

École Doctorale STIM  
Sciences et Technologies de l’In-  
formation et des Matériaux.

**Spécialité :** Automatique, Robotique et  
Traitement du Signal  
**Laboratoire :** IRCCyN  
**Équipe :** Systèmes Temps Réel

## Model checking for Systems Engineering: From EFFBDs to time Petri nets

Seidner, Charlotte

E-mail: charlotte.seidner@irccyn.ec-nantes.fr

**Abstract:** This paper describes how verification practice in Systems Engineering can be improved by providing efficient and usable tools based on model checking methods in safety and dependability analysis of complex, high-level models. We consider the widely-used model of Enhanced Functional Flow Block Diagrams (EFFBDs); after proposing a structural translation of EFFBDs to time Petri nets (TPNs), proved to preserve temporal behavior, we were able to extend a number of fundamental theoretical results on TPNs to EFFBDs. As an application of these results, we developed a simulation and verification software tool that benefits from TPNs powerful analysis techniques while concealing the inherent complexity to the end-user.

**Keywords:** *Model checking, Systems Engineering, time Petri nets, embedded systems design and modeling, formal verification.*

**Partnerships:** Sodus (*industrial partner*)

## 1 Introduction

**Systems Engineering and formal verification** Systems Engineering (SE) is defined by the International Council on Systems Engineering (INCOSE) as an “interdisciplinary approach” to perform the “realization of successful systems”, from the definition of “customer needs and required functionality early in the development cycle” to the “design synthesis and validation” steps [1]. Application fields are quite numerous and include defense, aerospace engineering, railroad transport, computer science etc.

Amongst the tasks assigned to the systems engineer lies *verification* and *validation* so as to ensure such properties as safety and dependability. However, the resort to usual, simple methods (such as performing simulations) is insufficient on large, complex models as exhaustive testing would be very long, if not impossible. On the other hand, *model checking* may address this issue by checking whether, given a formal expression of a property,  $\varphi$ , and a formal model of the system,  $S$ ,  $S$  verify  $\varphi$ .

Moreover, when considering the high-level behavioral models commonly used in SE, it appears more efficient first to supply them with a formal behavioral semantics, then to translate them into formal, lower-level models on which model checking techniques and results are well established.

**Our contribution** The work presented in this report focuses on *Enhanced Function Flow Block Diagrams* (EFFBDs), a graphical formalism widely used in SE projects.. Although no formal semantics was established (to our knowledge), this model has proved to be consistent and mature over the last decades. As for the lower-level model, time Petri nets or TPNs [2] appeared well-adapted to the EFFBD structure. Note that the material presented here is a largely simplified version of a technical report [3] to which the reader is referred for further information.

**Organization of the paper** Section 2 describes the EFFBD and TPN formalisms. Section 3 gives an insight on the translation method and properties while Section 4 introduces the software tool developed as an application of these properties. Finally, Section 5 concludes the paper by presenting our further research work.



Informally speaking, a TPN  $\mathcal{T}$  is a tuple formed of  $P$ , the set of *places* and  $T$ , the set of *transitions*, connected by *weighted arcs*. While places represent *states* of the modeled system (e.g. “the processor is idle” or “the function is executed”), transitions model the *events* that bring the system from a state to another (e.g. “an alarm is received” or “the function execution is completed”). Moreover, places may contain any number of *tokens*; they may indicate a level of resources or the validation of a condition. A *marking*  $M$  of the net is an element of  $\mathbb{N}^P$  such that  $\forall p \in P$ ,  $M(p)$  is the number of tokens in the place  $p$ . Finally, each transition holds a temporal interval indicating when it can be fired once *enabled*, i.e. since all of its input places had enough tokens (the quantity needed being indicated by the weight of the arc linking the places to  $t$ ).

The semantics of a TPN is also defined as a TTS  $\|\mathcal{T}\| = (Q, q_0, T, \rightarrow_{\mathcal{T}})$ ; states are formed of a marking  $M$  and a valuation  $v$ . The latter associates to each transition  $t$  the time elapsed since it was enabled.

### 3 Structural translation from EFFBDs to TPNs

The approach proposed in our work is to perform a structural translation by using *elementary TPN patterns* for each type of node and for each item. The complete TPN is then simply obtained by connecting patterns together.

We then defined an equivalence relation  $\sim$  between the states of any EFFBD  $\mathcal{E}$  and the states of its TPN  $\mathcal{T}_{\mathcal{E}}$  counterpart. It includes the following relations:

- an activity is equivalent to a certain marking (for instance, “function  $f$  is executing” corresponds to “place  $p_{Executing^f}$  contains one token”);
- a function clock valuation is equivalent to a certain transition clock valuation.

We proved that  $\mathcal{T}_{\mathcal{E}}$  simulates  $\mathcal{E}$ : from a state  $s$  of  $\mathcal{E}$  equivalent according to  $\sim$  (or simply  $\sim$ -equivalent) to a state  $q$  in  $\mathcal{T}_{\mathcal{E}}$ , every action (including time elapsing) that can be taken can also be taken from  $q$ ; moreover, the resulting states  $s'$  and  $q'$  are  $\sim$ -equivalent. Conversely, we showed that, when considering  $\sim^{-1}$ ,  $\mathcal{E}$  also simulates  $\mathcal{T}_{\mathcal{E}}$ . Therefore, the equivalence relation  $\sim$  is a *strong timed bisimulation*.

Therefore, it is possible to use the tools and results developed over the last decade on TPNs in order to perform powerful analyses on the EFFBDs without any information loss on the system behavior. Two of these results are given hereunder.

**Theorem.** *For any TPN  $\mathcal{T}$  with the semantics  $(Q, q_0, T, \rightarrow_{\mathcal{T}})$  and a given  $k \in \mathbb{N}$ , the following problem is decidable:*

$$\forall (M, v) \in Q, \quad \forall p \in P : \quad M(p) \leq k \quad ?$$

**Corollary.** *For any EFFBD  $\mathcal{E}$  with the semantics  $(S, s_0, \mathcal{N}, \rightarrow_{\mathcal{E}})$  and a given  $k \in \mathbb{N}$ , the following problem is decidable:*

$$\forall (A, C, I, \nu) \in S, \forall item \in \mathcal{I} : \quad I(item) \leq k \quad ?$$

As a result, it is always possible to check whether the item level of any EFFBD stays under a limit specified by the system designer, which is particularly needed when assessing the size of a system in the design process.

### 4 Kimono, an application tool

The different transformations have been implemented in a SE development platform, KIMONO, designed as a series of ECLIPSE plug-ins and specifically developed for a branch of the French Department of Defense [5]. KIMONO integrates a TPN-analysis software tool, ROMÉO, which includes a TPN editing module, a simulation engine and most notably a model checker performing on-the-fly verification of temporal logic properties [6].

Therefore, it is then possible to perform simulations of the EFFBDs by using the TPN simulator, in a complete transparent way. Furthermore, a verification module was added to KIMONO, in order to verify complex, safety-critical properties, expressed in a *natural language* by the designer, as described hereunder.

More specifically, the module performs the following successive steps:

1. create a safety property  $P$  expressed over the system functions;
2. transform this high-level property into a temporal logic formula  $F$  over the corresponding TPN;
3. verify  $F$  in ROMÉO;
4. compute the truth value of  $F$  and, if applicable, a “witness” of the formula (i.e. a sequence of transition firings leading to a state validating or contradicting  $F$ );
5. compute the truth value of  $P$  and a witness of the property in terms of a sequence of functions.

Available high-level properties are of the typical following form:

- “the model  $\mathcal{E}$  always reaches its final state”;
- “the model  $\mathcal{E}$  always reaches its final state in less than  $t$  time units” (with  $t \in \mathbb{N}$ );
- “given a set of functions  $\mathcal{F}$ , only one function in  $\mathcal{F}$ , at most, can be executed at the same time”;
- “given two functions  $f_1$  and  $f_2$ , executing  $f_1$  leads to the execution of  $f_2$  in less than  $t$  time units” (with  $t \in \mathbb{N}$ ).

It should eventually be noted that steps 2 to 4 in the verification process are totally concealed to the user: the only entities he has to manipulate are on the EFFBD model, whereas the actual computing is performed on the underlying TPN. Moreover, by obtaining a witness of the property in terms of high-level elements, and by using the simulation tool, it is possible to perform a sharp analysis of the system, thus showing the potential benefit of the method in system design, possibly at an early stage of the process. Even if only a coarse model of the system is designed, a number of analyses can be performed and the eventual design weaknesses effectively pointed out.

## 5 Conclusion

This paper has introduced a formal description and the behavioral semantics of EFFBDs; to our knowledge, it had never been formally expressed, although the language is widely used in SE design. This led to the definition of a transformation method from EFFBDs to TPNs, proved as preserving the temporal behavior of the high-level model. As a result, a number of fundamental properties, inherited from the research works carried on TPN were applied to EFFBDs. As a consequence, it is possible to perform safety assessment on models designed by a typical systems engineer via model checking techniques, in a completely transparent way.

Further work should focus on further investigating the translation of high-level properties to temporal logic formulas, and on the study of the complexity of the method algorithms. In addition, it is planned to extend the original EFFBD formalism to model “hybrid entities” such as continuous rates for the production or consuming of items, etc.

## References

- [1] INCOSE Technical Board. *Systems Engineering handbook: a “what to” guide for all SE practitioners*. INCOSE, 2a edition, June 2004.
- [2] P. Merlin. *A study of recoverability of computing systems*. PhD thesis, Dpt. of Computer Science, University of California, Irvine (CA), USA, 1974.
- [3] C. Seidner and O. H. Roux. On the formal verification of EFFBD models using a structural translation to time Petri nets. Technical Report RI2007-3-3695, IRCCyN, Nantes, France, October 2007.
- [4] J. Long. Relationships between common graphical representations in Systems Engineering. In *5<sup>th</sup> International Symposium of the INCOSE*, St. Louis (MO), USA, July 1995. Updated July 2002.
- [5] C. Seidner, J.-P. Lerat, and O. H. Roux. Usability of formal verification on EFFBD models: Applying Petri nets to Systems Engineering issues. In *17<sup>th</sup> International Symposium of the International Council on Systems Engineering (IS2007)*, San Diego, CA, June 2007.
- [6] G. Gardey, D. Lime, M. Magnin, and O. H. Roux. Romeo: A tool for analyzing time Petri nets. In *Proceedings of the 17<sup>th</sup> International Conference on Computer-Aided Verification (CAV’05)*, Lecture Notes in Computer Science, 2005.