DE GRUYTER OPEN

Applied Computer Systems

doi: 10.1515/acss-2014-0016

2014/16

The Perspective on Data and Control Flow Analysis in Topological Functioning Models by Petri Nets

Erika Asnina[1], Begoña Cristina Pelayo García-Bustelo[2],
[1]*Riga Technical University, Latvia*, [2]*University of Oviedo, Spain*

***Abstract* – The perspective on integration of two mathematical formalisms, i.e., Colored Petri Nets (CPNs) and Topological Functioning Model (TFM), is discussed in the paper. The roots of CPNs are in modeling system functionality. The TFM joins principles of system theory and algebraic topology, and formally bridges the solution domain with the problem domain. It is a base for further automated construction of software design models. The paper discusses a perspective on check of control and data flows in the TFM by CPNs formalism. The research result is definition of mappings from TFMs to CPNs.**

***Keywords* – Control flows, data flows, model verification, Petri nets, topological functioning model.**

I. Introduction

Software developers used to eliminate a process of problem domain modeling, since the main artifacts of their work usually are executable code and software requirements specification that is based on this code. Model Driven Engineering is a methodology that moves developers' focus from coding as such to the creation of transformable (and sometimes also executable) software architecture specifications. It uses main viewpoints and corresponding models (specifications) of Model Driven Architecture (MDA) [1], namely, a computation independent model (CIM), a platform independent (PIM) model, and a platform specific model (PSM). The CIM, a model discussed in this paper, describes system requirements and a way a system works within its environment, while details of the structure and realization of the software application are hidden or yet undetermined. It is business "owner's" viewpoint called a domain model and a domain vocabulary. The CIM usually is informal and its transformation mostly is manual [2]. The CIM is a specification of both software and the problem domain (or at least of its part related to software). Therefore, it should be able to provide a correspondence between them [2], [3]. But this is possible only if the CIM is formal [4].

Receiving valid code from a transformable model (a graphical specification) still is an open question. There are two aspects of this. The first one is transformation of models into code itself, and the second one is validity of models themselves. Here, directly the verification of one of CIMs, a Topological Functioning Model (TFM), is discussed, since it serves as a bridge that joins the problem domain and the solution domain in the computation independent manner [5], [6]. The cause of this research is manual verification of the TFM cause-and-effect flows at the present time. When the TFM is composed, each data and its control need to be checked on adequacy to the domain functioning. Certainly, this task does require participation of domain experts. The open question is automated (tool) assistance in this process, since in case of a large or complex graph the number of possible flows can be too big for human mind. On the other hand, Colored Petri Nets (CPNs) do provide formalism for tracing data moves and control activities. Therefore, if a TFM could be transformed to a CPN, then it would be possible to trace and check data and control flows presented in the TFM. The goal of this research is to find out the possibility of mapping data and control flows of the TFM and CPNs.

The paper is organized as follows. Section II describes formalism and previous application areas of the TFM in brief. Section III describes the formalism of Petri Nets (PNs), CPNs and their application areas in brief. Section IV discusses mappings of data and control flows from the TFM to CPNs. Section V highlights related work on application of PNs to business process modeling and verification of semantics in control and resource flows. In Conclusion, benefits and limitations of the results are discussed.

II. Topological Functioning Model

A. A Brief Introduction

The TFM is based on principles of algebraic topology and system theory. Mathematically, the TFM is represented in the form of a topological space (X, Θ), where X is a finite set of functional features (FFs), i.e., characteristics of the system under consideration, and Θ is the topology that satisfies axioms of topological structures; and it is represented in the form of a directed graph [7]–[10]. Topological spaces are described in detail in [11].

The detailed process of construction of the TFM is omitted in this paper, since the research objective is to analyze the possibility of the verification of functional characteristics of this model. In general, the construction includes definition of domain objects (a vocabulary), system's FFs and cause-effect relations among them, construction of a topological space of the problem domain, and separation of the TFM of the system from this topological space. The details are described in [12]–[16].

The TFM has topological (from algebraic topology) and functioning (from system theory) properties [13]. The topological properties are connectedness, closure, neighborhood and continuous mapping.

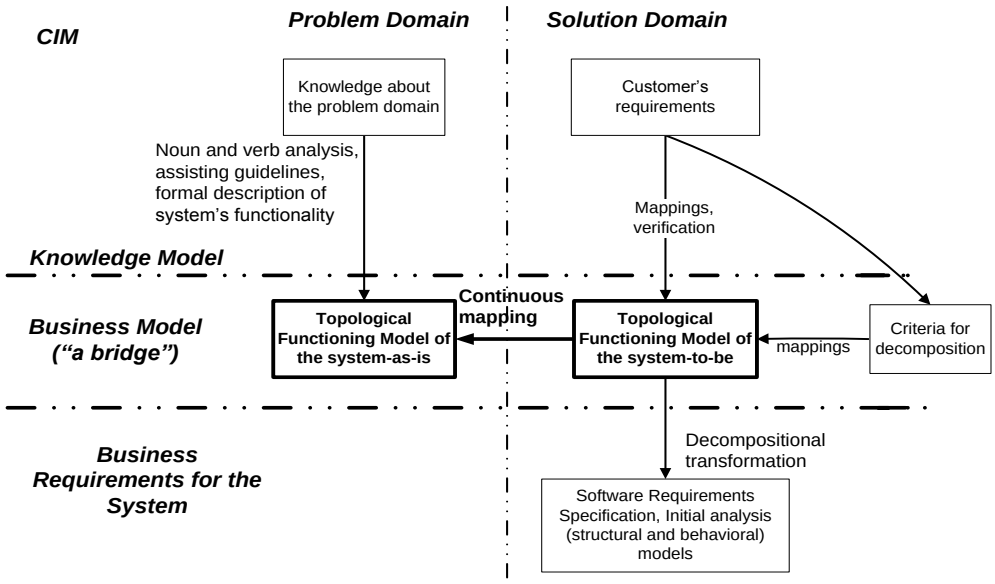

Fig. 1. "Bridging" the problem and the solution domains by using the TFM.

Mainly, they relate to formal determination of domain boundaries in terms of TFM FFs connected by cause-effect relations, and abstraction/refinement of the domain functionality.

The functional properties (Section II.C) are cause-effect relations, cycle structure (formed by FFs connected with cause-effect relations), inputs and outputs [10], [13].

B. TFM Application

The topological modeling of system functioning (TFM) was developed at Riga Technical University, Latvia. For the first time, its theoretic foundations were represented by Janis Osis in [7]. The TFM has been successfully used for the system analysis since the 1970s, and its application in different areas is being developed today as well. The recent more important studies concern its application for the purposes of biological systems modeling [17], and for introducing more formalism into the MDA framework and problem domain analysis, which grounds development of topological modeling language [3], [9], [12], [13]. Unfortunately, only publications after 1992 are available in English.

As mentioned in Introduction, the role of the TFM in the process of problem domain modeling is to "bridge" the problem domain and the solution domain through formal mathematical specification of knowledge of both domains. It is gained by using topological properties of the TFM as discussed in [2], [5], [10].

Fig. 1 illustrates the "bridging" at the business model level of the CIM. The TFM of the system in the problem domain (or "the system-as-is") is constructed by using analysis means for verbally expressed knowledge, namely, noun and verb analysis, assisting guidelines, and a formal specification for system functional characteristics. Customer requirements are mapped onto and verified by this TFM. During this process, a copy of the TFM is updated in accordance with the customer's software functional requirements. Customer requirements themselves also are updated by adding new or missing functionality that was not explicitly defined before and removing incorrect one (or out-of-date knowledge for the solution domain). The result is the TFM that specifies the solution domain ("the system-to-be"), which is in conformity with the TFM of the problem domain. Additionally, the verified software requirements specification is obtained. The TFM illustrates planned functionality holistically, and further development requires its decomposition on parts dedicated for implementation. According to a development plan and software requirements, criteria for TFM decomposition are identified. By using them, the TFM is decomposed into parts, which are transformed to parts of the initial analysis model. The TFM of the software is a subsystem of the TFM of "the system-to-be", and it serves as a root model for further decomposition and transformation into software design models. Exactly the important role that this holistic model plays in software development requires proper verification of model correctness. Certainly, the degree of verification of the correctness is limited with the possibility to check semantic data. However, usually this verification is performed manually with domain experts' help. It is hardened in case of a large or complex graph of the model.

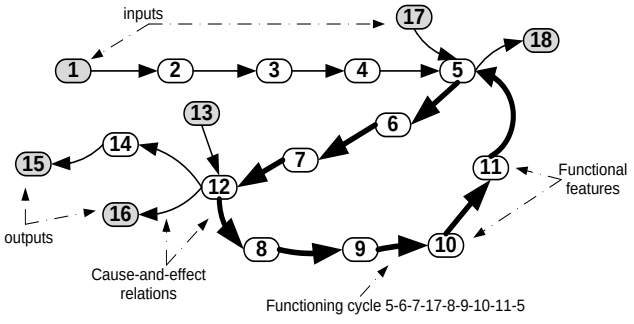

Fig. 2. The functional properties of the topological functioning model.

C. Formalism and Representation of Control and Data Flows in the Topological Functioning Model

TFM functional properties, which come from system theory, are closely related to specification of control and data flows within the domain. They are cause-effect relations, cycle structure (formed by FFs connected with cause-effect relations), inputs and outputs (Fig. 2). The explanation of each of them is given in detail in [13], [18]. Here, only the most important information is explained.

The definition of FF given in [13] is extended with incoming and outgoing combinations of cause-effect relations. It is specified as a unique 9-tuple <A, **R**, O, **PrCond**, **PostCond**, **Pr**, **Ex**, InRel, OutRel>, where:

- A is an action linked with a domain object;
- **R** is a result of that action (it is an optional element); it could be a domain object or a set of them, a message, a trigger for the effect event etc.;
- O is a domain object that gets the result of the action or a set **O** of objects, which are used in this action (in case when an item of **Ex** gets result R); it could be a role, a time period or a moment, catalogues etc.;
- **PrCond** is a set PrCond = {$prec_1$, …, $prec_i$}, where $prec_i$ is a precondition or an atomic business rule (it is an optional element) of the action;
- **PostCond** is a set PostCond = {$postc_1$, …, $postc_i$}, where $postc_i$ is a post-condition or an atomic business rule (it is an optional element) of the action;
- **Pr** is a set of responsible entities (systems or subsystems), which provide or suggest the action with the set of certain objects;
- **Ex** is a set of responsible entities (systems or subsystems), which enact the action;
- InRel is an expression of combinations of possible logical relations among incoming cause-effect relations in_1, in_2, …, in_i;
- OutRel is an expression of combinations of possible logical relations among outgoing cause-effect relations out_1, out_2, …, out_i.

A control flow in the TFM is represented by means of cause-effect relations from a cause FF to an effect FF. In the simplest case, a cause FF must have at least one effect, as well as an effect FF must have at least one cause. In complex cases, logical relations among cause-effect relations may form different logical combinations. The logical operators used within these combinations are negation NOR, conjunction AND, disjunction OR, and exclusive disjunction XOR. Cause-effect relations connect causally dependable FFs and may form functioning cycles. Causes and effects are stimulus sent to the system by the external environment (*inputs*) and reactions sent to the external environment by the system (*outputs*).

In other words, a cause-effect relation is a control flow from one FF to another one, which also transfers data/resources from one functional processing to another; while the FF is a data/resource handling node.

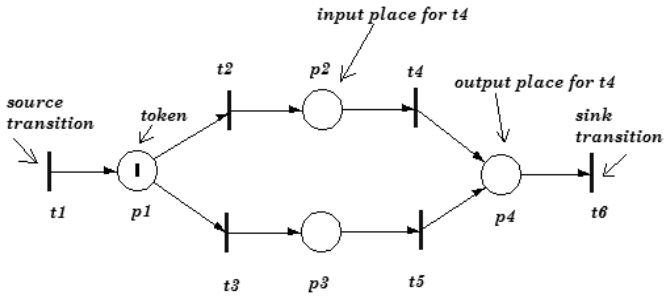

Fig. 3. An abstract Petri net.

III. PETRI NETS

Petri Nets (PNs) is a formal mathematical and graphical language for modeling systems with concurrency and resource sharing, which can be applied for any area or system that can be described graphically like flow charts. The concept of the PNs was introduced by Carl Adam Petri in 1962 [19]. PNs can be used to represent flows of both control and data. One of powerful PNs extensions are Colored Petri Nets (CPNs) [20]–[22]. In CPNs, colors denote types of tokens, while transitions conduct operations on multiple sets of tokens of those colors. CPNs have a module concept allowing CPN models to be organized into several modules, which are called by pages [23], thus allowing the hierarchical order. CPN models can be constructed by using CPN Tools [24].

Another kind of PNs that is used within this work is the so-called work flow (WF) nets. A Petri Net is a WF net if [25]: i) The Petri net has two special places – source and sink; ii) The sink place is only reachable from the source place with at least one token in it (this guarantees the proper termination of the process transaction); and iii) There is no dead transition in the PN with initial marking in the source place. On the basis of the WF nets, the net model can be extended to any CPN model and apply benefits of Petri Net theory. In this research the WF extension to CPNs is used [26] but without scheduling aspects.

A. Petri Net Formalism

A Petri net is a particular kind of a digraph (directed, weighted, bipartite), which contains two kinds of nodes – places and transitions, which are connected by directed arcs (Fig. 3).

Arcs are directed from a place to a transition or from a transition to a place. Arcs are labeled with their weights (classically they are positive integers). The digraph has an initial state called the *initial marking* $M0$. A marking is denoted by M, an m-vector, where m is the total number of places. $M(p)$ represents the pth component of M and is the number of tokens in the place p. The mark on arcs represents a number of tokens that each arc can transfer to or from the place. Usually in graphical representation places are illustrated as circles, and transitions are drawn as bars or boxes [19].

As mentioned in [19], the simple but very important rule in PN theory is the rule for transition *enabling* and *firing*. For instance, the concept of conditions and events is widely used in modeling. In the PNs, places represent conditions, and transitions represent events. An event has a certain number of

conditions that must be true in order to generate the event. The same, a transition has a certain number of input and output places correspondingly to the pre-conditions and post-conditions of the event (Fig. 3). The status of the conditions, *true* or *false*, is indicated by the presence or absence of a token in the place, correspondingly. If a PN is used for modeling data sharing, then *k* tokens in the place may indicate that *k* data items are available. As mentioned in [19], interpretation of places and transitions may differ, e.g., i) pre-condition, event, post-condition, ii) input data, computation step, output data, iii) resources needed, task or job, resources released, etc. The transition (firing) rule is as follows [19]:

- A transition *t* is said to be enabled if each input place *p* of *t* is marked with at least *w*(*p*, *t*) tokens, where *w*(*p*, *t*) is the weight of the arc from *p* to *t*.
- An enabled transition may or may not fire (depending on whether or not the event actually takes place).
- A firing of an enabled transition *t* removes *w*(*p*, *t*) tokens from each input place *p* of *t*, and adds *w*(*t*, *p*) tokens to each output place *p* of *t*, where *w*(*t*, *p*) is the weight of the arc from *t* to *p*.

A source transition is a transition without any input place and it is unconditionally enabled, while a sink transition is a transition without any output place, whose firing consumes tokens, but does not produce any [19].

B. Control and Data Flow Representation in Petri Nets

Control flows can be modeled in different ways. The common construct used for this purpose is illustrated in Fig. 3, when place *p*1 has two outgoing arcs: one to transition *t*2 and one to transition *t*3. This construct is called a conflict, choice, or decision. There are some basic examples that are useful in modeling [19]:

- Decisions are represented by using state machines (the previously mentioned construct of *p*1, *t*2 and *t*3 in Fig. 3), but state machines do not represent the synchronization of parallel activities;
- Parallel activities (i.e., logical operator AND on the outgoing control flows from the event) are represented by concurrent transitions (which are causally independent, i.e., one transition may fire before or after or in parallel with the other). Apart from that, each place in the net must have exactly one incoming and one outgoing arc (Fig. 4a);
- Conflicts (i.e., logical operators OR and XOR on the outgoing control flows) are a more complex case. Two events *e*1 and *e*2 are in conflict if either *e*1 or *e*2 can occur but not both (XOR), and they are concurrent if both events can occur in any order without conflicts (OR). The situation when conflicts and concurrency are mixed is called confusion (Fig. 4b).

Data flows can be modeled as a dataflow computation. As mentioned in [19], in this case tokens denote data values and the availability of data. Transitions represent operations on these data, and can be expressed as formal computational statements.

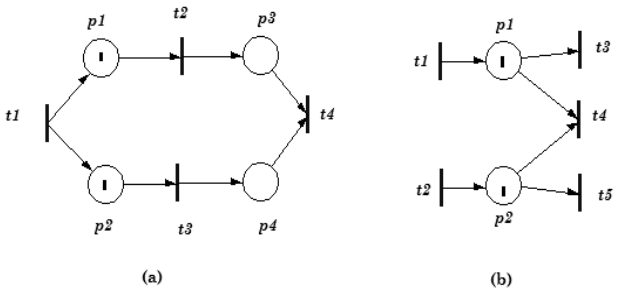

Fig. 4. Petri nets for representation of parallel activities (a) and symmetric confusion (b).

C. Properties of Petri Nets

There are two types of properties which can be studied with a Petri Net model, namely, behavioral and structural [19]. Here only behavioral properties are discussed.

Behavioral properties are reachability, boundedness, liveness, reversibility and home state, coverability, persistence, synchronic distance and fairness [19]. Just a part of them, i.e., those which are necessary for integration with the TFM are discussed in brief below:

- *Reachability* is a property, when each sequence of firings will result in a sequence of markings.
- *Liveness* is related to the complete absence of deadlocks in the system. A PN is said to be live if it is possible to ultimately fire any transition of the net by progressing through some further firing sequence. Thus, a live PN guarantees deadlock-free operation.
- *Reversability and Home State.* A PN is said to be reversible, if one can always get back to the initial marking or state. In many applications, it is important not to get to the initial state, but to the home state. A marking *M'* is said to be a home state, if for each marking *M* in the set of all possible markings reachable from the initial marking *M*0, *M'* is reachable from *M*.
- *Persistence* is a property, when, for any two enabled transitions, the firing of one transition will not disable the other. A transition in a persistent net, once it is enabled, will stay enabled until it fires. It could be called a conflict-free net.

IV. Integration of TFM and CPN Formalisms

The TFM is used as a holistic business model of the system, while the CPN considers the system at a more detailed level of abstraction. The weakness of the TFM is a lack of mechanisms for its verification. The only mechanism that exists is domain expert reviews. The strong point of the CPNs is a mechanism for modeling and analysis of different functional aspects of the system, where the most interesting for this research is the transition of data within the system operation. The research goal is transformation of the TFM into the CPN and analysis of its correctness by using native mechanisms.

A. The TFM and the CPN Models

The one common weakness of all models based on graphs, including TFMs and PNs, is the size and complexity of graphs. For PNs, this weakness hardens analysis of behavioral properties, especially reachability. Therefore, CPNs use the modularization principle, where a module (called a page) represents a part of the model and can be analyzed faster. The question is what construct of the TFM can be related to a page?

The TFM represents the entire system under discourse holistically. Even in case of a middle-sized system the graphical representation, digraph, is very large and hardly reviewed. In order to handle this weakness, the TFM supports two modularization principles – hierarchical levels of abstraction and decomposition:

- Hierarchical levels of abstraction. The TFM uses simplification and refinement mechanisms between topological spaces, namely, continuous mapping, in such a way supporting creation of hierarchical levels of abstraction [7], [2], [13]. In this case, the model remains holistic, and a specific FFs can be transformed to pages, keeping all cause-effect relations among abstracted FFs as relations among corresponding pages.
- Decomposition. Another principle is decomposition of the TFM into a set of business processes or use cases by using business goals as criteria for the decomposition [12]–[14]. Then, each use case or business process can be transformed to the CPN model and analyzed. In this case, causal dependencies among use cases or business processes must be additionally analyzed.

In general, the process of analysis of CPN models can be separated in two parts: first, to perform the analysis of each module of the system, and second, to analyze co-work of modules. Thus: 1) the TFM must be refined till the level, when

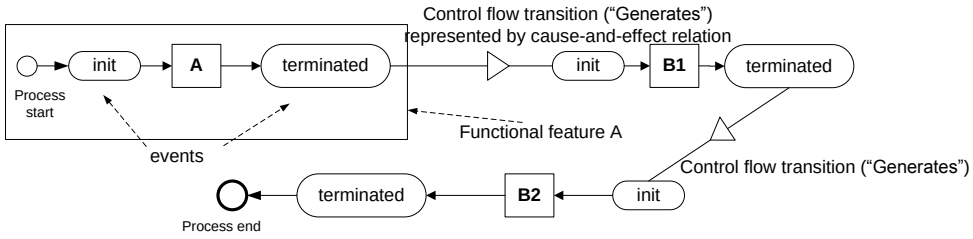

Fig. 5. The initiation and conduction of FF.

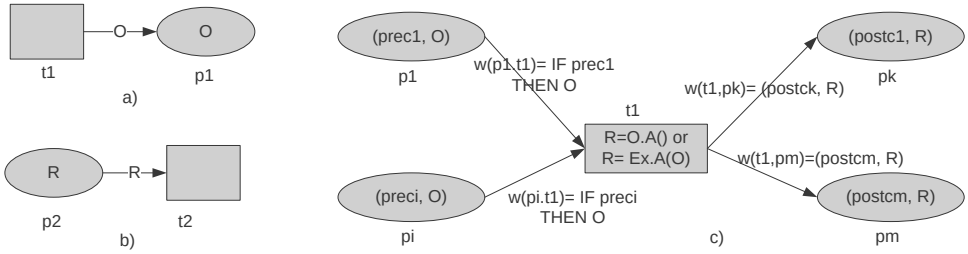

Fig. 6. Presentation of TFM elements with CPN constructs.

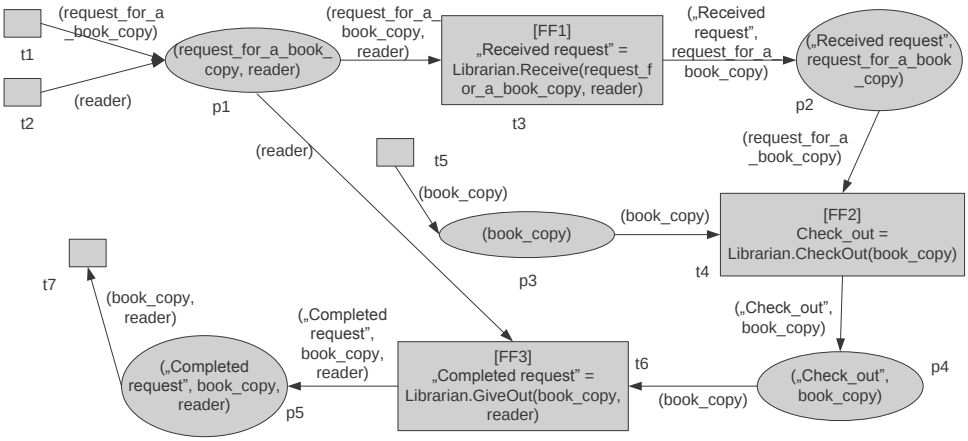

Fig. 7. The example of CP Net mapped from the TFM FFs.

every FF is specialized, i.e., it cannot be refined further from the desired viewpoint; 2) Then, modules must be defined within the TFM. This could be done by using decomposition criteria – business or system goals. For each goal, a set of FFs and binary cause-effect relations among them is assigned. At this step, shared functional parts can be determined and separated as particular modules. After that, these modules can be simplified and represented within a single model. This model should specify causal dependencies among modules; 3) Then, each part and the entire model should be transformed to the CPN model. This will give the possibility to analyze the behavioral properties of the whole system and its parts.

B. Presentation of Functional Features and Cause-effect Relations by means of the Petri Net Theory

As previously mentioned, the TFM elements are: 1) an FF that could be input, output and intermediate, and 2) a cause-effect relation that denotes causal dependency of an effect from a cause. In its turn, CPNs elements are places, colors, tokens, transitions, arcs and arc weights. Mappings from the TFM to the CPN are defined below.

The initialization and conduction ofthe FF in terms of business process execution is illustrates in Fig. 5. Conduction of FF begins with the initiation event "init". Then, FF is conducted. After that a termination event "terminated" is raised, which generates a transition to the next FF. In dependency of the termination success, the next FF is or is not initiated [13].

As mentioned in Section II.C, the FF is a unique 9-tuple <A, **R**, O, **PrCond**, **PostCond**, **Pr**, **Ex**, InRel, OutRel>. A domain object or a set of them (O) and the results (**R**) as well as entities **Pr** and **Ex** are denoted as token colors. Since FF performs action A, it must be represented as a transition. The initiation of the FF in terms of PN theory is the enabling of the transition. The conduction of FF is the firing of the transition. Thus, transition *t* denotes such action A, which is performed by one object O (or one entity from **Ex**) of the FF. CPNs allow representing arc weights as conditions and corresponding transited token colors in the general form "IF precondition THEN (<NumberOfTokens, TokenColor>)" (<NumberOfTokens> with value 1 could be skipped in the description of the weight). Therefore, weights of those arcs, which come from place *p* to transition *t*, should be presented as an expression "$w(p, t)$ = IF $prec_i$ Then O". In turn, weights of those arcs, which come from transition *t* to place *p*, should be presented as an expression "$w(t, p) = (postc_i, \mathbf{R})$". **R** of a FF is a result of the conduction of action A. Post-conditions **PostCond** represent the system states after firing transition *t*.

In the general case, an intermediate FF (i.e., that is not input and output) represents a part of system's inner functionality. It is generated by a cause or combination of causes and generates an effect or combination of effects. For this case, CPNs suggest using a combination of places and transitions as presented in Fig. 6(c) and explained above.

An input FF represents a part of functionality that provides an unlimited set of input resources or state changes in the system, for example, in case of FF "arriving of a person" represents an unlimited number of persons, who can activate the system as an input signal. During constructing the CPN model, an input FF is represented by PN source transition and one place, where an arc is directed from the transition to the place (Fig. 6(a)).

An output FF represents a part of functionality that triggers an output event and state changes. For example, it could be a resource that is produced by the system and transited to the external environment. For this case, CPNs suggest using a place that transfers a set of tokens to a PN sink transition, which does not produce any token (Fig. 6(b)).

Within the CPN model, cause-effect relation representation is not possible as such, because it is a causal relation between two transitions. CPNs do not allow creating arcs directly between two transitions. However, cause-effect relations dictate sequence and enabling of transitions. In general, the termination event of one FF (the cause) and possibility of initiation of another FF (the effect) occurs when the system has one and the same state (Fig. 5); therefore, the combination of places and incoming and outgoing arcs between two transitions could be considered representation of the causal dependency.

The example of the Petri Net that illustrates the provided mappings is shown in Fig. 7. It is a representation of three FFs: FF2 "Checking out a book_copy" by Librarian that is generated by FF1 "Receiving the request_for_a_book_copy from a reader" by Librarian and generates FF3 "Giving the book_copy to a reader" by Librarian.

C. Analysis of PN Behavioral Properties for Verification of Resources in the TFM

Verification of the reachability and liveness of PNs could help in verification of construction of the system functionality in the TFM, since it allows discovering deadlocks and those system states, which are not reachable ever. In terms of the TFM, it means that there is no such functional part, which could not be ever executed or which leads to the unstable operation of the system. Apart from that, this means that each output FF in the TFM can be reached under certain conditions, thus providing proper communication with the external environment of the system. By verifying these properties, it is possible to discover inconsistencies in resource specification in the TFM. The fact that a place is not reachable or a transition cannot be ever fired means that transition of tokens is not specified correctly, i.e., some FFs are not specified or their specification is not complete.

Concepts of reversibility and home state in PNs could be attributed to the concept of functioning cycle structure in TFMs. The functioning cycle is a very important concept in the TFM. The presence of cycles distinguishes a TFM from a simple digraph. The functioning cycle is a chain of those FFs connected by cause-effect relations, which are vital for system (subsystem) functioning. However, the concept of home state is more suitable for this purpose than reversibility, since the TFM may have many cycles of different orders.

And the last concept, persistence, allows verification of absence of conflicts. Apart from that, all marked graphs are persistent [19].

V. Related Work

The advantages of ordinary PNs and their extensions are well-known for academic researchers. There are many studies on transformation of process models to Petri net models in order to analyze them further.

The author in [27] suggests an iterative procedure model for the management of business process. In the beginning, the informal model is constructed in Event-driven Process Chain (EPC) language. Then, through the suggested procedure this model is transformed to a formal PN model, which is used for analysis of the constructed EPC model, and finally serves as the base for coordination of the process at the run time. The idea is similar, i.e., support of the business process modeling at the very beginning of this process.

Another interesting approach is presented in [28], where the authors use causal PNs. These nets are similar to the WF nets mentioned above. The authors suggest the VIP approach for modeling and validating business process specifications by simulation, especially their performance characteristics.

Authors in [29] verify a web-based system on problem-based learning by defining a new class of PNs, namely, Activity Flow Nets. The main idea is to transform UML activity diagrams to this kind of PNs and verify correctness of activities and control flows. However, the verification mostly is based on reviews of the constructed models.

Another very interesting approach is given in [30], where the authors use a similar idea for verification of workflows by using Work Flow Colored Petri nets, an extension of WF nets by CPN elements. It is necessary for the analysis of resource flows within the processes and scheduling the processes themselves.

In more recent studies, authors suggest the application of ordinary, timed and stochastic Petri nets for web-service construction, modeling and orchestration [31]–[33], etc.

VI. Conclusion

This research investigates the possibility of integration of two formalisms, TFM and PNs. The necessity of such integration is based on the fact that checking the TFM requires human activity by now. However, it is hard in case of large or complex graphs. Therefore, automated assistance would be valuable. This could be provided by the properties of Petri Nets. The common point of both formalisms is the fact that they specify functionality of the system. However, the TFM does it at the computation-independent level (a set of process instances level), while the PNs at a more specialized – execution level (a process instance level). This makes it possible to verify the TFM at the execution-level by PNs mechanisms.

The main idea suggested here is common for all the related works, i.e., modeling of the problem domain by means of the suitable modeling language and technique, and then transformation of the constructed model into a model of some kind of PNs for structural and behavioral property analysis. This research focuses on the transformation from the TFM into WF CPNs. The main result is the set of mapping rules from TFM to PN elements (however, informally described). The analysis of the obtained PN model can give a list of errors on the resource transferring and specification within the system. Therefore, the necessity of improvement of the functional description in the original TFM now is based on the facts, not only on developer's mental speculations.

There are also some limitations. The first is a complex check of semantic validity of the constructed model by using only PNs; but here domain ontology can help. The second is the complexity of created models. As discussed in this paper, the use of different modularization principles can handle this issue. And the last one is a lack of automated transformation from TFM to PN. The proposed idea is planned to be verified practically in order to assess method applicability. The last one is the direction of further research.

References

[1] OMG. MDA Guide Version 1.0. OMG, 2003. [Online]. Available: www.omg.org, [Accessed: Sept. 10, 2013].

[2] E. Asnina and J. Osis, "Computation independent models: bridging problem and solution domains," in *Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development MDA & MTDD 2010, In conjunction with ENASE 2010*, Athens, Greece, July 2010. Portugal: SciTePress, 2010.

[3] J. Osis and E. Asnina, "A Business Model to Make Software Development Less Intuitive," in *Proceedings of 2008 International Conference on Innovation in Sofware Engineering (ISE 2008).* Dec. 10–12, 2008, Vienna, Austria. IEEE Computer Society Publishing, 2008.

[4] J. Osis and E. Asnina, "Is Modeling a Treatment for the Weakness of Software Engineering?" in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. Hershey – New York: IGI Global, 2011, pp. 1–14. http://dx.doi.org/10.4018/978-1-61692-874-2.ch001

[5] E. Asnina and J. Osis, "Topological Functioning Model as a CIM-Business Model," in *Model-Driven Domain Analysis and Software Development: Architectures and Functions.* Hershey, New York, USA: IGI Global, 2011, pp. 40–64. http://dx.doi.org/10.4018/978-1-61692-874-2.ch003

[6] J. Osis and E. Asnina, "Derivation of Use Cases from the Topological Computation Independent Business Model," in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. Hershey, New York: USA, IGI Global, 2011, pp. 65–89. http://dx.doi.org/10.4018/978-1-61692-874-2.ch004

[7] J. Osis, "Topological Model of System Functioning," in *Automatics and Computer Science*, J. of Acad. of Sc., no. 6, 1969, pp. 44–50.

[8] J. Osis, "Formal Computation Independent Model within the MDA Life Cycle," in *International transactions on system science and applications*, 2006, pp. 159–166.

[9] J. Osis, "Software development with topological model in the framework of MDA," in *Proceedings of the 9th CaiSE/IFIP8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CaiSE'2004*. Riga, Latvia: RTU, 2004.

[10] J. Osis and E. Asnina, "Topological Modeling for Model-Driven Domain Analysis and Software Development," in *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. Hershey, New York: USA, IGI Global, 2011, pp. 15–39. http://dx.doi.org/10.4018/978-1-61692-874-2.ch002

[11] W. F. Basener, *Topology and Its Applications*. New Jersey: John Wiley and Sons, Inc., 2006. http://dx.doi.org/10.1002/9780470067949

[12] J. Osis, E. Asnina and A. Grave, "Formal Problem Domain Modeling within MDA," in *Communications in Computer and Information Science (CCIS). Software and Data Technologies*. Berlin: Springer-Verlag, 2008, pp. 387–398. http://dx.doi.org/10.1007/978-3-540-88655-6_29

[13] J. Osis and E. Asnina, *Model-Driven Domain Analysis and Software Development: Architectures and Functions*. Hershey, New York, USA: IGI Global, 2011. http://dx.doi.org/10.4018/978-1-61692-874-2

[14] U. Donins, J. Osis, A. Slihte, E. Asnina, and B. Gulbis, "Towards the Refinement of Topological Class Diagram as a Platform Independent Model," in *Model-Driven Architecture and Modeling-Driven Software Development: ENASE 2011, 3rd Whs. MDA&MDSD*, 2011, pp. 79–88.

[15] J. Osis, E. Asnina, A.Grave, "Formal Computation Independent Model of the Problem Domain within the MDA," in *Information Systems and Formal Models, Proceedings of the 10th International Conference ISIM'07*. Opava, Czech Republic: Silesian University, pp. 47–54, 2007.

[16] J. Osis, E. Asnina, A. Grave, "MDA Oriented Computation Independent Modeling of the Problem Domain," in *Proceedings of the 2nd International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2007)*. Barcelona, Spain, pp. 66-71, 2007.

[17] J. Osis and L. Beghi, "Topological modelling of biological systems," in D. A. Linkens and E. R. Carson (eds) *Proceedings of the third IFAC Symposium on Modelling and Control in Biomedical Systems (Including Biological Systems)*. Oxford, UK: Elsevier Science Publishing, 1997.

[18] E. Asnina, J. Osis, and A. Jansone, "System Thinking for Formal Analysis of Domain Functioning in the Computation Independent Model," in *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012), Poland, Wrocław,* 29–30. June, 2012. Portugal: Insticc, 2012.

[19] T. Murata, "Petri Nets: Properties, Analysis and Applications," in *Proceedings of the IEEE 77*, no. 4, 1989, pp. 541–580. http://dx.doi.org/10.1109/5.24143

[20] L. Kristensen M. S. Christensen and K. Jensen, "The practitioners' guide to coloured Petri nets," in *Int Journal on Software Tools for Technology Transfer*, issue 2. Springer-Verlag, 1998, pp. 98–132. http://dx.doi.org/10.1007/s100090050021

[21] K. Jensen, "An Introduction to the Theoretical Aspects of Coloured Petri Nets," in *A Decade of Concurrency, Lecture Notes in Computer Science*, vol. 803. Springer-Verlag, 1994, pp. 230–272.

[22] Ch. Lakos, "Object Oriented Modelling with Object Petri Nets," in *Concurrent Object-Oriented Programming and Petri Nets, Lecture Notes in Computer Science*, Vol. 2001. Springer, 2011, pp. 1–37. http://dx.doi.org/10.1007/3-540-45397-0_1

[23] L. M. Kristensen, J. B. Jørgensen and K. Jensen, "Application of Coloured Petri Nets in System Development," in *ACPN 2003, LNCS*, 3098, edited by J. Desel, W. Reisig and G. Rozenberg. Springer-Verlag Berlin Heidelberg, 2004, pp. 626–685. http://dx.doi.org/10.1007/978-3-540-27755-2_18

[24] AIS Group. CPN Tools Homepage. [Online]. Available: http://cpntools.org/ [Accessed: Jul. 10, 2013].

[25] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management," in *The Journal of Circuits, Systems and Computers*, vol. 8, no. 1, 1998, pp. 21–66.

[26] L., Dongsheng, J. Wang, S. C. F. Chan, J. Sun, L. Zhang, "Modeling workflow process with colored Petri nets," in *Computers in Industry* vol. 49, no. 3, 2002, pp. 267–281. http://dx.doi.org/10.1016/S0166-3615(02)00099-4

[27] J. Dehnert, "Four Steps Towards Sound Business Process Models, " in *Petri Net Technology for Communication-Based Systems: Advances in Petri Nets*, edited by Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg and Herbert Weber, vol. LNCS 2472. Germany: Springer-Verlag Berlin Heidelberg, 2003, pp. 66–82. http://dx.doi.org/10.1007/978-3-540-40022-6_4

[28] J. Desel and E. Thomas, "Quantitative Engineering of Business Processes with VIPbusiness," in *Petri Net Technology for Communication-Based Systems: Advances in Petri Nets*, Vol. LNCS 2472, edited by Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg and Herbert Weber. Gernamy: Springer-Verlag Berlin Heidelberg, 2003, pp. 219–242. http://dx.doi.org/10.1007/978-3-540-40022-6_11

[29] V. R. L. Shen, W. Yu-Ying, Y. Cheng-Ying, and Y. Szu-Tso, "Verification of problem-based learning systems using modified petri nets," in *Expert Systems with Applications* vol. 39, no. 16 (November 2012), pp. 12636–12649. http://dx.doi.org/10.1016/j.eswa.2012.05.019

[30] Zh. Xiao, and M. Zhong, "A method of workflow scheduling based on colored Petri nets," in *Data & Knowledge Engineering* vol. 70, no. 2 (February 2011), pp. 230–247. http://dx.doi.org/10.1016/j.datak.2010.11.005

[31] Y. Yi, "An Extended Stochastic Petri Nets Modeling Method for Collaborative Workflow Process," in *Physics Procedia* vol. 33, 2012, pp. 1547–1552. http://dx.doi.org/10.1016/j.phpro.2012.05.251

[32] V. Valentín, H. Macià, J. J. Pardo, M. E. Cambronero, and G. Díaz, "Transforming Web Services Choreographies with priorities and time constraints into prioritized-time colored Petri nets," in *Science of Computer Programming*, vol. 77, no. 3 (March 2012), 2012, pp. 290–313. http://dx.doi.org/10.1016/j.scico.2011.05.002

[33] S. Chemaa, F. Bachtarzi, and A. Chaoui, "A High-level Petri Net Based Approach for Modeling and Composition of Web Services," in *Procedia Computer Science* 9, 2012, pp. 469–478. http://dx.doi.org/10.1016/j.procs.2012.04.050

Erika Asnina received M. Sc. in Computer Systems in 2003 and a Doctoral Degree (Dr. sc. ing.) in Information Technology with specialization in system analysis, modeling and design in 2006 from Riga Technical University.

She has been an Associate Professor at the Department of Applied Computer Science at Riga Technical University since 2013. She also worked as a Software Developer. She is an author of 30 conference papers, 4 book chapters and 1 book. Her research interests include software quality assurance, model-driven and object-oriented software engineering.

The Latvian Academy of Sciences awarded her and the co-author, Janis Osis, for the book "Model-Driven Software Development: Architectures and Functions" in 2011. She was also awarded as a scholarship laureate of the target program "For Education, Science and Culture" by the Latvian Education Fund in 2004 and 2005.

Address: Department of Applied Computer Science, Riga Technical University, Meža Str. 1/.3, Riga, LV-1048, Latvia;

E-mail: erika.asnina@rtu.lv

Begoña Cristina Pelayo García-Bustelo is a Lecturer at the Computer Science Department of the University of Oviedo. She obtained a Doctoral Degree in Computer Engineering from the University of Oviedo. Her research interests include object-oriented technology, Web engineering, eGovernment, modeling software with BPM, DSL and MDA.

Address: Computer Science Department, University of Oviedo, Edificio de la Facultad de Ciencias. C/ Calvo Sotelo s/n. 33007 Oviedo (Asturias, España);

E-mail: crispelayo@uniovi.es