
- Step 2: Apply the Gauss-Jordan method to the incidence matrix. If the rank of the incidence matrix is less than the number of unknowns, then go to step 3. Else, if the rank of the incidence matrix is equal to the number of unknowns then finish, the system has no transition invariants.
- Step 3: Get the basic solution of the system representing for the incidence matrix.

Place invariants

- Step 1: Calculate the transpose of the incidence matrix.
- Step 2: Apply Gauss Jordan elimination method. If the rank of the incidence matrix is less than the number of unknowns, then go to step 3. Else, if the rank of the incidence matrix is equal to the number of unknowns then finish, the system has no place invariants.
- Step 3: Get the basic solution of the system represented by the incidence matrix.

Reviewing Step 3: Method to get the basic solution from the system of homogeneous linear equations:

Let be the system of linear equations: $Ax = 0$, where A is an m x n matrix (m equations and n unknowns), and let k be the rank of the matrix . If the solution space of the system is S, then $dim(S) = n - rank(A) = n - k$ [1]. To find a basis for S, it is necessary to find $n - k$ vectors that generate the solution space S. Applying Gauss Jordan elimination method the incidence matrix can be transformed into a diagonal matrix (step 1). After that it is necessary to clear the linearly independent variables.

Supposing that the independent variables are: $x_{k+1}, x_{k+2}, \ldots, x_n$, then there are $n - k$ independent variables. For each set of values assigned to independent variables, a set of values of dependent variables are obtained. To obtain the $n - k$ desired vectors, each of the independent variables are assigned with the following values:

$$x_{k+1} = 1, x_{k+2}, x_{k+3} = 0, \ldots x_n = 0 \tag{4}$$

$$x_{k+1} = 0, x_{k+2} = 1, x_{k+3} = 0, \ldots x_n = 0 \tag{5}$$

$$x_{k+1} = 0, x_{k+2} = 0, x_{k+3} = 1, \ldots x_n = 0 \tag{6}$$

. . . . . .

$$x_{k+1} = 0, x_{k+2} = 0, x_{k+3} = 0, \ldots x_n = 1 \tag{7}$$

Thus it is possible to obtain $n - k$ vectors of S to form a base.

Table 1. Algebraic conditions for structural properties

Structural Properties	Necessary and Sufficient Conditions
Structural Boundedness	$\exists x > 0, A^T x \geq 0$
Conservativeness	$\exists x > 0, A^T x = 0$
Repetitiviness	$\exists y > 0, Ay \geq 0$
Consistency	$\exists y > 0, Ay = 0$

4.5 Analyzing other structural properties that can be analyzed using the invariants

Tab. 1 shows some structural properties that can be analyzed using the invariants.

Analyzing Table 1 it is possible to conclude that place invariants are crucial to analyze Structural Boundedness and Conservativeness. Transition invariants are a step to analyze Repetitiveness and Consistency. Each solution corresponding to a homogeneous system of linear equations shows the way the system can be traversed starting from the home state (Daniel Lamch 02).

The study of the transition invariants also allows to analyze the liveness property. For checking the liveness property, the solution of transition invariants must be searched. This feature consists in adding up vectors corresponding to solutions of a homogeneous system of linear equations. If all elements of the resulting vector are greater than zero, the system is free from deadlock states.

GHENeSys supports a state equation which is formally defined in del Foyo (2001) which allows the analysis of the net properties, similar to the one used in classical net. Thus the algorithm described above was developed and inserted into the current version of GHENeSys, allowing the calculation of invariants as in the case study that follows.

5. A CASE STUDY USING INVARIANTS TO ANALYSZE REQUIREMENTS

The classical cat-and-mouse problem, was introduced in (Wonham and Ramadge, 1987),and is a popular example in the field of discrete event system control (Yamalidou *et al.*, 1996). This problem can be described as follows: A cat and a mouse are placed in a maze as shown in Fig. 2. In the initial state the cat is in the room 3 and the mouse is in the room 5. Each port on the maze can be traversed in the direction indicated, and is used exclusively by the cat or the mouse.

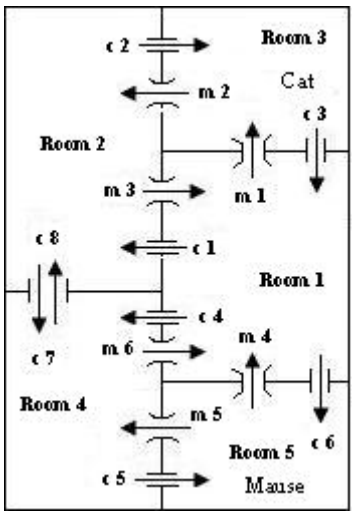


Figure 2. The cat and mouse maze.

Fig. 3 shows the modeling of the problem in GHENeSys, where the places represent the rooms and the transitions represent the doors. Each net has five sites representing the five rooms. The transitions represent the capacity of each animal to move from one port to another. There are one token in a place indicating that the respective animal modeled by the token is in the room modeled by the place. In the initial marking the cat is in room 3 and the mouse is in room 5. The incidence matrix of the GHENeSys model for the cat and mouse problem is shown in Fig. 4.

To calculate the *place invariants* it is necessary to get the transpose of the incidence matrix (A^T) shown in Fig. 4, and then apply the method of Gauss to obtain a diagonal matrix equivalent to the matrix (A^T) and also the rank of the

[1]Dimensions of space

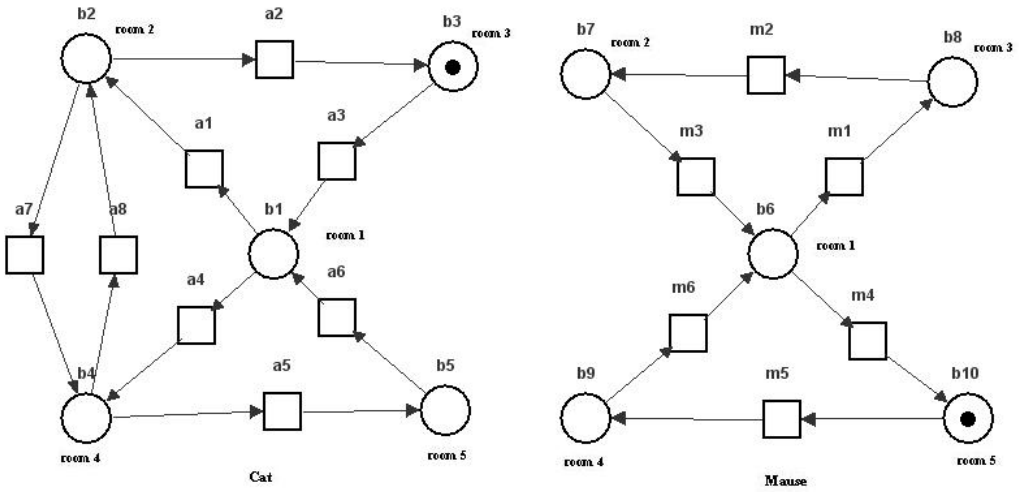
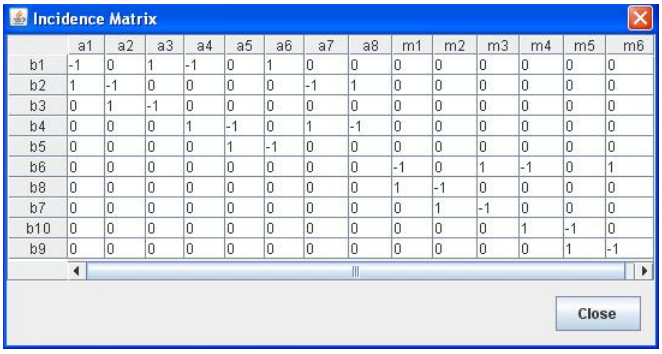


Figure 3. Modelling in GHENeSys the cat and mouse problem.



	a1	a2	a3	a4	a5	a6	a7	a8	m1	m2	m3	m4	m5	m6
b1	-1	0	1	-1	0	1	0	0	0	0	0	0	0	0
b2	1	-1	0	0	0	0	-1	1	0	0	0	0	0	0
b3	0	1	-1	0	0	0	0	0	0	0	0	0	0	0
b4	0	0	0	1	-1	0	1	-1	0	0	0	0	0	0
b5	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
b6	0	0	0	0	0	0	0	0	-1	0	1	-1	0	1
b8	0	0	0	0	0	0	0	0	1	-1	0	0	0	0
b7	0	0	0	0	0	0	0	0	0	1	-1	0	0	0
b10	0	0	0	0	0	0	0	0	0	0	0	1	-1	0
b9	0	0	0	0	0	0	0	0	0	0	0	0	1	-1

Figure 4. Incidence matrix of the cat and mouse problem obtained by GHENeSys system.

incidence matrix($R(A^T)$). As a result of this calculation is obtained that $R(A^T) = 8$. The $R(A^T)$ is smaller than the number of unknowns, and therefore the size of the basic solution is two. Place invariants obtained are shown in Fig. 5 (a).

With the place invariants obtained can we formulate the following equations:

$$M(b1) + M(b2) + M(b3) + M(b4) + M(b5) = 1 \tag{8}$$

$$M(b6) + M(b7) + M(b8) + M(b9) + M(b10) = 1 \tag{9}$$

Eq. (8) means that at any given time the cat only can be in one of these places: *b1, b2, b3, b4, b5* and Eq. (9) means that at any given time the mouse only can be in one these places: *b6, b7, b8, b9, b10*; i.e, for each pair of places, one from the cat and one from the mouse, associated with the same room, must never contain more than one token as a requirement of the problem.

Steps to calculate the invariants transition are almost the same as the steps to calculate the invariants place. The only difference is that calculating transition invariants take the incidence matrix shown in the Fig. 4, and not the transpose of the matrix as in the place invariant. Fig. 5 (b) shows the transition invariants obtained.

The results for the invariant analysis match the classical one found in the literature and the difference in using the proposed method is not just the speed of the process but the fact that with the proposed algorithm allows to generate the linearly independent basis of invariants an therefore the whole space.

6. CONCLUSION

Understanding that design is a more complete process that starts in the elicitation of requirements goes trough the analysis of these requirements, converge to a specification from which a model must be generated to allow further analysis, validation and requirement, the unification between design and modeling depends on the representation used. Such representation should be able to synthesize different "models" from a meta model prescription (or from different meta model prescriptions) and make it to converge in a unique general model which is analyzed to validate requirements. Then another model must be generated from the validated specs to bring what is conventionally called a "model" which properties should be the desired ones.

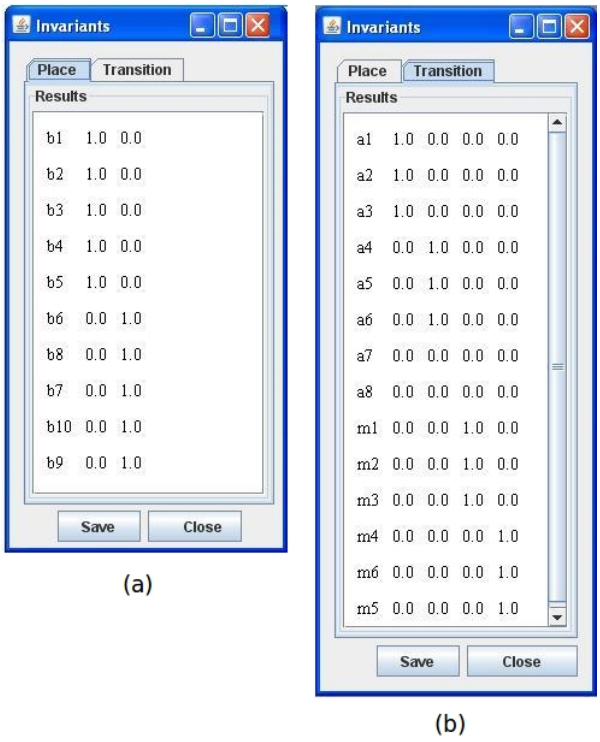


Figure 5. (a) Place invariants and (b) transition invariants obtained by the GHENeSys system.

In this work it implicit that Petri Nets would stand for the requirements model as well as to the "design model", and since the same representation is used it makes it dispensable the second modeling phase. Also the resulting model can be the basis for verification using Invariant or other property analysis, even including time. The intention is to use this platform environment to analyze all Petri Net extensions, including Object-oriented nets to contribute to the discussion of the ISO/IEC committee in what concerns the compatibility of these extensions with the net formalism.

7. ACKNOWLEDGEMENTS

Two of the authors, Arianna Z. S. Olivera and José Armando S-P Miralles are partially supported by CAPES and CNPq.

8. REFERENCES

Bezivin, J. and Gerbe, O., 2001. "Towards a precise definition of the OMG/MDA framework". In *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*. IEEE Comput. Soc, pp. 273–280. ISBN 0-7695-1426-X. doi:10.1109/ASE.2001.989813. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=989813`.

Billington, J., Christensen, S., van Hee, K., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C. and Weber, M., 2003. "The petri net markup language: Concepts, technology, and tools". In *Applications and Theory of Petri Nets 2003: 24th International Conference*. Eindhoven, The Netherlands, pp. 1023–1024. URL `http://www.springerlink.com/content/rp1dqtlmqr5q665b`.

Bouyekhf, R. and el Moudni, A., 2005. "On the analysis of some structural properties of petri nets". *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, Vol. 35, pp. 784– 794.

Clarisó, R., Rodríguez-carbonell, E. and Cortadella, J., 2005. "Derivation of nonstructural invariants of petri nets using abstract interpretation". In *Application and Theory of Petri Nets and Other Models of Concurrency: Proceedings of the 26th International Conference, volume 3536 of Lecture Notes in Computer Science*. SpringerVerlag, pp. 188–207.

Colom, J.M., Campos, J. and Silva, M., 1990. "On liveness analysis through linear algebraic techniques". In *In Proceedings of the Annual General Meeting of ESPRIT Basic Research Action 3148 Design Methods Based on Nets (DEMON*.

del Foyo, P.M.G., 2001. *GHENeSys: Uma Rede Estendida Orientada a Objetos para Projeto de Sistemas Discretos*. Master's thesis, Escola Politécnica da Universidade de São Paulo.

del Foyo, P.M.G., 2010. *Verificação Formal de Sistemas Discretos Distribuídos*. Ph.D. thesis, Escola Politécnica da Universidade de São Paulo.

Harper, R., 2010. *Programming in Standard ML - Working Draft*. Carnegie Mellon University.

Hillah, L., Kordon, F., Petrucci, L. and Trèves, N., 2006. "Pn standardisation: A survey." *Lecture Notes in Computer Science : Formal Techniques for Networked and Distributed Systems - FORTE 2006, Volume 4229, 2006*, pp. 307–322.

Hillah, L.M., Kordon, F., Petrucci, L. and Trèves, N., 2010. "Pnml framework: An extendable reference implementation of the petri net markup language". In *Petri Nets*. pp. 318–327.

ISO, 2011. "International harmonized stage codes." URL `http://www.iso.org/iso/en/widepages/stagetable.html`.

ISO/IEC, 2002. *High-level Petri Nets - Concepts, Definitions and Graphical Notation - International Standard Final Draft ISO/IEC 15909*.

ISO/IEC, 2005. *Software and Systems Engineering - High-level Petri Nets, Part 2: Transfer Format, International Standard WD ISO/IEC 15909*, wd version 0.9.0 edition.

Jensen, K., Kristensen, L.M. and Wells, L., 2007. "Coloured petri nets and cpn tools for modelling and validation of concurrent systems". *International Journal on Software Tools for Technology Transfer*, Vol. 9, No. 3-4, pp. 213–254. ISSN 1433-2779. doi:10.1007/s10009-007-0038-x. URL `http://www.springerlink.com/index/10.1007/s10009-007-0038-x`.

Kindler, E. and Weber, M., 2001. "The petri net kernel an infrastructure for building petri net tools". *International Journal*, pp. 486–497. doi:10.1007/s100090100055.

Lautenbach, K., 1987. "Linear algebraic techniques for place/transition nets". *Springer-Verlag*, pp. 142–167.

Lin, F. and Wonham, W., 1990. "Decentralized control and coordination of discrete-event systems with partial observation". *Automatic Control, IEEE Transactions on*, Vol. 35, No. 12, pp. 1330–1337. ISSN 0018-9286. doi: 10.1109/9.61009.

Merlin, P. and Faber, D., 1976. "Recoverability of communication protocols–implications of a theoretical study". *Communications, IEEE Transactions on [legacy, pre-1988]*, Vol. 24, No. 9, pp. 1036–1043.

Miralles, J.A.S.P. and Silva, J., 2010. "Enquadrando ghenesys dentro do petri net standard". In *Anais do XVIII Congresso Brasileiro de Automática*.

Miyagi, P.E., 1988. *Control System Design, Analysis and Implementation of Discrete Event Production Systems by using Mark Flow Graph*. Ph.D. thesis, Tokyo Institute of Technology, Tokyo.

Murata, T., 1989. "Petri nets: Properties, analysis and applications". *Proceedings of the IEEE*, Vol. 77, pp. 541–580.

Peters, G. and Wilkinson, J.H., 1975. "On the stability of gauss-jordan elimination with pivoting". *ACM New York, NY, USA*, Vol. 18, pp. 20 – 24.

Petrucci, L., 2007. "Iso/iec 15909-2, concepts of high-level nets and cpn". In *Presentation at the Workshop on Petri Nets Standards (PNS'07, associated with PETRI NETS'07), Siedlce, Poland.*

Pnml.org, 2011. "Pnml.org - pnml reference site". URL `http://www.pnml.org/tools.php`.

Ramchandani, C., 1974. "Analysis of asynchronous concurrent systems by timed petri nets". Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

Ramos, R.L.C.B. and Silva, J.R., 1998. "A formal model for integrated complex dynamic systems". In *5th IFAC Workshop on Intelligent Manufacturing Systems - IMS'98*.

Silva, J.R. and Miyagi, P.E., 1996. "A formal approach to pfs/mfg: a petri net representation of discrete manufacturing systems". In *Studies in Informatics and Control*. IC Publications, Romenia.

Silva, M. and Colom, J.M., 1988. "On the computation of structural synchronic invariants in p/t nets". Vol. 340 of *Lecture Notes in Computer Science*, pp. 386–417.

Silva, M., Terue, E. and Colom, J.M., 1998. "Linear algebraic and linear programming techniques for the analysis of place/transition net systems". In S.B.. Heidelberg, ed., *Lectures on Petri Nets I: Basic Models*, Vol. 1491 of *Lecture Notes in Computer Science*, pp. 309–373.

Tanimoto, S., Yamauchi, M. and Watanabe, T., 1996. "Finding minimal siphons in general petri nets". *IEICE Trans. on Fundamentals in Electronics, Communications and Computer Science*, Vol. 11, pp. 1817–1824.

WAKUDA, M., YAMAUCHI, M., TAOKA, S. and WATANABE, T., 1999. "A fast and space-saving algorithm fmsn for computing petri net invariants with supports containing all specified nodes". *EIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, Vol. 99, No. 204, pp. 37–44.

Wonham, W.M. and Ramadge, P.J., 1987. "On the supermal controllable sublanguage of a given language". *SIAM J. Control Optim.*, Vol. 25, pp. 637–659. ISSN 0363-0129. doi:10.1137/0325036. URL `http://portal.acm.org/citation.cfm?id=37121.37130`.

Yalamov, P.Y., 1991. "On the backward stability of gauss-jordan elimination". In *Computing*, Springer Wien, Vol. 47 of *Mathematics and Statistics*, pp. 193–197.

Yamalidou, K., Moody, J., Lemmon, M. and Antsaklis, P., 1996. "Feedback control of petri nets based on place invariants". *Automatica*, Vol. 32, pp. 15–28. ISSN 0005-1098.

9. Responsibility notice

The author(s) is (are) the only responsible for the printed material included in this paper