Modelling Business Processes Using Evolutionary Generated Petri Nets

Jan Plucar[1], Ondrej Grunt[2], Pandian Vasant [3], Ivan Zelinka [4]
{jan.plucar@vsb.cz [1], ondrej.gruntr@vsb.cz[2], pvasant@gmail.com[3]}

VSB-Technical University of Ostrava,17. listopadu 15, 708 33 Ostrava[1,2]
Universiti Teknologi PETRONAS,Department of Fundamental and Applied Sciences
32610 Seri Iskandar, Malaysia[3]

Abstract. Evolutionary algorithms or computations (EC) are commonly used for solving problems such as optimization and reverse engineering of complex systems. In this paper, we present bio-inspired approach in a process of evolving design for the workflow processes. Workflow management systems often offer modelling capabilities to support construction of business processes. However, constructing formal workflow model for the extensive business process is a non-trivial task. Workflow process could be described as bipartite graph and therefore interpreted by Petri Nets. The goal is to describe a process for evolving and verification of petri nets.

Keywords: Evolving petri nets, Analytic programming, Genetic algorithm

1 Introduction

Every company management is trying to obtain reliable partners in business and make their company successful and profitable. Many methods and methodologies are used to achieve this goal. The quality of services offered by the company is a crucial factor when establishing partnership. In the past, several standards that are trying to assess maturity level of company were introduced. One such standard is Software Process Improvement and Capability Determination (SPICE), which is described in the ISO / IEC 15504 framework. Thanks to the success of the standard, more specialized versions such as Automotive SPICE, Medi SPICE or Enterprise SPICE were developed. All these standards have one thing in common. It is description of company's business processes, which are then evaluated and improved. Improvement of the business process is always difficult, and initiatives often fail to achieve better results. Therefore, it is important to understand the initial baseline level of the process, and re-assess the situation after improvements have been performed.

This article lays the foundation for the possibility of automatic business process creation with the respect to expected process parameters and outputs. In the first part of this paper, methods, languages and algorithms that were used are presented. Then visualized overview of the system is described and proposed solution is discussed.

2 Petri Nets

From this point, we will focus on one business process that will be represented by Petri net (PN). Such PN will be used as a model of real process. Petri Nets is a modeling language designed by Carl Adam Petri capable of describing parallel and distributed systems [1]. PN model is graphically represented by directed bipartite graph consisting of two types of nodes:

- Places: nodes representing current condition of a net normally indicated by circle symbol.
- Transitions: nodes representing events occurring in the net normally indicated by bar symbol.

Nodes in PN are connected by directed arcs, which can be divided into three groups: input arcs, output arcs and inhibitor arcs. Arcs can only connect place to a transition and vice versa.

Places in PN may contain tokens, which are indicated by a number of dots corresponding to number of tokens present at a place. Tokens move through the net by firing an enabled transition (transition that has at least one token present at its input place). Distribution of tokens in the net is called marking. The initial state of the net is then called initial marking.

As basic PN provide only immediate transitions, where token moves immediately after a transition is enabled, stochastically determined delay can be applied to a transition to reflect events more accurately [2]. Such transition then fires a token only after it is enabled and amount of time given by stochastically determined delay passes. This extension of PN language is called Stochastic Petri nets (SPN) [3,4,5] and was chosen as tool for modelling of our business process.

Definition 1 A Petri Net is a four-tuple *PN = (P,T,F,M0)* where:

1. P is a set of places.
2. T is a set of transitions.
3. F, $F \subset (P\ x\ T) \cup (T\ x\ P)$ is a set of arcs.
4. M_0 is the initial marking.

2.1 PN properties

One of the advantages of PN approach is its ability to analyze resulting PN (or SPN) model in objective matter by proving properties of the net [6]. Their interpretations then heavily depend on the purpose of the constructed model. For our PN representation of business process, following properties were deemed as the most useful in further verification of the model:

Reachability - Let M_y and M_x be two different markings in a net. We call marking M_i reachable from marking M_j if there is a sequence σ_M such that $M_i[\sigma_M\rangle M_j$. Reachability is used to estimate the probability of a PN model being in a given marking *M*. All markings that are reachable from the initial marking then form reachability set $RS(M_0)$ of the model (an example of reachability set is shown in Figure 1).

M_0	$=$	$2p_1$	$+$			p_3				
M_1	$=$	p_1	$+$	p_2	$+$	p_3				
M_2	$=$	p_1	$+$							p_5
M_3	$=$			p_2	$+$	p_3	$+$	p_4		
M_4	$=$			p_2	$+$					p_5
M_5	$=$							p_4	$+$	p_5
M_6	$=$	p_1	$+$			p_3	$+$	p_4		
M_7	$=$			$2p_2$	$+$	p_3				
M_8	$=$					p_3	$+$	$2p_4$		

Fig. 1. Reachability set example

However, reachability set only contains information about markings and not about transitions sequences leading to such markings. To obtain this information, reachability graph (*RG*) can be constructed, in which each node represents a marking in reachability set (i.e.M_0, M_1, etc.). These nodes are then connected with directed arcs, each labeled by a transition that fired (i.e.t_0, t_1, etc.). An example of resulting graph is shown in Figure 2.

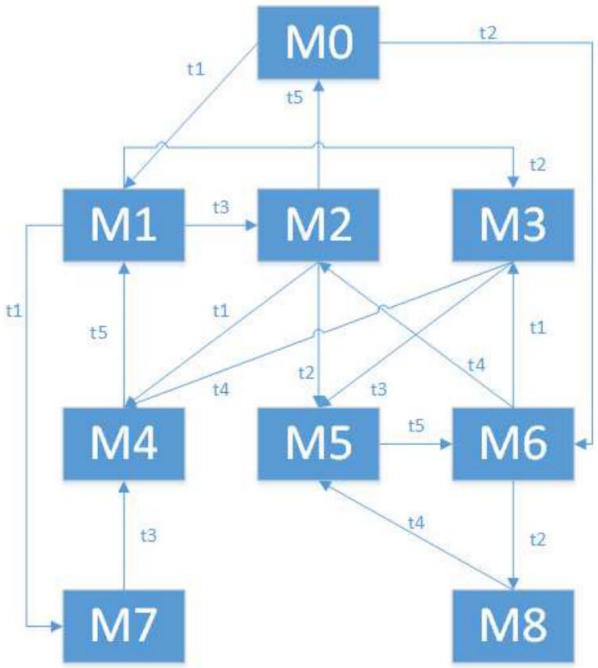

Fig. 2. Reachability graph for reachability set from Fig.1

Absence of deadlock - Deadlock is a state of PN in which no transition can fire, thus barring tokens from moving through the net. A PN model is deadlock-free, if all models resulting from all possible parametric initial markings do not contain a deadlock. While deadlock-freeness cannot be defined for PN models, it is possible to determine if PN model can potentially have a deadlock by examining its graph structure.

Liveness - A transition *t* is said to be live in PN model if and only if, for each marking M_i reachable from the initial marking M_0 exists a marking M_j reachable from M_i such that *t* is enabled in M_j. PN model is said to be live if all $t \in T$ are live in it.

Transition that is not live is said to be dead. For each dead transition *t*, it is possible to find marking *M* such that *t* is not enabled in none of markings in *RS*(*M*).

Important consequence of liveness is that PN model in which at least one transition *t* is said to be live cannot contain a deadlock. However, that is not sufficient condition for PN model to be live, as it can contain some dead transitions as well. Such PN model is then said to be partially live.

In PN model analysis, liveness relates to the possibility of a transition to fire infinitely often.

Boundedness - A place p is said to be *k*-bounded in PN model if and only if, for each reachable marking *M* the number of tokens in $p \leq k$. PN model is said to be *k*-bounded if every $p \in P$ is *k*-bounded.

Boundedness of PN model implies the finiteness of the state space. If PN model comprises of *N* places and is *k*-bounded, the number of states does not exceed $(k+1)^N$.

2.2 Analysis techniques

To prove PN properties, state space analysis techniques are used. State space analysis techniques are inherently non parametric with respect to the initial marking, since they require its complete instantiation. These techniques are based on the construction of *RS* and *RG* of PN system and thus are feasible only if *RS* and *RG* are finite.

Following the construction of *RG*, properties of PN system may be proved using graph analysis algorithms:

Reachability - Marking M_j is reachable from marking M_i in PN system if its RG contains directed arc from M_i to M_j.

Absence of deadlock - Deadlock in PN system may be identified by looking for a node in *RG* with no output arcs. Absence of such node is sufficient for the system to be absent of deadlock.

Liveness - Transition *t* in PN system is live if and only if the *RG* contains no dead marking and *t* labels some arc of every strongly connected component of *RG* (i.e. each node in graph can be reached from every other node).

Boundedness – Place $p \in P$ of PN system is *k*-bounded if and only if

$$k = \max_{M \in RS} \mathrm{M(p)}$$

2.3 Workflow nets

PN method is capable of modeling workflow, i.e. process which has clearly distinguished input place, which has no input arc, and output place, which similarly has no output arc and that an addition of arc connecting input and output place results in strongly connected net. PN representing workflow is called Workflow Petri net (WPN) [7].

In terms of important properties mentioned in previous section, WPN has a deadlock and the deadlock place is reachable from all markings. Connecting input and output place by directed arc then results in live model.

One important property of WPN is soundness of constructed model. It is believed, that every well-designed business process can be represented by sound WPN [8, 9, 10]. Let N be WPN and let N_r be WPN with added arc between input and output place. Then N is sound if N_r is live and k-bounded for some non-infinite k. Soundness property is necessary condition for the application of learning algorithms to retain correct structure of a net. Properties of resulting WPN or PN model may then be proved by application of previously mentioned analysis techniques.

3 Experiment Overview

Petri nets are useful tool for visualisation of the processes. However, constructing and optimizing PN that describes process with hundreds of possible transitions and places is time and resources demanding task. Therefore, Evolutionary Petri Nets (EPN) are proposed as a solution to this task.

Typical feature of evolutionary algorithms is that they are based on working with populations of individuals. We can represent the population as a matrix $M \times N$where columns represent the individuals. Each individual represents a solution to the current issue. In other words, it is set of cost function argument values whose optimal number combination we are looking for. One of early evolutionary algorithms can be found in Price [11]. The main activity of evolutionary algorithms is the cyclic creation of new populations which are better than the previous ones. Better population is a population whose individuals have better fitness. In our case, it is EPN that generates expected output and demonstrates best properties, see 2.1. Similar approach was proposed in [12] and [13], where authors used genetic programming (GP) as a primary tool for PN construction.

Overview of the proposed solution is visualised in Figure 3. There are several aspects that must be defined before system starts. Most important are PN building blocks, system loop termination condition and PN outputs. Then initial population of PNs is created and system is ready to start. Once started, system loops several tasks:

- Every PN in population is simulated in Petri nets simulator. In our research, we have so far used own simple solution for the simulator software. In the future work, PNs will be transformed to petri nets modeling language to get access to more sophisticated simulator tools. Output of the simulator is set of PNs in their final state.
- Genetic algorithm selects most suitable PN which will be used as a member of new population. This member is then modified using Analytic programming (AP), that is described in section 4

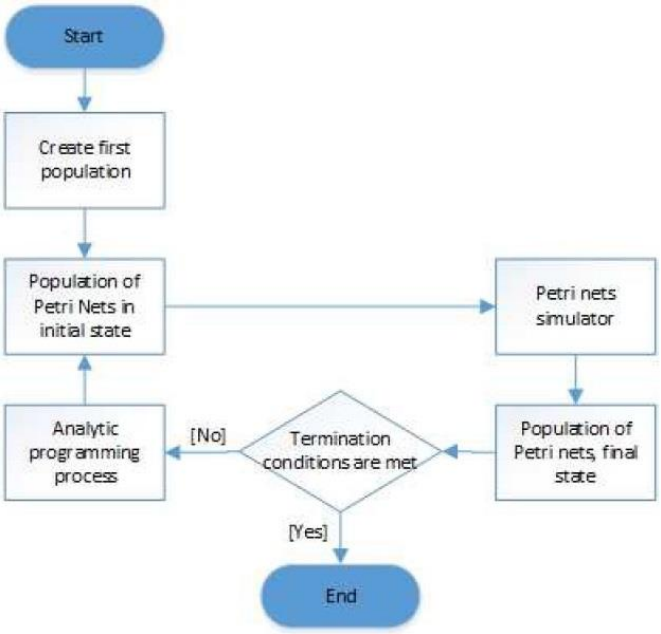

Fig. 3. System overview

4 Analytic programming

Analytic programming was inspired by the numerical methods in Hilbert functional spaces and by GP. The principles of AP are somewhere between these two philosophies: From GP stems the idea of the evolutionary creation of symbolic solutions, whereas the general ideas of functional spaces and the building of resulting function by means of a search process (usually done by numerical methods such as the Ritz or Galerkin method) are adopted from Hilbert spaces. Like genetic programming or grammatical evolution, AP is based on a set of functions, operators and so-called terminals, which are usually constants or independent variables, for example:

- functions: Sin, Tan, Tanh, And, Or
- operators: +, -, *, /, dt
- terminals: 2.73, 3.14, t

All these ‘mathematical’ objects create a set from which AP tries to synthesize an appropriate solution. The main principle of AP is based on discrete set handling (DSH). Discrete set handling itself can be seen as a universal interface between evolutionary algorithm and the problem to be solved symbolically. That is why AP can be performed using almost any evolutionary algorithm. In this case, Differential Evolution (DE) has been used,

see section 5. Analytical programming, together with a few basic examples, is discussed in more detail in [14] and [15].

Briefly, in AP, individuals consist of non-numerical expressions (operators, functions) as described above, which are in the evolutionary process represented by their integer indexes. This index then serves as a pointer into the set of expressions and AP uses it to synthesize the resulting function-program for cost function evaluation.

5 Differential evolution

Differential evolution (DE) is one of the methods used in evolutionary computations. It is typical for the evolutionary algorithms to work with population of individuals. Each individual consists of a vector of parameters that represent solution to the problem. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand.

DE is originally due to Storn and Price [16]. Books have been published on theoretical and practical aspects of using DE in parallel computing, multi objective optimization, constrained optimization, and the books also contain surveys of application areas. Excellent surveys on the multi-faceted research aspects of DE can be found in journal articles like [11, 17, 18, 19].

```
For I = 0 to Generation do
      For j = 0 to NP do
            Select j-individual
            Select three random individuals from population
            Breed new individual
            Compute the new individual fitness
            Choose a better one from both individuals to new
            population
      End
End
```

Where input parameters are:

- NP, F, CR, N: described in section 6
- X: initial population (vector)
- fcost: function returning fitness of current solution

For our purposes the DE/rand/1 algorithm mutation has been choosen. The notation DE/rand/1 specifies that the vector *v* to be perturbed is randomly chosen and the perturbation consists of one weighted vector.

$$v = x_{r1,j}^{G} + F \times (x_{r2,j}^{G} - x_{r3,j}^{G}) \qquad (1)$$

Due to this mutation, new individuals are not affected by the temporary best individual from generation and space of possible solutions is searched through uniformly. More detailed description of Differential Evolution can be found in Price [11].

6 Experiment design

Our model was evolved using combination of analytic programming and differential evolution. Proposed solution was implemented in C# programming language. DE uses fitness function in order to breed more suitable individuals. This fitness function was represented as a rate of difference between expected outputs and EPN output. We have tried various different configurations of DE parameters. Values of these constants were chosen empirically based on the performance of DE algorithm. These values are shown in the Table 1. Other parameters like number of generations and population size for both methods were determined from the literature [16, 20]. Most promising result is shown in Fig. 4. This EPN reflects mobile phone switching between networks during upload process of files between server and mobile phone in the environment comprising of 2G, 3G and 4G networks. Evolution of EPN fitness is depicted in Fig. 5.

Table 1. Parameters setting

Parameter name	Value
Number of generations	50
NP	100
F	1
CR	0.5
N	2

Where parameters are:

- NP: population size
- F: mutational constant
- CR: crossover threshold
- N: dimension of problem

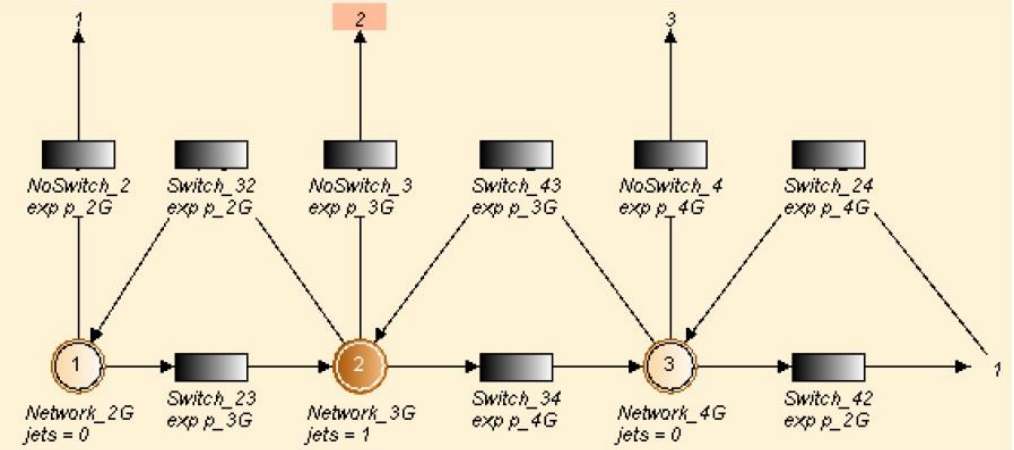

Fig. 4. Evolved petri net

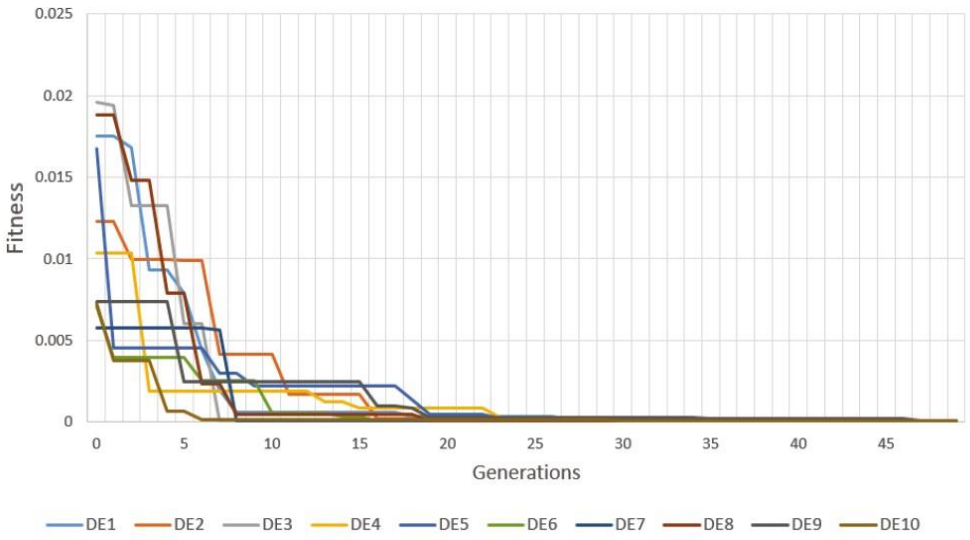

Fig. 5. Fitness evolution

7 Conclusion

In this article, we have outlined the basic elements of the experiment, which aims to define process for petri nets evolution. One of the advantages of PN approach is its ability to analyze resulting PN (or SPN) model in objective matter by proving properties of the net. We have described properties that are being analysed in early stage of experiment and many more

may be introduced later. In section 6 we have mentioned basic parameters setup of the system and one of the resulting EPNs. More technical description along with experiment performance measurements and other examples will be presented in future papers. Analytic programming proved promising results, its potential has to be however verified on larger amount of evolved petri nets.

Acknowledgment

This work was supported by The Technology Agency of the Czech Republic TACR TF01000091 and SGS No. SGS 2016/175, VSB-Technical University of Ostrava.

References

[1] Peterson, James L.:Petri Nets. ACM Computing Surveys 1977; 9 (3), pp. 223--252.
[2] Symons, Fred J.W.: Description And Definition Of Queueing Systems By Numerical Petri Nets. ATR, Australian Telecommunication Research 1980; 13 (2), pp. 20--31.
[3] Florin, G., Long, P., Natkin, S.: Evaluation of Functional Reliability of Information Systems by Stochastic Petri Nets. 1980.
[4] Molloy, M. K.: Discrete Time Stochastic Petri Nets. IEEE Transactions on Software Engineering 1985; SE-11 (4), pp. 417--423.
[5] Florin, G., Natkin, S.: Les reseaux de Petri stochastiques. Technique et Science Informatiques 1985; 4(1).
[6] Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.:Modeling with Generalised Stochastic Petri Nets. Universita degli Studi di Torino, 1994.
[7] Esparza, J., Leucker, M., Schlund, M.: Learning Workflow Petri Nets. Fundamenta Informaticae -- Applications and Theory of Petri Nets and Other Models of Concurrency 2010; 113 (3-4), pp.205--228.
[8] Yang, X., Yu, T., Xu, H.: A Novel Framework of Using Petri Net to Timed Service Business Process Modeling. International Journal of Software Engineering and Knowledge Engineering 26 (4), pp. 633--652 (2016).
[9] Pang, S., Yan, B., Liu, X., Jia, H.: A novel approach for dynamic business processes and process changes with Petri net. Journal of Computational and Theoretical Nanoscience 12 (7), pp. 1457--1461 (2015).
[10] De Rezende, L.P., Julia, S., Cardoso, J.: Possibilistic WorkFlow nets for dealing with cancellation regions in business processes. ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems 2, pp. 126--133 (2016).
[11] Price, K.: Differential evolution: a fast and simple numerical optimizer. In: Proc. 1996 Biennial Conference of the North American Fuzzy Information Processing Society, pp. 524–527. IEEE Press, New York (1996)
[12] Nobile, M. S., Besozzi, D., Cazzaniga, P., Mauri, G.:The Foundation of Evolutionary Petri Nets.CEUR Workshop Proceedings, 988, pp. 60-7 (2013)
[13] Nummela, J., Juistrom, B.A., Evolving, petri nets to represent metabolic pathways, GECCO 2005 - Genetic and Evolutionary Computation Conference (2005)
[14] Zelinka, I., Oplatkova, Z., Nolle, L., Analytic programming - Symbolic regression by means of arbitrary evolutionary algorithms, International Journal of Simulation: Systems, Science and Technology, 6 (9), pp. 44-56, (2005)
[15] Senkerik, R., Kominkova Oplatkova, Z., Pluhacek, M., Zelinka, I., Analytic programming—a new tool for synthesis of controller for discrete chaotic Lozi map, Lecture Notes in Electrical Engineering 307, pp. 137-152 (2014)

[16] Price, K., Storn, R.: Differential Evolution – A simple evolutionary strategy for fast optimization. Dr. Dobb's Journal 264, 18–24 and 78 (1997).
[17] Price, K.: Genetic Annealing. Dr. Dobb's Journal, 127–132 (October 1994)
[18] Malamura, E., Murata, T.: Hybrid system modeling and operation schedule optimization for gas transportation network based on combined method of DE, GA and hybrid petri net.Proceedings - 2016 5th IIAI International Congress on Advanced Applied Informatics, pp. 1032--1035, (2016)
[19] Letia, T.S., Kilyen, A.O.: Evolutionary synthesis of hybrid controllers. Proceedings - 2015 IEEE 11th International Conference on Intelligent Computer Communication and Processing, pp. 133--140 (2015)
[20] Price, K.: An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 79–108. McGraw-Hill, London (1999)