Analyzing Business Process Architectures

Rami-Habib Eid-Sabbagh and Mathias Weske

Hasso Plattner Institute at the University of Potsdam
{rami.eidsabbagh,mathias.weske}@hpi.uni-potsdam.de

Abstract. In recent years, Business Process Management has gained maturity in private and public organizations. Organization own large process collections. Organizing, analyzing, and managing them becomes more complex. In the course of this development, research on Business Process Architectures has gotten more attention over the last decade. A Business Process Architecture describes the relationships between business processes within a process collections as well as the guidelines to organize them. However, formalization and verification techniques are still missing in this context. To overcome this gap we propose a novel Petri net based Business Process Architecture formalization. Based on this, we can resort to known Petri net verification techniques for the analysis of Business Process Architectures patterns and anti-patterns in regard to their structural and behavioral properties. Our methodology is evaluated on a real use case from the public administration.

Keywords: business process architecture, analysis, formalization.

1 Introduction

Business Process Management (BPM) has become an integral part of modern organizations and public administrations. In the course of constant improvement efforts, large process collections have been accumulated in companies. Managing all these processes is a difficult task. Business Process Architectures (BPA) provide guidelines for organizing business processes within a process collection and relate them along aspects of interest, e.g. goals, functions, or objects. A variety of approaches have been proposed in literature [1]. However, most of them try to assure consistent process architectures only on a high level of abstraction, focusing on single processes only and ignoring their interdependencies. Taking a holistic view at the interdependencies of business processes is a major necessity to assure correct business process collaboration [2,3,4].

An example from the public sector[1] shows the importance of analyzing business process interdependencies for undesired behavior. The founding of a new enterprise consists of performing many public services, of which a selection is depicted as EPC models in Fig. 1. Each public service for itself seems to result in a desired outcome. It is not visible that they depend on message flows or the

[1] EU Services Directive Realization in Berlin – https://www.ea.berlin.de/web/guest/home

C. Salinesi, M.C. Norrie, and O. Pastor (Eds.): CAiSE 2013, LNCS 7908, pp. 208–223, 2013.

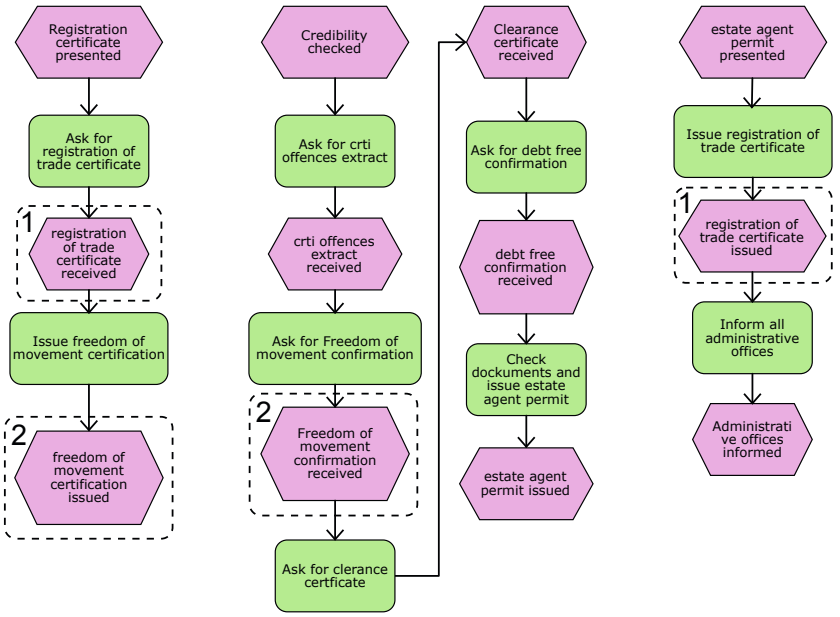

Fig. 1. EPC Business Process Models depicting three public services

outcome of other business processes, highlighted by the dashed boxes in Fig. 1. From this perspective, we cannot say anything about the correctness of their interaction, e.g. the successful founding of a new enterprise.

In [5], we proposed a new conceptual framework for the design of BPAs that describes business process interdependencies and provides a pattern based approach to examine them. 30 BPA patterns were identified. The presented BPA framework and its patterns, however, lack clear behavioral semantics. Hence, it is very complex and tedious to analyze BPAs and detect errors with proposed patterns. The patterns are limited to analyze direct pairwise interdependencies between two processes at a time. Undesired business process interdependencies that spawn over three or more processes cannot be detected.

To avoid ambiguities in interpretations, we transform BPAs to Trigger-Flow nets, a structural subclass of Petri nets that have clear semantics [6,7,8,9,10]. This transformation also allows for analyzing BPAs with known Petri net analysis techniques for their structural and behavioral properties.

The remainder of this paper is structured as follows, Section 2 introduces our assumptions, the definitions of Business Process Architectures and Trigger-Flow nets. Section 3 describes the transformation of BPAs and their process interdependencies into Trigger-Flow nets. In Section 4 we transform and analyze BPA patterns for their structural and behavioral properties, and categorize them. An evaluation of our approach with a real use case from the public administration is presented in Section 5. Section 6 embeds our approach into current research followed by the conclusion in Section 7.

2 Foundations

The transformation of BPAs into Trigger-Flow nets builds on the definition of BPAs in [5] which will be extended by a temporal order of events in processes. Events depict a business process's external interaction with other business processes, e.g. the event order received . Also other business process elements that are involved in business process interaction like sending activities can be reflected in events, e.g. the event notification sent. Hence, business processes are reduced to a set of events that occur in a sequence. The definition of a Business Process Architecture is as follows:

Definition 1 (Business Process Architecture). *A Business Process Architecture is a tuple* (E, V, L, I)*, in which:*

- E *is a set of events, partitioned into start events,* E^S*, end events* E^E*, intermediate throwing events* E^T *and intermediate catching events* E^C*.*
- V *is a partition of* E *and represents a set of business processes and each member represents a business process.*
- $v \in V$ *is a sequence of events,* $v =< e_1, ..., e_n >$ *such that* $e_1 \in E^S$ *is a start event,* $e_i \in E^C \cup E^T$ *for* $1 < i < n$ *any number of intermediate events, and* $e_n \in E^E$ *an end event*
- $\forall v \in V : \forall e_s \in E^S \cap v, \forall e_i E^T \cup E^C \cap v, \forall e_e E^E \cap v$ *must hold* $e_s \lessdot e_i \lessdot e_e$*.*
- $L \subseteq E \times E$ *is the flow relation, partitioned into synchronous flows* L^S *and asynchronous flows* L^A*.*
- $I \subseteq E \times E$ *is the trigger relation, partitioned into synchronous triggers* I^S *and asynchronous triggers* I^A*.*

Notice that we use v instead of V_n to refer to single processes of a BPA. The set $\bullet e = \{e' \in E^E \cup E^T | (e', e) \in I \vee (e', e) \in L\}$ contains the events with an outgoing relation to $e \in E$. The set $e\bullet = \{e' \in E^S \cup E^C | (e, e') \in I \vee (e, e') \in L\}$ consists of the events with an incoming relation from $e \in E$ [5].

A BPA is a set of business processes and their relations with each other. It exposes the complex business process interdependencies and interactions within large process collections which are reflected in their trigger and flow relations. Processes, that are in a trigger or information flow relation with each other, can only be triggered or get a message by their relation partners. They do not get triggered or get an information flow from the external environment. We assume for the transformation from BPAs to Trigger-Flow nets that business processes only have one start, one end event, and any number of intermediate events in regard to structural composition. For behavioral aspects, we assume that a start event always occurs before all intermediate events of a process, and the end event after all intermediates events have occurred, respectively. Intermediate events of a process must occur, and occur only once. Events of one process cannot occur in parallel. In graphical BPA representations events occur in temporal order of the reading direction from left to right as in Fig. 2. This means that an intermediate event e_i will happen before the intermediate event e_{i+1} of the same process if intermediate event e_i is placed left from the other intermediate event e_{i+1}.

For the definition of Trigger-Flow nets, we consider the standard Petri net definition [6,10] and extend it by defining internal, incoming and outgoing places following Martens [3]. Trigger-Flow nets are a structural subclass of Petri nets and defined as follows.

Definition 2 (Trigger-Flow net).
A Trigger-Flow net is a tuple $N = (P, T, F, M, M_0)$*, in which:*
- $P = \{p_1, p_2, ..., p_m\}$ *is a finite set of places*
- P *is partitioned into pairwise disjoint sets of internal places* P^N*, incoming places* P^I*, and outgoing places* P^O
- $T = \{t_1, t_2, ..., t_n\}$ *is a finite set of transitions*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a finite set of arcs depicting a flow relation*
- $P \cap T = \emptyset$ *and* $P \cup T \neq \emptyset$
- $M : P \to \mathbb{N}$ *denotes the marking of a Trigger-Flow net* (P, T, F) *mapping the set of places onto natural numbers including* 0
- M_0 *is the initial marking of the Trigger-Flow net* (P, T, F, M, M_0)

For $X = P \cup T$ we denote the preset of a node $x \in X$ as $\bullet x = \{x' \in X | (x', x) \in F\}$ and the postset of a node $x \in X$ as $x\bullet = \{x' \in X | (x, x') \in F\}$. A transition t is enabled to fire if all its input places are marked with a token. Places can hold several tokens at a time. In the following, we consider above Trigger-Flow net (TF net) definition. Based on presented concepts, we introduce the transformation from BPA into TF nets in the following section.

3 Mapping Business Process Architecture to Trigger-Flow nets

This section presents the formalization of BPAs by transforming them to TF nets. First, we present the transformation of BPA processes into single TF nets. Then, we define TF net composition rules to map the trigger and information flow concepts of BPAs onto TF nets.

3.1 Formalization of Single BPA Process Behavior

A Business process is a causally (temporally) ordered set of events $v =< e_1, e_2, ..., e_n >$ with start event $e_1 \in E^S$, end event $e_n \in E^E$, and with events $e_2, ..., e_{n-1}$ being either throwing intermediate events $\in E^T$ or catching intermediate events $\in E^C$.

Definition 3 (BPA Process Transformation Formalization). *Let* $v =< e_1, e_2, ..., e_n >$ *be a sequence of events that describes a business process in a Business Process Architecture and let* $N = (P, T, F, M, M_0)$ *be a Trigger-Flow net. The transformation from a BPA process into a Trigger-Flow net is defined as follows:*
- $T^V = \{t_{e_i} | e_i \in v\}$
- $P^N = \{p_{e_i} | e_i \in v \wedge 1 \leq i < n\}$

- $P^O = \{p_o\} \cup \{p'_{e_i} | e_i \in E^T \cap v\}$
- $P^I = \{p_i\} \cup \{p''_{e_i} | e_i \in E^C \cap v\}$
- $P = P^N \cup P^O \cup P^I$
- $F = \{(p_i, t_{e_1}), (t_{e_n}, p_o)\} \cup \{(t_{e_i}, p_{e_i}) | t_{e_i} \in T \wedge p_{e_i} \in P^N\} \cup \{(p_{e_i}, t_{e_{i+1}}) | t_{e_{i+1}} \in T \wedge p_{e_i} \in P^N\} \cup \{(t_{e_i}, p'_{e_i}) | t_{e_i} \in T \wedge p'_{e_i} \in P^O\} \cup \{(p''_{e_i}, t_{e_i}) | t_{e_i} \in T \wedge p''_{e_i} \in P^I\}$
- $M_0 = [p_i]$

Fig. 2 shows an exemplary transformation of a BPA process p with events s, t, c, e into a TF net. Each event of that process is transformed into a specific TF net construct as depicted in Fig. 2. In a second step the different TF net constructs are merged to one TF net.

A **start event** s of a process p is transformed into a TF net transition t_{e_1} with one incoming initial place and one internal place. The incoming place is the initial place p_i of the latter resulting TF net and serves as input place of the transition t_{e_1}. The internal place p_{e_1} is the output place of transition t_{e_1}. The initial place p_i is marked wit a token.

Each **intermediate throwing event** t is transformed into a transition t_{e_i} with two internal places $p_{e_{i-1}}$ and p_{e_i}, and one outgoing place p'_{e_i}. In Fig. 2 this is depicted by the transformation of event t into the transition t_{e_2} with internal place p_{e_1} as its input place, and the internal place p_{e_2}, and outgoing place p'_{e_2} as its output places. The outgoing place p'_{e_2} depicts that the TF net needs to be connected with a communication partner.

Similarly each **intermediate catching event** c is transformed into a transition t_{e_i} that is connected to two internal places $p_{e_{i-1}}$ and p_{e_i}, and one incoming place p''_{e_i}. The places p_{e_2} and p''_{e_3} are the input places of the transition t_{e_3}, and p_{e_3} serves as its output place. Places p_{e_2} and p''_{e_3} show that the transition t_{e_3} waits for input from another process but also needs to be ready internally to fire.

An **end event** e is transformed into a transition t_{e_n} that is connected to an internal place $p_{e_{n-1}}$ and one outgoing place p_o. Place p_o also marks the end place of latter TF net. This is represented in Fig.2 by the TF net construct consisting of one input place p_{e_3}, transition t_{e_4}, and output place p_o. The end place p_o is at the same time an outgoing place that may provide input to other processes.

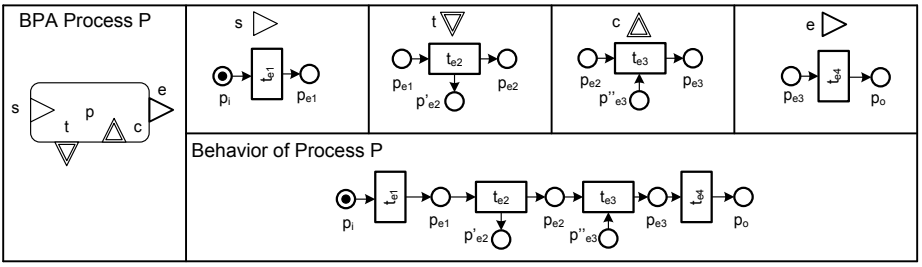

Fig. 2. Transformation of BPA process events into Trigger-Flow net structure

To connect the resulting TF net constructs their internal output and input places are merged. E.g., in Fig. 2 the output place p_{e_1} of transition t_{e_1} and the input place of transition t_{e_2} are merged as they are identical. The same procedure is repeated for the internal output and input places of transition t_{e_2},t_{e_3} and t_{e_4}. The result is a TF net that represents the behavior of the BPA process p. In contrast to the BPA process representation, the TF-net has a clear behavioral semantics. It is obvious that first transition t_{e_1} fires, followed by transition t_{e_2}. Transition t_{e_3} can only fire if it also gets input from another process. If transition t_{e_3} gets an input it can fire, else the TF net is in a deadlock. After t_{e_3} has fired, transition t_{e_4} is enabled and can fire ending the process. In regard to structural composition, it is immediately apparent that the places p'_{e_2} and p''_{e_3} lead to a structurally not sound TF net. The incoming and outgoing places hint at interdependencies with other processes.

3.2 Formalization of BPA Trigger and Flow Relationships in Trigger-Flow Nets

This section describes the composition rules for the transformed TF nets. They map the trigger and flow relations between business processes in a BPA to TF nets. In the following, we will refer to TF nets resulting from the transformation of the BPA processes p, q, and r as TF nets A, B and C, respectively. The composed TF net will be referred to as TF Net N. After transforming the BPA processes into TF nets, the resulting TF nets are connected through t_α transitions according to the trigger or flow relations defined in the BPA. The flow and trigger relations are defined as pairs of source and destination events $(e_s, e_d) \in I \cup L$, e.g. (e_1, s_2), e_1 being the source and s_2 the destination. The t_α transitions have always an outgoing place of the TF net of the source event as their input place, and an incoming place of the destination event's TF net as their output place. Fig. 3 depicts the composition rules in graphical representation.

Composition rule 1 maps the trigger relation $(e_1, s_2) \in I$ from an end event of process p to a start event of another process q. TF net A is connected via its end place and the t_α transition to the start place of TF net B. The marking is removed from the initial place of TF net B. The token will be passed on by TF net A when it triggers TF net B. We notice this kind of relation from end to start event results in a structurally and behaviorally sound TF net N.

Composition rule 2 describes trigger relation $(t_1, s_2) \in I$. Process p triggers process r through its intermediate event t_1. To represent this trigger relation the intermediate outgoing place of TF net A is connected through a t_α transition to the start place of TF net B. The marking is removed from the initial place of TF net B. TF net N has one marked initial place and two end places.

Composition rules 3 and 4 describe the information flow relation by sending and catching a message flow through intermediate events in rule three as well as by passing an information flow through an end event to an intermediate event in rule four, respectively. In rule three the flow relationship $(t_1, c_2) \in L$ is presented by the connection of the outgoing place of transition pt_{e_2} of TF net A via the t_α transition with the incoming place of transition qt_{e_2} of TF net B.

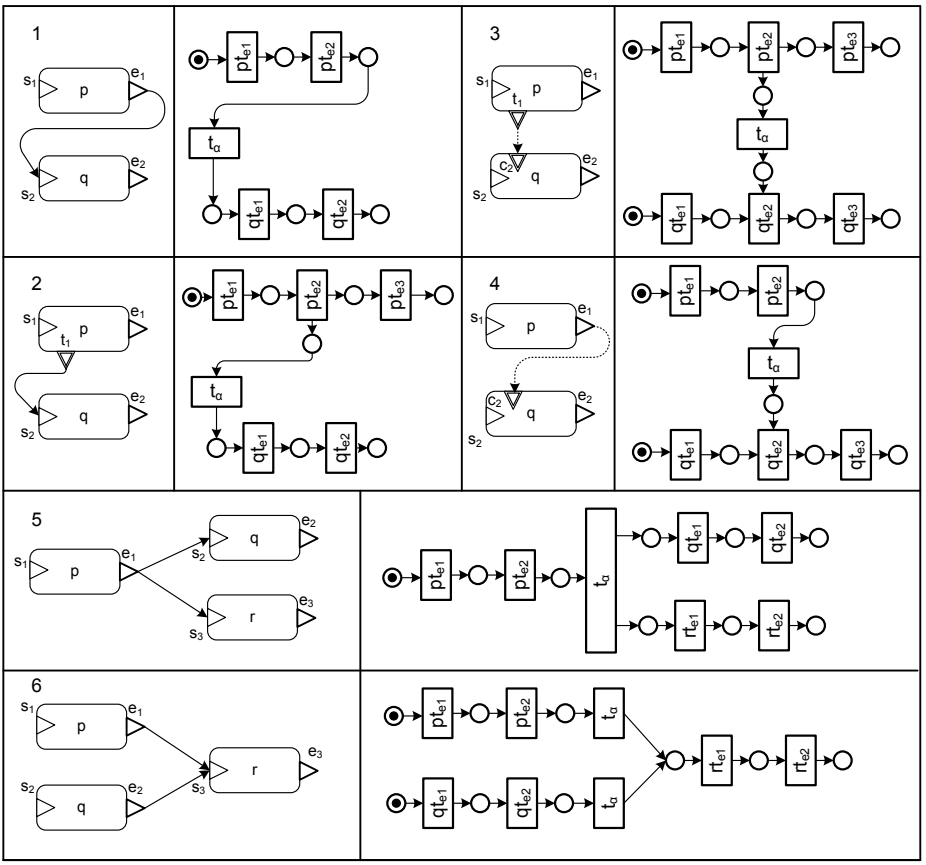

Fig. 3. TF net composition rules

The composed TF net N has two initial and end places. Composition rule four is similar, except from the source of the composition being the end place of TF net A. The t_α transition connects the end place of TF net A with a t_α transition to the incoming place of transition qt_{e_2} of TF net B. The composed TF net N has two initial places and only one end place. Note, that transitions pt_{e_1} and qt_{e_1} in rule 3, 4, and 6 are not synchronized and can fire independently.

Composition rule 5 defines the triggering or messaging of several processes by one process, i.e. a source event takes part in several trigger or flow relations. Depicted in Fig. 3, process p triggers process q as well as process r. The end event e_1 of process p takes part in two trigger relation pairs (e_1, s_2) and (e_1, s_3). In this case first TF net A will be connected to TF net B as described in rule one. To represent the second trigger relation only the initial place of TF net C has to be connected as second output place to the the transition t_α. Both initial markings from Petri B and C are removed. When transition t_α fires, it passes on tokens to two concurrent branches, former TF nets B and C.

Composition rule 6 describes the receiving of several triggers or flows, i.e. an event takes part as destination event in several trigger or flow relations. In Fig. 3, event s_3 takes part in $(e_1, s_3) \in I$ and $(e_2, s_3) \in I$. In this case, composition rule one is executed two times. A transition t_{α_1} and t_{α_2} is introduced for each relation. However, both share the same output place, the initial place of TF net C. The token is removed from the initial place of TF net C. TF net C can be triggered by TF net A or B or by both, one after the other. Rules two to six result in structurally not sound TF nets.

Following Martens [3], we compose two TF nets A and B to a new TF net N by introducing a new transition and connecting the outgoing and incoming places according to the flow or trigger relations defined in the BPA. In the following, we define the composition rules in a formal way.

Definition 4 (TF net Composition Rules). *Let $A = (P_a, T_a, F_a, M_{a_0})$ and $B = (P_b, T_b, F_b, M_{b_0})$ be two TF nets. Let T_α be a set of connector transitions and T_α, T_a, T_b pairwise disjoint. Let F_α the flows connecting the two TF nets A and B and F_α, F_a and F_b pairwise disjoint. The composed TF net $N = A \oplus B$ is represented by $N = (P_N, T_N, F_N, M_{N_0})$ such that:*

- $T = T_a \cup T_b \cup T_\alpha$
- $T_\alpha = \{t_{e_s} | (e_s, e_d) \in I \cup L \wedge (e_s, e_d) \in (v_a \times v_b) \cup (v_b \times v_a)\}$
- $P = P_a \cup P_b$
- $F = F_a \cup F_b \cup F_\alpha$
- $F_\alpha = \{(p_{e_s}, t_{e_s}) | p_{e_s} \in P^O, t_{e_s} \in T_\alpha \wedge (e_s, e_d) \in I \cup L \wedge e_s \in v_a \cup v_b\} \cup \{(t_{e_s}, p_{e_d}) | p_{e_d} \in P^I, t_{e_s} \in T_\alpha \wedge (e_s, e_d) \in I \cup L \wedge e_d \in v_a \cup v_b\}$
- $M_0 = [N_{p_i}]$

In general, after having transformed all BPA processes, all trigger and flow relations $(e_s, e_d) \in I \cup L$ of a Business Process Architecture are mapped to the transformed TF nets. For each event that participates as source in a trigger or flow relation in a BPA exactly one t_α transition and exactly one arc is inserted. This arc connects the outgoing place from the source event transition of the according relation as input place to the t_α transition. For each destination element in a relation an arc from the corresponding t_α transition to the incoming place of the destination event transition is drawn. All remaining initial places p_i of composed TF net that were not connected to a t_α transition are marked with a token. The outgoing places can either be the end place or the outgoing places of an intermediate event transition of a TF net. A TF net's incoming places are either its initial place or an incoming place of an intermediate transition. Inserting T_α transitions to describe the trigger and flow relationships allows us to model concurrent behavior when a process triggers several other processes as in rule 5 or the merging of two concurrent processes in a join structure as depicted in rule 6 in Fig. 3.

4 Business Process Architecture Pattern Properties

In [5], 30 BPA patterns and anti-patterns are proposed to identify desired and undesired structural and behavioral properties. These include different trigger

and flow patterns, e.g. send and receive, broadcast, and multi-instance patterns as well as dead event, loops or deadlock patterns as examples for anti-patterns. Ten of those patterns are depicted in Fig. 3 and Fig. 4. The patterns proposed in [5] provide a first means to detect errors between two processes in a Business Process Architecture. However, for large BPAs their application becomes complex. In order to find out more about their properties we examine the BPA patterns by applying our BPA to TF net transformation and categorize them in a later step.

In total we examined 23 patterns, of which 10 are considered regular patterns and 13 anti-patterns respectively. From the 30 original patterns, we excluded seven multi-instance and hierarchical patterns. Their transformation and analysis will be part of future work. For structural properties we looked at the number of input and output places as well as structural soundness. We considered different soundness criteria, boundedness and liveness as well as the number of deadlocks and dead events for the analysis of behavioral properties. A representative set of BPA patterns, depicted in Fig. 4, will serve as example to illustrate our approach. These patterns were selected as they cover most of the structural aspects and show the varied behavioral properties observed in BPA patterns.

Our course of action consisted of three steps, firstly the transformation of BPA patterns to TF nets, secondly the analysis of the TF nets for their structural and behavioral properties with LoLA[2], and thirdly the categorization of transformed BPA patterns. The transformation of BPA trigger and flow relations results into TF nets with different number of start and end places. If they were structurally not sound, we converted them to Workflow nets to achieve structural soundness [3]. In case a TF net had several initial places as in pattern 8 in Fig. 4 we inserted a new transition with one input place that uses the initial places of the TF net as its postset. Several end places (see pattern 8, 27 Fig. 4) of a TF net serve as pre-set of a newly inserted transition with only one output place. When the TF net lacked initial or end places, e.g. pattern 19, 24, 25 in Fig. 4, we inserted a transition with one input or output place respectively, and connected the transition to the according initial place or end place of one of the TF nets in the pre-composition state. The resulting Workflow nets were then examined with LoLA for their behavioral properties.

The examination of the BPA patterns shows interesting results. After the first step of our analysis, the transformation of BPA patterns into TF nets, most of the BPA patterns are not structural sound. Either they have several or no start, and several or no end places, or a combination of both. The results of the overall examination can be grouped into five categories.

The ***Sound TF Nets (SN)*** category includes structurally, behaviorally sound, and bounded TF nets. These are the TF nets that depict the regular patterns 1, 6, 7, 18, 23, and 28. In [5] patterns 23 and 28 are considered anti-patterns in environments of synchronous communication which does not apply for the asynchronous TF net environment.

[2] LoLA–A Low Level Petri Net Analyser `http://www.informatik.uni-rostock.de/tpp/lola/`

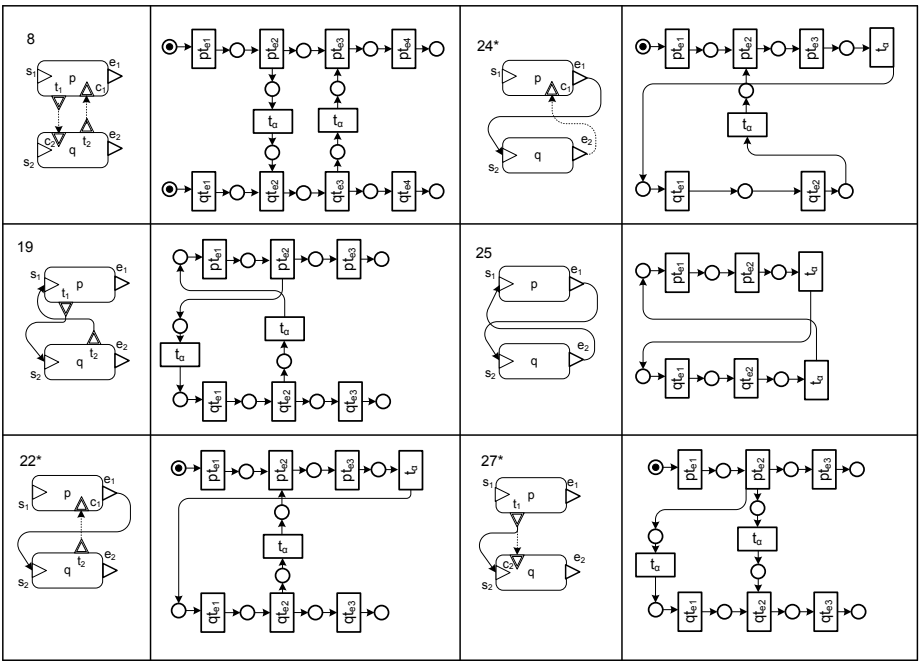

Fig. 4. Representative BPA Patterns Mapped to Trigger-Flow nets

2nd Iter. Sound TF Nets (2SN) After converting the TF nets depicting regular BPA patterns 2, 3, 4, 5, 8, 13 and anti-patterns 15, 25, 27 to Workflow nets, they become structurally, and behaviorally sound, and bounded TF nets.

Dead Nets (DN) Dead nets are TF nets that can never fire for lacking an initial place, e.g. pattern 19 in Fig. 4. The anti-patterns 15, 16, 19, 20, 25, and 26 are grouped into this category.

Producer Nets (ProN) Workflow nets in this category are unbounded and expose a livelock. They produce unlimited amounts of token. After being converted to Workflow nets, the TF nets depicting pattern 16, 19, 20, and 26 are moved from the dead nets category into this one.

Deadlocks (DL) This category includes TF nets that expose deadlocks. These are TF nets representing anti-patterns 17, 21, 22, and 24. Most of them have several dead transitions ranging from two to six. The TF net representing pattern 22, however, is structurally sound but does not become a regular pattern in contrast to the findings in [5]. The TF nets stay in this category, also after conversion to structural sound Workflow nets, as the deadlocks remain.

The results obtained from the analysis of BPA anti-patterns are shown in Table 1. As all regular patterns have the same properties and were grouped in the Sound Petri or 2nd Iteration Sound TF net category, they are not listed in the table. The verification showed that most of the BPA patterns and anti-patterns are structurally not sound. The regular BPA patterns represent desired behavior. Interestingly, some of the anti-patterns become behaviorally sound

Table 1. Properties of BPA patterns

Properties	**Patterns**													
	18	**23**	**28**	**27**	**15**	**25**	**16**	**20**	**26**	**19**	**22**	**17**	**24**	**21**
Start Place	1	1	1	1	0	0	0	0	0	0	1	1	1	1
End Place	1	1	1	2	0	0	1	1	1	2	1	0	0	2
Struct. Sound	yes	yes	yes	no	no	no	no	no	no	no	yes	no	no	no
WF net Sound	yes	yes	yes	no	no	no	no	no	no	no	no	no	no	no
Weak Sound	yes	yes	yes	no	no	no	no	no	no	no	no	no	no	no
Relaxed Sound	yes	yes	yes	no	no	no	no	no	no	no	no	no	no	no
Livelock	no	no	no	no	no	no	yes	yes	yes	yes	no	no	no	no
Bounded	yes	yes	yes	yes	yes	yes	no	no	no	no	yes	yes	yes	yes
Live	no	no	no	no	no	no	no	no	no	no	no	no	no	
Dead Trans.	0	0	0	0	2	4	3	5	4	6	5	2	4	6
Deadlock	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1*st* It. Categ.	SN	SN	SN	NSS	DN	DN	DN	DN	DN	DN	DL	DL	DL	DL
2*nd* It. Categ.	-	-	-	2SN	2SN	2SN	2SN	ProN	ProN	ProN	-	DL	DL	DL

after converting them to Workflow nets, e.g. pattern 15 and 25. Anti-patterns 18, 23, and 28 expose soundness from the beginning and are regular patterns in asynchronous communication environments. Revived dead nets resulted in producer nets that are stuck in a livelock, e.g. pattern 16, 19, or 20. Determining a sensible attachment position for an initial/end place for dead/live TF nets requires further research and will be looked at in detail in future work.

5 Evaluation

Pattern based approaches provide means for classifying process interactions and interdependencies according to their structural composition or behavioral interaction properties. The BPA patterns proposed in [5], of which ten were exemplary illustrated in Fig. 3 and Fig. 4, can be used to identify desired process composition or find irregularities between process pairs in a BPA. The approach is limited to direct interdependencies between two business processes only. Behavioral properties like boundedness, livelocks, deadlocks or dead nets cannot be observed over several business processes, i.e. indirect interdependencies cannot be examined. In previous section we could already observe the strength of our approach by clearly stating the structural and behavioral properties of BPA patterns. The following use case from the public administration, the enterprise founding process, will show that the transformation of a BPA into one composed TF nets and its analysis with LoLA leads to finding indirect interdependencies and errors that cannot be detected with the pattern based verification technique presented in [5].

The EU-Service directive, passed in 2006, led to a restructuring of service provisioning in the public administrations across Europe. Suddenly, public administrations had to provide citizens and companies with a single point of contact

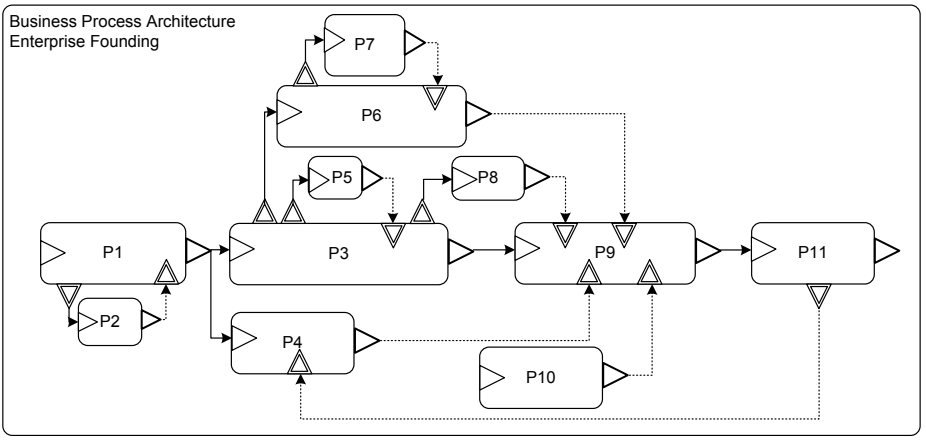

Fig. 5. Business Process Architecture depicting enterprise founding

for public services that were previously offered by many different administrative offices. For managing the huge amount of public services they were grouped according to life situations. With the centralization of service provisioning, it becomes apparent that also the interdependencies between public services need to be analyzed. The life situation, enterprise founding, consists of eleven public service processes. Examining them one by one does not provide much insight as it is unclear if their interaction leads to a desired output. E.g. Fig. 1 shows three EPCs depicting three public service processes from the overall enterprise founding process. On first sight, it appears that they are sound and independent from each other. On second sight, we notice interdependencies when looking at the events that match, e.g. the events in the dashed boxes in Fig. 1. The interdependencies between these processes can easily be reflected in a BPA as shown in Fig. 5. Looking closer at the example, it is obvious that some processes depend directly on each other, e.g. process p_1 and p_2, or p_3 and p_9. Applying the BPA patterns to a larger example with many interdependencies becomes rather complex. We can find many regular patterns between two processes, e.g. the interaction between processes p_6 and p_7, processes p_4 and p_9, processes p_9 and p_{11}, or processes p_{11} and p_4. The BPA appears to be faultless. The indirect interdependencies between processes p_4, p_9 and p_{11} stay hidden.

The transformation of the BPA into TF nets enables a thorough analysis of the BPA. Fig. 6 depicts the transformation of the BPA into one TF net that was then analyzed with the LoLA tool. The analysis showed that the TF net is structurally not sound as it has two initial places from process p_1 and p_{10}. The process p_{10}, check free of debt, is not triggered by any other process but is required as input for process p_9 issue estate agent permit. This could hint at a problem as process

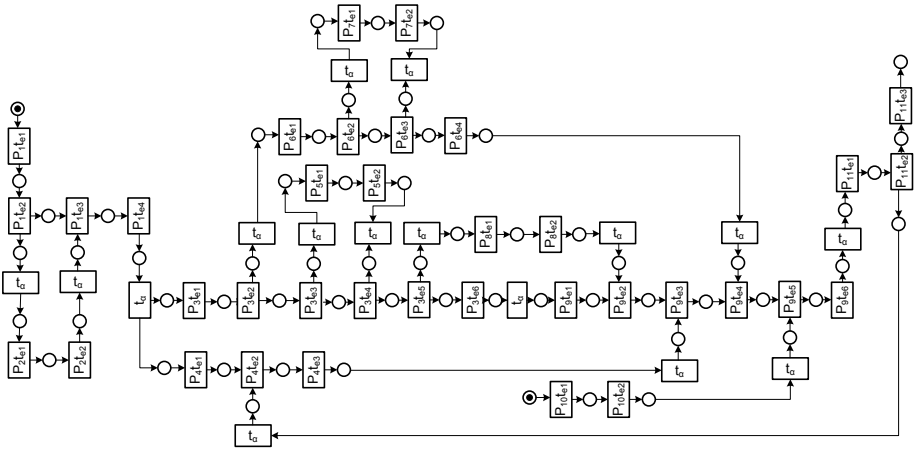

Fig. 6. Business Process Architecture behavior

p_{10} cannot know when to check for free debt and for whom. A trigger relation is missing. The first analysis iteration showed that the overall architecture has structural inconsistencies. The process depends on two independent inputs. After converting the TF net into a Workflow net, the analysis of behavioral properties found a deadlock and 12 dead transitions. The TF net is neither sound nor weak sound, however it is bounded. Process p_4 inhibits process p_9 as it waits for input from process p_{11}. However, p_{11} never starts as it is triggered by p_9 which cannot terminate due to p_4. By transforming the BPA into a TF net we clearly specify the behavior of the BPA. As demonstrated in our use case of the public administration we can detect behavioral errors that are hidden from the pattern matching approach presented in [5]. In regard to structural composition our approach may indicate problems.

6 Related Work

Applying Petri nets to workflow management systems has many advantages [11]. One of the advantages beside their clear semantics, is the existence of abundant Petri net verification techniques. [7,12,8,9] formalize EPCs, BPMN, and workflow models with Petri nets and provide them with clear behavioral semantics that can be analyzed. [7,8,9,13] show the effectiveness of formalizing business process models in Petri nets. E.g., Dijkman et al. [8] provide a BPMN to Petri net transformation to clarify ambiguities in the BPMN specification finding a number of deficiencies there. Our BPA to Trigger-Flow net transformation can be grouped to those approaches. It goes even further by also mapping the process interdependencies found in BPAs.

Lohmann et al. [9] investigate the transformation of process modeling languages into Petri nets. Transformations are mainly used to verify the structural and behavioral semantics of the source languages [9]. Mendling [13] found out

that most analysis of process collections use verification techniques based on decomposition, or combinations of techniques based on reduction and reachability. The studies examined by Lohmann et al. [9] and Mendling [13] look only at each process model in a collection individually and ignore their interdependencies. Our BPA approach combines the advantages of Petri net transformation with the holistic view of BPA on process model collections and their interdependencies. Many of the BPA approaches take rather a high level point of view and do not provide formal verification techniques to examine the interdependencies and inherent interactions of business processes within large business process collections. An extensive overview of BPA approaches is given by Dijkman et al. [1].

Literature on service composition, workflow modules, process orchestrations, process choreographies, and also open nets deal with business process interaction and provide approaches to verify behavioral or structural properties of business process interaction. Decker and Weske [14] highlight the need for the examination of behavioral consistency and compatibility of interacting processes in process choreographies. They introduce a framework for analyzing several service consistency specifications and service compatibility definitions for interacting services. Many approaches resort to Petri net based verification approaches [15,16,3,17,18]. Puhlmann and Weske [19] take a different approach and examine process orchestrations and choreographies with dynamic bindings using the π-calculus.

The foundational concepts of service interaction, basic interaction patterns and anti-patterns are introduced by van der Aalst et al. [18]. They use open nets to compose two interacting services and analyze their interactions for controllability. Martens [3] investigates the composition of web services and their interaction in regard to their compatibility and usability. Baldan et al. [15] define open nets to model inter-organizational process behavior and describe the composition of two interacting open nets along their common subnets. Their prime aim is to model the composition of interacting process rather than verifying their correct flow interaction. Weinberg [17] presents a methodology that analyzes open net transformation of WS-BPEL processes for controllability and calculates their operating guidelines using interaction graphs.

Glabeek and Storck [16] analyze workflow module interaction in regard to proper termination. They check local properties of participating Workflow nets to assure global termination of interacting workflow modules. Our approach in contrast assumes that business processes are locally sound and takes a global perspective. Due to the reduction of business processes to their events, and their trigger and flow relations the state space of a resulting BPA is considerably decreased and can be automatically analyzed, e.g. with LoLA.

In regard to Petri net composition, our approach resembles to [15,3] using incoming and outgoing places, but extends it with the composition of TF nets to represent trigger relations. Following [3,18], we define the transformation of BPAs to TF nets in order to represent their behavior and analyze their interdependencies, i.e. the correct behavior of BPAs. Our transformations does not only

allow for analyzing message interaction of processes like the other approaches but also for examining their trigger relations that are prevalent in BPAs.

7 Conclusion

In recent years managing and organizing large business process collections has become a major challenge and an integral part of BPM research. Especially, Business Process Architectures for organizing processes and describing their relations get stronger attention. Following these developments, this paper introduced a novel Petri net based formalization of Business Process Architectures, Trigger-Flow nets, that enable the analysis of Business Process Architectures with known Petri net analyzing techniques in regard to their structural and behavioral properties. Examining BPA patterns in this way, we identified five pattern categories. The effectiveness of our approach was demonstrated on a real use case from the public administration, the enterprise founding process. We found several dead events and a deadlock that could not be identified by a pattern based verification technique. Future work will deal with analyzing multi-instance pattern as well as developing a tool for automatic design of BPAs from large process collections and their verification.

References

1. Dijkman, R.M., Vanderfeesten, I., Reijers, H.A.: The Road to a Business Process Architecture: An Overview of Approaches and their Use. BETA Working Paper WP-350, Eindhoven University of Technology, The Netherlands (2011)
2. Green, S., Ould, M.: A framework for classifying and evaluating process architecture methods. Software Process: Improvement and Practice 10(4), 415–425 (2005)
3. Martens, A.: Analyzing web service based business processes. In: Cerioli, M. (ed.) FASE 2005. LNCS, vol. 3442, pp. 19–33. Springer, Heidelberg (2005)
4. Barros, A., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W., Benatallah, B., Casati, F., Curbera, F.: Service Interaction Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
5. Eid-Sabbagh, R.H., Dijkman, R.M., Weske, M.: Business Process Architecture: Use and Correctness. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 65–81. Springer, Heidelberg (2012)
6. Murata, T.: Petri nets: Properties, analysis and applications. In: Proceedings of the IEEE, vol. 77(4), pp. 541–580 (April 1989)
7. van der Aalst, W.M.P.: Formalization and verification of event-driven process chains. Information and Software Technology 41(10), 639–650 (1999)
8. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and Analysis of Business Process Models in BPMN. Inf. Softw. Technol. 50(12), 1281–1294 (2008)
9. Lohmann, N., Verbeek, E., Dijkman, R.: Petri Net Transformations for Business Processes – A Survey. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 46–63. Springer, Heidelberg (2009)
10. Weske, M.: Business Process Management: Concepts, Languages, Architectures, 2nd edn. Springer (2012)

11. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers 08(01), 21–66 (1998)
12. Pankratius, V., Stucky, W.: A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets. In: APCCM 2005, pp. 79–88. Australian Computer Society, Inc. (2005)
13. Mendling, J.: Empirical Studies in Process Model Verification. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 208–224. Springer, Heidelberg (2009)
14. Decker, G., Weske, M.: Behavioral consistency for B2B process integration. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 81–95. Springer, Heidelberg (2007)
15. Baldan, P., Corradini, A., Ehrig, H., Heckel, R.: Compositional Modeling of Reactive Systems Using Open Nets. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 502–518. Springer, Heidelberg (2001)
16. van Glabbeek, R.J., Stork, D.G.: Query Nets: Interacting Workflow Modules That Ensure Global Termination. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 184–199. Springer, Heidelberg (2003)
17. Weinberg, D.: Efficient Controllability Analysis of Open Nets. In: Bruni, R., Wolf, K. (eds.) WS-FM 2008. LNCS, vol. 5387, pp. 224–239. Springer, Heidelberg (2009)
18. van der Aalst, W.M.P., Mooij, A.J., Stahl, C., Wolf, K.: Service Interaction: Patterns, Formalization, and Analysis. In: Bernardo, M., Padovani, L., Zavattaro, G. (eds.) SFM 2009. LNCS, vol. 5569, pp. 42–88. Springer, Heidelberg (2009)
19. Puhlmann, F., Weske, M.: Interaction Soundness for Service Orchestrations. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 302–313. Springer, Heidelberg (2006)