A Taxonomy of Consistency Models in Dynamic Migration of Business Processes

Ahana Pradhan and Rushikesh Krishnakant Joshi

Abstract—The paper presents a taxonomy of notions of *consistency* in the context of dynamic migration of business processes covering two decades of research on dynamic process change in various workflow approaches including graph based and net based approaches. A consolidated view that brings out the generic nature of the consistency problem independent of the respective workflow model contexts is developed through the taxonomy framework. The consistency models are unified based on a common Petri net formalism. The classification into nine major consistency classes is derived primarily from differences arising out of the past, present and future of the post migration state. These are further grouped into four categories: *structure based*, *trail based*, *lookahead trace based* and *live* consistency models. Known usages of these models from the literature are also discussed. Moreover, several examples of dynamic workflow migration scenarios are presented to demonstrate the usefulness and applicability of the consistency models.

Index Terms—Business process, dynamic migration, consistency, process schema, trace, workflow nets

1 Introduction

Flexible process management systems are necessary for business enterprises to enable them to effectively react to business agilities such as changing requirements, changing technologies, changing contexts, which trigger changes to business processes. Research addressing the problem of *dynamic process change* began in the mid-90's with contributions on dynamic evolution and adaptation in process based systems. The early approaches include the formulation of the problem by Ellis et al. [1], its elaboration with the role of change primitives w.r.t. consistency by Reichert et al. [2], a formal *schema* (the process structure) compliance notion by Casati et al. [3], and an elaboration of workflow schema evolution in presence of versions by Kradolfer et al. [4].

Workflow processes have been a significant playground requiring the abilities of dynamic change. Since the inception of early ideas, the research in dynamic change in processes has matured into various finer aspects such as instance specific dynamic adaptations [5], [6], *schema evolution* [7], [8], [9], [10], [11], *flexible* and therefore *incremental* process modeling [12], [13], and handling of and adapting to *non-compliant* instances [14], [15].

Dynamic changes can be of various kinds, such as instance specific dynamic changes, ad-hoc changes to handle exceptions, or evolutionary changes reflecting across all or applicable instances. In all these change situations, the notion of consistency becomes necessary to decide upon the next state after the migration. Whenever a process changes, consistent instance migration becomes inevitable for achieving correctness of active information about the states of the instances. The scope of this survey is the issue of consistency and its solutions in this context. A Petri net based notation is used for modeling the control-flow structures of business processes.

1.1 The Issue of Consistency

Dynamic migration is a promotion of an instance of a workflow process into an instance of an evolved version of the process. Dynamic migration to a structurally different process may lead to unexpected behavior in the remaining execution of the migrating instance, if migration does not ensure consistency. This situation typically happens when the migration strategy is not derived by carefully consulting and deliberating the old and the new capabilities of the evolving business process in light of the business goals of migration. For instance, in the context of an evolutionary change, the business goal may be to maximize reuse of completed tasks while migrating instances. Keeping this goal in mind, the instances may be transferred to corresponding equivalent states in the new process, with equivalence defined so as to ensure that they bear the same history. Two primary issues that arise out of this situation are, first, whether a given process instance state is migratable, and second, if so, to which state. As noted by Weske [16, pp. 340], the consistency rules for identifying migratable instances play a role of prominence in solving this problem.

The literature has used many variations in both workflow modeling formalizations and the consistency criteria for migration. This survey analyzes these existing varied notions, and develops a consolidated view on the notion of consistency that brings in the existing variants of consistency models under a single umbrella. To achieve the same, a taxonomy of consistency models is presented. The consistency classes are defined in terms of Petri net models. A discussion on the applicability of each of the classes is provided through pragmatic examples of dynamic workflow migration. Let us

• *The authors are with the Indian Institute of Technology Bombay, Mumbai, Maharashtra 400076, India. E-mail: {ahana, rkj}@cse.iitb.ac.in.*

Manuscript received 25 Dec. 2016; revised 8 June 2017; accepted 22 July 2017. Date of publication 3 Aug. 2017; date of current version 8 June 2018. (Corresponding author: Ahana Pradhan.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2017.2735413

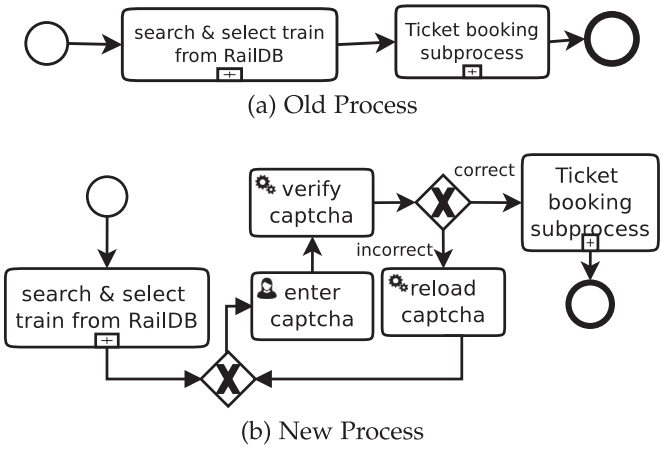


Fig. 1. Example of schema change in on-line ticket booking process.

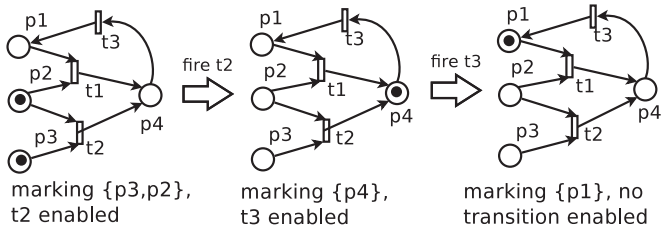


Fig. 2. Example of transition firing and token movement.

begin with an illustration demonstrating the need for *consistency* in dynamic process change.

1.2 Process P_1: Online Ticket Booking

A process *schema* represents the static structure of the process in terms of elements such as tasks, gateways and transitions. A process may have many *running instances* at any given time. For example, a ticket booking process may have k running instances if k users are active. Every instance of a process has its corresponding *runtime state* in the schema, which represents the tasks that are completed. First an example in BPMN [17], a widely used modeling notation for business processes is provided. In subsequent sections, the notations in terms of Petri nets are used. This choice is made to facilitate description and precise demarcation of various consistency criteria.

Consider a BPMN process for an online ticket booking website as shown in Fig. 1a. A user first searches for trains and selects one from the results. Next, the ticket booking subprocess is enabled. Due to security reasons, this simple process needs to be modified with an additional intermediate step of human verification as shown in Fig. 1b.

The old process may evolve even when user sessions are active. These active sessions are running instances of the old process. Migration of such instances without redoing the work done earlier facilitates the users to smoothly migrate into the new process without experiencing a bump (such as experiencing invalidation of search results). Dynamic migration not only avoids a server shutdown, but it also makes the migration experience smooth.

Let us consider two specific instances of the old process: (i) a user has selected a train after completing the train search and is yet to enter into the second subprocess, and (ii) a user activity is already inside the ticket booking subprocess. The states of the old and the new processes after completion of their respective *search and select train* subprocesses can be said to be *migration equivalent*. However, if the execution has proceeded inside ticket booking, it must continue in the old process, since it is otherwise a contradiction to call it verified in the new process. Thus the first instance satisfies the consistency, but not the second.

1.3 Organization of the Paper

The next section provides preliminaries on Petri nets for modeling workflow processes, and overviews the issue of consistency and schema compliance. Section 3 develops the taxonomy framework for the consistency models. Sections 4, 5, 6, and 7 elaborate the consistency classes in the taxonomy with the help of examples, discussing occurrences in the literature. Section 8 discusses a notion of *business consistency* to capture parameters external to the process specification.

2 Preliminaries

Petri nets offer simplicity and an intuitive appearance for formal modeling of business processes as pointed out in [18]. This section provides a background to application of Petri nets to workflow modeling through Workflow nets (WF-nets) [19]. In the rest of the paper, the WF-net version of Petri nets is used as a common reference model for representing processes and for analyzing consistency. In this section we summarize the terminologies and definitions that are used throughout the paper.

2.1 Elementary Petri Net

A Petri net consists of two types of nodes, which are places (circles) and transitions (bars or boxes). Arcs connect places to transitions and transitions to places. This net structure can model dynamics of discrete event-based systems when *tokens* (dots or filled circles) are put into places. A place containing a token is called a *marked place*. The set of all marked places make up a *marking* in the net. In *elementary* nets, no place can contain more than one token at any point of execution, but multiple tokens may exist in the net as far as this constraint is satisfied. Transitions are active elements, which change markings in the following manner.

A transition is *enabled* when each of its *pre-places* have a token, and all its *post-places* are empty. When a token enables one or more transitions, it gets *consumed* by a *firing* of one of the enabled transitions. When an enabled transition *fires*, it consumes one token from each of its pre-places, producing one token in each of its post-places. This process causes a change in the marking.

Fig. 2 depicts an example of a change in the marking of a net. The initial marking $\{p_2, p_3\}$ enables transition t_2, resulting in marking $\{p_4\}$, which further enables transition t_3, which fires resulting in marking $\{p_1\}$. Marking $\{p_1\}$ is a stuck state in this net, as no more transition is enabled in spite of a token being present in the net.

A Petri net can be used to model real-world events by transitions. Buffers, storages and, pre- and post-conditions are mapped to places. Accordingly, a token in a place represents the presence of data in the corresponding buffer/storage element, or a `true` status of the corresponding condition. The reader is referred to Murata [20] for a comprehensive introduction to properties, analysis techniques and engineering application of Petri nets.

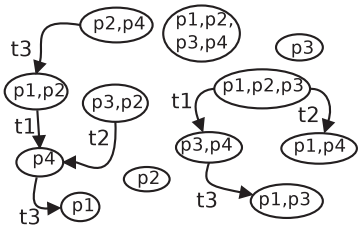


Fig. 3. State-space of the net in Fig. 2.

Pre- and Post-Sets. For a node x (place or transition) in a given Petri net, the set of immediate predecessor nodes (transitions or places) of x forms the pre-set of x, which is denoted by $\bullet x$. Similarly, the set of immediate successor nodes construct the post-set of x, which is denoted by $x\bullet$.

Firing Sequence (Trace). The following notations represent the dynamics of a net. A *firing sequence* σ is a sequence of transition firings (and not markings) from one marking to another. A unit transition of marking $m_1 \xrightarrow{t} m_2$ in a net denotes that the firing of transition $\mathbf{t}$ changes the marking in the net from m_1 to m_2. As a more generalized form, transition $m_1 \xrightarrow{\sigma} m_2$, where σ is a firing sequence $t_1 t_2 \dots t_n$ denotes that the firing sequence σ changes the marking from m_1 to m_2 through the corresponding sequence of markings that can be obtained from the firing sequence σ. Multiple firing sequences between two markings occur in nets with choice and concurrency. In Fig. 2, we have $\{p_2, p_3\} \xrightarrow{t_2} \{p_4\}$, and $\{p_2, p_3\} \xrightarrow{t_2 t_3} \{p_1\}$.

Restriction. Given T as the set of all transitions in a net, a restriction [21] of the firing sequence σ to a set $X \subseteq T$, is denoted by $\sigma|_X$. Restriction $\sigma|_X$ is a sequence of transitions, which includes only those transitions in σ that belong to X, and it excludes the rest. For example, $t_1 t_2 t_3 t_4|_{\{\mathbf{t}_3, t_1\}} = t_1 t_3$. This notion of restriction has been used later to define consistency models.

State-Space and Reachability Graph. The set of all possible markings in a given net, and the transitions among them make the state-space of the net. Fig. 3 depicts the state-space of the net given in Fig. 2. States in a state-space represent markings in the net. Given an initial marking, the state-space that is reachable is termed as the *reachability graph*. Fig. 4 shows the reachability graph for initial marking $\{p_1, p_2, p_3\}$ for the state-space.

2.2 Workflow Net (WF-Net)

The notion of WF-net [19] was introduced for applying the Petri net model to workflow modeling and analysis. It is a restricted type of elementary Petri net. *Sequence*, *Choice*, *concurrency* and *loop* are the four commonly occurring control-flow patterns in workflow models. They are often referred to as *primitive workflow patterns* or *routing constructs* [19]. The patterns can be nested. Several other workflow patterns have also been well-documented in the literature [22], [23].

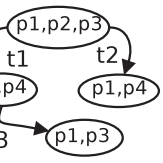


Fig. 4. Reachability graph of the net in Fig. 2 with initial marking $\{p_1, p_2, p_3\}$.

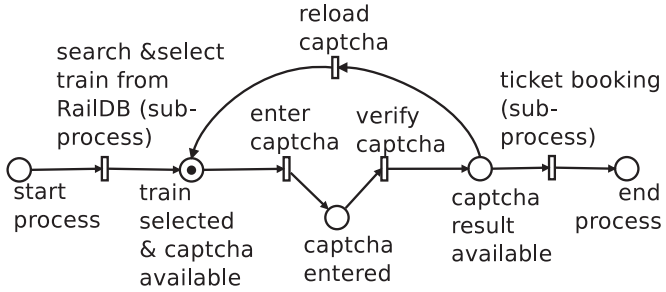


Fig. 5. WF-net model with tasks modeled as transitions.

The structure of a WF-net N is defined as a tuple (P, T, F), where P is a finite set of places, T is a finite set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs. The properties of WF-nets are summarized below:

- *Unique Source:* $init \in P$ is the only source place. The source place is identified by condition $\bullet init = \emptyset$.
- *Unique Sink:* $end \in P$ is the only sink place. The sink place is identified by condition $end\bullet = \emptyset$
- *Connectedness:* Apart from the source place $init$ and the sink place end, all other places and all transitions appear on at least one directed path from $init$ to end.
- *Unique Initial Marking:* One token in the source place $init$ gives the initial marking M_0.
- *Unique Terminal Marking:* When the token reaches the sink place, all other places $p \in P \backslash \{end\}$ are in unmarked state.
- *Behavioral Well-formedness:* (i) No Dead Transition: Every transition $t \in T$ can be fired from some marking M reachable from M_0 (ii) Proper Termination: Every marking M reachable from M_0 eventually reaches terminal marking.

  Set $\mathbb{R}(M)$ gives the set of reachable markings from a given marking M. Therefore, $\mathbb{R}(M_0)$ is the valid set of markings in the corresponding WF-net.

2.2.1 Modeling of Tasks Using Transitions

WF-net is often used to model a workflow schema with transitions as tasks. The places represent pre- and post-conditions of respectively their successor and predecessor tasks. The *source place* captures the initial condition that triggers the start. Similarly, the *sink place* models the terminating condition for the workflow. A marking represents a runtime state of the process instance. Following the convention described in [19] for WF-nets, a schema is modeled by a net and its instance by a marking in the net.

Fig. 5 depicts a WF-net representation of the railway ticket booking process in Fig. 1b. The labels of places represent conditions, and the transitions correspond to BPMN activities. The net includes a marking {*train selected & captcha available*} corresponding to an instance, in which, the user has selected a train to book ticket, while the captcha is yet to be keyed in. It can be noted that, though the captcha conditions *correct* and *incorrect* in the BPMN model shown in Fig. 1b are not shown explicitly in this net model, they are mapped to two non-deterministic choice (XOR) paths.

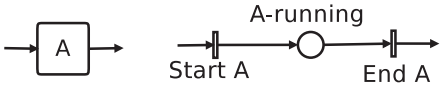


Fig. 6. Modeling of a workflow task as a sequence in WF-net model.

2.2.2 Modeling of Tasks Using Places

Though transitions are commonly used to model tasks, places can also be used for the same purpose. In such a case, a pre-transition and a post-transition respectively represent the start and end events corresponding to that activity. When a token is inside a place, it indicates that the activity is in state *running*. Fig. 6 illustrates the scheme of this approach for a real-world activity A.

As a concrete example, in the case study of wine production workflow presented in [24], the activity of *grape harvesting* is modeled as a sequence of *harvest begin* (a transition), *harvesting* (a place) and *harvest end* (a transition). Consequently, a token in the place *harvesting* represents the local state of grapes being harvested.

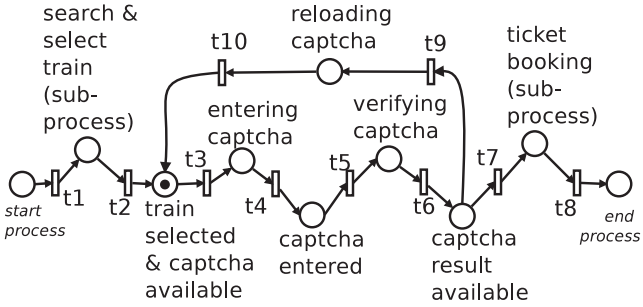
The mapping of the BPMN process shown in Fig. 1b to WF-net with places as tasks is given in Fig. 7. Pairs of transitions are used to model start and end events of the activity *placed* between them. Besides, there are pre- and post-conditions of real-world activities (which are sequences $transition\text{-}place\text{-}transition$), which are also modeled as places. The difference between the models in Figs. 5 and 7 is that one transition in the former corresponds to a sequence transition-place-transition in the latter. The latter model is more explicit, due to which the running state of the activities can be naturally tapped through markings. This fact comes handy in formulating a general framework for modeling consistency. It is noted that, a mixed model with both places and transitions used as tasks in the same net is also available in the literature as in [24].

2.2.3 Markings as a Set of Places

Behaviorally WF-nets are elementary nets. Hence, in every possible marking in the net-dynamics, a place can hold only one token at most. A marking is often represented as a column matrix, where rows represent places. Marked and unmarked places are identified through 1s and 0s respectively (since at most one token is in a place). The marking shown in Fig. 5 is represented as column vector $(01000)^T$. Another representation of markings is a marking as a set of places. The same marking can be represented as set *{train selected & captcha available}*. In this paper, this set representation of markings is followed.

2.3 Consistency versus Compliance

Many authors have used the term *compliance* to mean consistency or vice versa blurring the distinction between them. Examples of the use of the term *compliance* are *schema compliance* and *instance compliance* of Reichert et al. [9], *instance compliance* of Sadiq et al. [25]. Examples of the use of term *consistency* are *schema consistency* and *instance consistency* of Casati et al. [3], and *consistency* of Ellis et al. [1]. The terms *compliance* and *consistency* are separated to mark the difference between the two separate concepts of *process schema change* and *instance state change*. The categorization in the survey by Song and Jacobson [26] defines *schema compliance* to correspond to static process change, whereas *instance consistency* corresponds to dynamic process change.



t1: start searching, t2: complete selection, t3: start typing captcha, t4: end typing, t5: verification start, t6: complete verification, t7: start ticket booking, t8: finish booking, t9: start reloading, t10: reloading complete

Fig. 7. WF-net model with tasks modeled as places.

We note that, static change versus dynamic change, and schema change versus instance state change are two orthogonal dimensions. It can also be noted that though the literature tends to closely tie process modeling notations with chosen model of consistency as observed from the previous survey on consistency criteria [27], the notions of consistency are generic enough to be independent of the modeling notation that is used to represent workflow processes. This is the main theme of this taxonomy.

2.3.1 Consistency in Instance Migration

The task of determining whether an instance of an old workflow process can be migrated to a new workflow process, and if so, precisely as which instance of the new process, is in the purview of *consistency*. Whether an individual instance state is a valid state is a problem of reachability. The problem of consistency on the other hand, deals with migration of an instance to a particular reachable state in the new process such that the new state can be seen as equivalent to the old state in a given application context. The rules of such equivalence may change from business to business. However, there are certain common formulations, which are brought out through our classification. Consistency rules thus define the criteria for equivalence between the states of two processes. The equivalence may be a partial mapping. The classification of consistency is dealt with in depth in the rest of the paper. Before discussing these classes, the related notion of schema compliance is briefly noted below.

2.3.2 Process Schema Compliance Based on Migratability

The issue of consistency becomes relevant only if there is some sort of compliance between the two workflow processes. Minimally, for two processes to be compliant under migration, at least one instance should be consistently migratable under some agreeable consistency criteria. The degree of compliance is defined as the share of migratable instances. Given a specific consistency criterion, migratability may prevail over all, some or no instances. Accordingly, the two processes can be called as fully complaint, partially complaint, or non-compliant. Again, the degree of compliance may vary significantly in terms of the share of migratable instances in partial compliance.

TABLE 1
Taxonomy Framework for Consistency Models

Consistency Model	Parameters									
	Time-frame used			How to Compare two Trace sets			How is a Trace represented			Is it Structure based
	Past based	Present based	Future based	equal	subset	superset	set of tasks	sequence of tasks	purged trace	
Structural equivalence (SE)	✓				*NA*					✓
Trace equivalence (TE)	✓			✓				✓		
History equivalence (HE)	✓			✓			✓			
Purged-trace equivalence (PTE)	✓			✓				✓	✓	
Purged-history equivalence (PHE)	✓			✓			✓		✓	
Live (L)		✓			*NA*					✓
Strong lookahead (SL)			✓	✓				✓		
Accommodative lookahead (AL)			✓			✓		✓		
Weak lookahead (WL)			✓		✓			✓		

2.3.3 Approaches to Process Compliance in the Literature

An old process schema is *compliant* to a new schema if both are free from control-flow and data-flow errors, and if they satisfy certain property that is an invariant relation between the two schemas. An example of such property based compliance is the approach of Sun and Jiang [10] who relate the schemas in terms of their traces. Another example is that of the approach of Van der Aalst and Basten [8] that uses branching bisimulation [28] to compare the schemas.

WIDE graph [3] and ADEPT WSM-net [9] also define schema compliance in terms of traces. Song and Jacobson [26] discuss compliance based on the sameness in *public views* of two processes.

In spite of property based compliance, instance level consistency models also need to be worked out for actual migration. However, though the separation offers a more generic equivalence in terms of compliance, as evident from the literature, we note that to define instance level consistency, it is not necessary to base it on a separate definition of schema level compliance.

3 A Taxonomy of Consistency Models

Now that the background of workflow processes, and the notion of consistency in migration has been developed, let us shift the focus to the parameters that are influential in establishing equivalence between two *control-flow states* of two given nets. Based on these parameters, a taxonomy of consistency models is then introduced. The paper also discusses how these consistency models manifest in the literature on dynamic process migration. Many non-Petri net workflow modeling languages exist in the literature, a few of which such as ADEPT WSM-nets [2] consider data-flow in addition to control-flow for defining consistency. For those cases, their control-flow consistency aspect relating to Petri net models is brought out separately. The primary aim of this survey and the taxonomy is to formulate generic notions of consistency in terms of Petri nets, to exemplify them through elaborate cases, and finally to relate them to those hidden behind various model specific formulations presented in the earlier research and as reviewed in the earlier surveys [27], [29].

3.1 Parameters of Consistency

The taxonomy framework classifying the models of consistency is brought out in Table 1. As shown in the table, the models of consistency can be distinguished from each other based on several parameters. They are obtained from the following basis: (i) Role of *time-frame* (logical time) relative to the current marking (past, present, future), (ii) The comparison operator used to compare two trace sets from old net and the new net (equality, subset relation, superset relation), (iii) Trace representation structure used (as sets of tasks, sequences of tasks, as purged sets or as purged sequences), and (iv) Role of the net-structure in defining consistency.

Fig. 8 shows a marked WF-net, on which its present state, and the regions in its past and future are identified. The present state is the marking $\{p_4, p_5\}$, which is composed of structural elements of the net, i.e., the places. The past can be viewed in two ways: either in terms of the traversed net, which is a subnet structure, or in terms of the firing sequence that reached the shown marking from the initial marking. In some cases, since the exact trace cannot be deciphered from a marking, it is assumed to be available through the use of logs. Alternatively, in absence of exact logs, a set including all such non-deterministic choice-paths may also be used. In real-world workflow scenarios, the trace is typically represented in terms of event logs. Lastly, the future can similarly be viewed either by the structure or by the trace. A future trace or *lookahead trace* is a possible firing sequence from the current marking to reach the terminal marking. Since there may be many such possible traces, we speak of the *set* of lookahead traces.

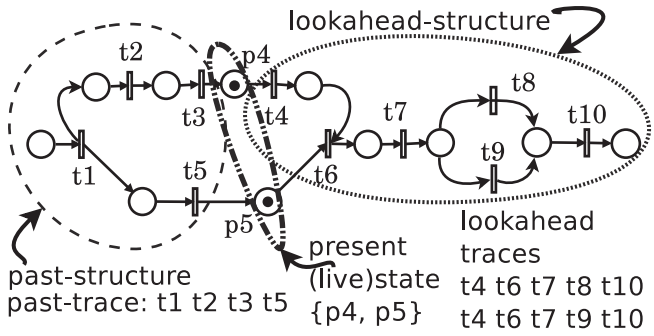


Fig. 8. Focus regions corresponding to parameters of consistency.

Apart from control-flow, consistency may depend on factors external to the process model such as business data, business rules and business equivalences. These cases are grouped under a common umbrella model of *Business Consistency*. We separately discuss this model in Section 8, since it does not use any of the parameters listed in Table 1.

3.2 Consistency Classes

The above parameters give rise to various ways to compare two states from two nets for determining consistency of a migrating instance. Thus, two markings M and M' respectively from nets N and N' can be compared based on their past, present or future. The taxonomy given in Table 1 brings out these variations depicting a classification of consistency models covering possible ways of comparisons between the two given states to establish migration equivalence between them. These are Structural equivalence (SE), Trace equivalence (TE), History equivalence (HE), Purged-trace equivalence (PTE), Purged-history equivalence (PHE), Live (L), Strong lookahead (SL), Accommodative lookahead (AL), and Weak lookahead (WL) models.

The table identifies consistency classes based on the parameters as discussed previously. The structure of the table provides a generic framework that may also be used for obtaining newer models of consistency. The classes in the taxonomy can be viewed from (i) a trace based view, or from (ii) a structure based view, or from (iii) a time-frame view (w.r.t. the current marking as either past, present, or future based). These views are discussed below before individual classes are dealt with in detail. These three views have overlaps as it can be noticed from the table and from the discussion that follows.

3.2.1 Structure-Based Models

Structure-based models are based on net elements. Consistency may be established by comparing the traversed subnets. The model SE belongs to this class. If only places of the current marking sets are used to compare, we have another structural model, which is L.

3.2.2 Trace-Based Models

When the old state M is compared to the new state M' based on trace, the comparison can be made based on either past traces (trails) or future traces. The past-based and future-based models are hereafter referred to as *trail based* and *look-ahead* models respectively. Equivalence of these traces (past or future) can be defined in terms of sets or sequences. Also, some consistency models may permit removal of tasks from traces, which is referred to as purging. These variations are noted below.

- *Sequences and sets*: The order of task-execution in a trace can either be considered or ignored. Consequently, either set equality or strict ordering can be used to compare traces. The terms *trace* and *history* are used to denote consistency, respectively when sequences and sets are used for determining equivalence. Models TE, PTE, SL, AL and WL use strict ordering, whereas, models HE and PHE use set based comparison of tasks in their traces.

TABLE 2
Symbols Used in Definitions

Term	**Old Net** $N = (P, T, F)$	**New Net** $N' = (P', T', F')$
Initial Marking	M_0	M'_0
Terminal Marking	M_E	M'_E
Current Marking	M	M'
A firing sequence from some marking M_1 to some marking M_2	$M_1 \xrightarrow{\sigma} M_2$	$M'_1 \xrightarrow{\sigma'} M'_2$
Firing sequence from initial to current marking (Trace)	σ, such that $M_0 \xrightarrow{\sigma} M$	σ', such that $M'_0 \xrightarrow{\sigma'} M'$
Set of fired transitions in firing sequence σ	$s(\sigma)$	$s(\sigma')$
Set of lookahead traces	$F_M = \{\sigma \mid M \xrightarrow{\sigma} M_E\}$	$F_{M'} = \{\sigma' \mid M' \xrightarrow{\sigma'} M'_E\}$

- *Purging*: On the other hand, in purging model, some tasks such as *deleted tasks* from traces may be ignored. The models that use purging in traces are PTE and PHE.

3.2.3 Time-Frame Based Models

In time-frame, three dimensions, past, present and future w. r.t. the current marking are considered.

- *Past:* When only the firing sequences starting from the initial markings to the current marking are considered, we get four trace-based consistency models: TE, PTE, HE and PHE. In structure based models, SE is past-based.
- *Present:* In this class, only the place labels of markings in the two nets are considered to determine equivalence. We have one model in this group: L.
- *Future:* In terms of comparisons between possible future traces starting from current marking up to the terminal, we list three *lookahead* models: SL, AL and WL.

3.2.4 Discussion Template

For discussing the consistency models for dynamic migration of markings from an old net to a new net, the notations listed in Table 2 are used. In the subsequent sections, the nine classes of consistency in the taxonomy are discussed in detail. For each case, the following template is followed: (i) First a WF-net based formulation and a definition is provided. (ii) Next, an illustrative example in WF-net notation is given. (iii) Use of the consistency model in the literature are then discussed with fresh examples. The examples are sketched out in the respective native modeling notations and also in equivalent WF-nets, and then (iv) we discuss how the consistency notion fits into the taxonomy.

A thumb rule that we followed while carrying out translations from non-Petri-net models to Petri net models was to accurately capture the parameters relevant to the consistency criteria used by the respective approaches. Such a translation is necessary to consolidate different classes of consistency under a single formalism, and to portray the

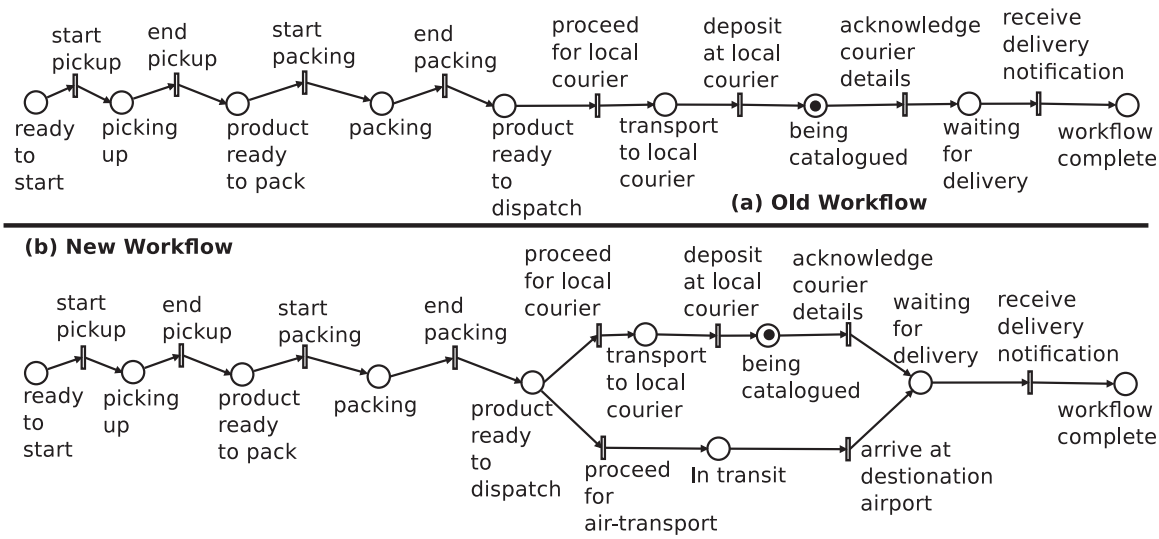


Fig. 9. Example of an evolving goods shipping workflow.

notion of consistency and its variations independent of different modeling notations. The following guidelines have been followed: (i) the marking bear the same history as that of the original model (ii) the same set of activities are captured (iii) all data dependencies are captured with data elements modeled by additional places. Thus, we make sure that data dependencies wherever needed are not left out. As a result, traces in Petri net translations capture the data dependencies that required by their consistency models.

4 Structural Equivalence

A highly restrictive and a past-based approach for finding migration equivalence is to structurally match the executed (traversed) portion of the partially completed instance against the altered schema. The motivation behind this approach is to ensure that the completed portion of the workflow is the same even after the structural modifications, and then to resume the execution into a possibly new future thereof. In Petri net formalism, since a workflow instance is modeled by a marking in the schema-net, the traversed net elements (the places, arcs and transitions) have to be derived from the firing sequence that reached the current marking. Logs are often used in this case. Now, the definition of *structural equivalence* is provided (using notation in Table 2).

4.1 Formulation

Let T_s be the set of transitions that appear in the trace σ reaching to the current marking M from initial marking M_0 in the old net, i.e., $T_s = s(\sigma)$. Let P_s be the set of pre-places of the transitions belong to T_s, i.e., $P_s = \{p | p \in \bullet t, t \in T_s\}$. At some point each member of P_s held a token, and all such members are in set P_s. Let F_s be the set of arcs that appear in the subnet covered by T_s and P_s, i.e., $F_s = \{(T_s \times P_s) \cup (P_s \times T_s)\} \cap F$. Let N_s be the traversed subnet in old net N, i.e., $N_s = (P_s, T_s, F_s)$. Let marking M' in the new net N' is a marking with traversed subnet N'_s which is constructed for N' similarly corresponding to marking M' in N'.

Structural equivalence defines two given markings to be consistent when their respective already traversed-subnets are structurally the same, i.e., each of the traversed subnets consist of the same sets of places, transitions and arcs. Following the formulation of traversed-subnet tuples N_s and N'_s, the equality condition is given below.

Definition 1. *Structural Equivalence between markings M and M' is defined in terms of traversed subnets: $N_s = N'_s$, which expands to $P_s = P'_s$, $T_s = T'_s$, and $F_s = F'_s$.*

Each member of the tuple being a set, set equality is used for member equality for tuple members noted in the expanded terms above. When structural equivalence is met, the traversed portion of the old instance is a subnet in the new schema yielding a valid marking that is seen as consistent migration for the old marking. From among those discussed in this taxonomy, this is the strictest model of consistency in terms of structural changes to the past-structure. It can be observed that this model permits changes only in those portions of the schema that are not executed. An example of use of this definition is given next.

4.2 Process P_2: An Order Shipping Process

Figs. 9a and 9b respectively show an old and a new workflow for an e-commerce shipping process. Items are shipped from the warehouse to a remote dispatch center. The old workflow starts with picking up the product from the warehouse. After packaging, it is transported to local courier for delivery. The new policy suggests to use an air-mail service for premium purchases of some of the products, such as non-fragile products weighing under 7 kgs. Therefore, the new workflow adds an alternative choice of transporting the item by air-mail. The two markings shown in the figure are consistent by structural equivalence. Let us see how. For the old net in Fig. 9, the following values can be obtained:

M_0 = {*ready to start*}, M = {*being cataloged*}
σ = *start pickup-end pickup-start packing-end packing-proceed for local courier-deposit at local courier.*
T_s = *{start pickup, end pickup, start packing, end packing, proceed for local courier, deposit at local courier}.*
P_s = *{ready to start, picking up, product ready to pack, packing, product ready to dispatch, transport to local courier}*

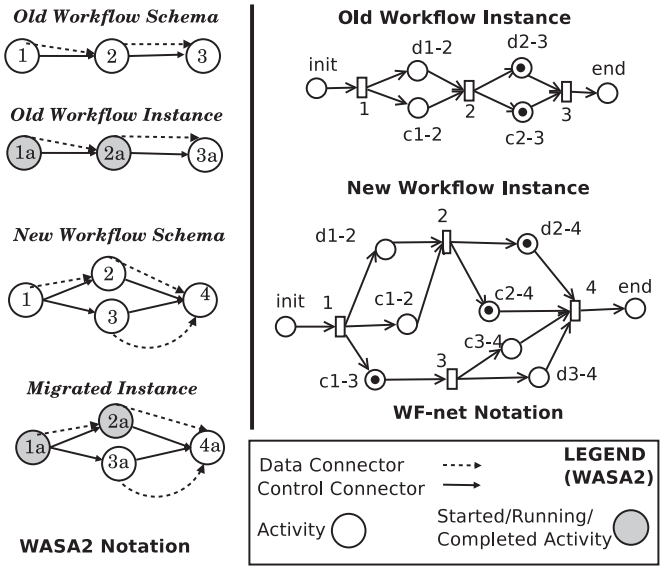


Fig. 10. Example of structural equivalence in $WASA_2$ approach.

F_s = *{(ready to start, start pickup), (start pickup, picking up), (picking up, end pickup), . . . (packing, end packing),(end packing, product ready to dispatch), (product ready to dispatch, proceed for local courier), (proceed for local courier, transport to local courier), (transport to local courier, deposit at local courier)}*

N_s= tuple (P_s, T_s, F_s)

Now it can be observed that in the new net if $M' = \{$being cataloged$\}$, tuple N'_s turns out to be exactly the same as tuple N_s above, which establishes structural equivalence between markings M and M'.

4.3 Appearances in the Literature

Consistency in $WASA_2$ Approach. $WASA_2$ [30] is one of the first-generation workflow management systems providing dynamic change facilities on workflow schema design. Their workflow model [31] is based on directed graphs. The nodes (circles) in a workflow graph model activities (or sub-workflows). Directed control (solid arrows) and data connectors (dotted arrows) connect the nodes, modeling the causalities. Additionally, activities that are under execution are shaded. A workflow is constructed in Fig. 10 to show an example migration scenario represented in the $WASA_2$ modeling notation. The notation represents instances independently by showing executed nodes shaded. The figure also provides its equivalent WF-net by adding d places for capturing data dependencies, c places as capturing control dependencies, and by using transitions for $WASA_2$ activities.

The consistency of instance migration is determined by identifying an instance level valid one-to-one mapping of the nodes and the arcs from only the executed portion of the instance, which amounts to the traversed structure. When such a mapping does not exist, the instance is *not migratable*.

In this case, by going through the old WF-net in Fig. 10, it can be observed that $M = \{d2\text{-}3, c2\text{-}3\}$. Hence, $T_s = \{1, 2\}$, $P_s = \{d1\text{-}2, c1\text{-}2, init\}$, $F_s = \{(init, 1), (1, d1-2), (1, c1-2), (d1-2, 2), (c1-2, 2)\}$. We have $N_s = (P_s, T_s, F_s)$. For new marking $M' = \{c2\text{-}4, d2\text{-}4, c1\text{-}3\}$, we have $T'_s = \{1, 2\}$, $P'_s = \{d1\text{-}2, c1\text{-}2, init\}, F'_s = \{(init, 1), (1, d1\text{-}2), (1, c1-2), (c1-2, 2), (d1-2, 2)\}$. Thus, $N'_s = N_s$, since $P_s = P'_s, T_s = T'_s$ and $F_s = F'_s$, proving structural equivalence.

The Approach of Qiu et al. Fig. 11 constructs an example migration scenario in the directed graph based notation of Qiu et al. approach [32]. Nodes are activities, which are connected by directed arcs. The figure also shows its equivalent WF-net.

In their approach, the migration mechanism is based on identification of *bypassable nodes* of the old running workflow. *Bypassable* nodes represent activities that are not needed after migration. A node becomes modified in any of the two ways: (i) its *property*, i.e., the associated application task is modified as in task replacement (ii) its *input connectors* are modified. Any node which is already *finished* and not modified is considered bypassable (in other words, done). For instance, if a node that is already executed in the old instance has a new inserted node as its predecessor in the new workflow, the old node is considered to be *not bypassable* due to violation of condition (ii). Non-bypassable nodes are required to be re-executed after migration. Such a state, if it exists, can be found by rolling back. In the figure,

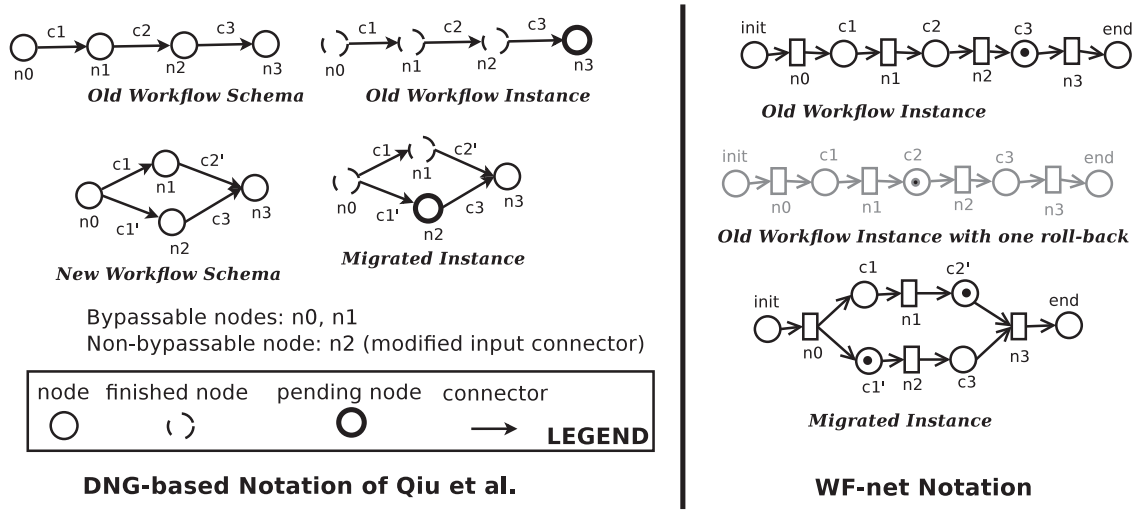


Fig. 11. Example of structural equivalence in Qiu et al. notation.

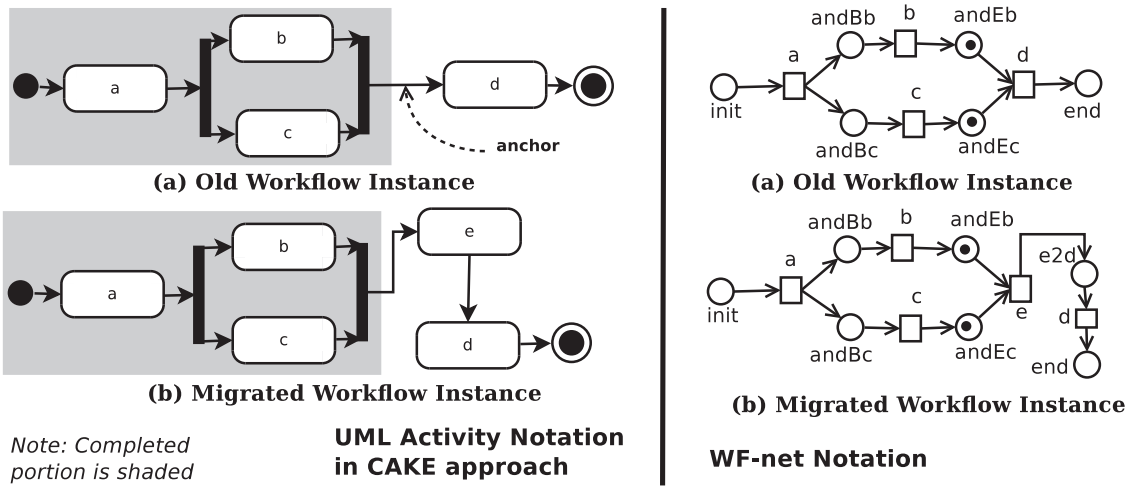


Fig. 12. Example of structural equivalence in CAKE approach.

inputs to n_2 and n_3 have been modified, which make them non-bypassable. Accordingly, a token in n_2 or n_3 is rolled back for obtaining a consistent migration. The WF-net in lighter font in Fig. 11 shows this computation of finding a structurally equivalent state for the given old instance.

In this case, structural equivalence to the rolled-back state is established by the following computation: old marking $M = \{c_2\}$. Hence, $T_s = \{n_0, n_1\}, P_s = \{c_1, init\}$, $F_s = \{(init, n_0), (n_0, c_1), (c_1, n_1)\}$. We have $N_s = (P_s, T_s, F_s)$. For new marking $M' = \{c'_2, c'_1\}$, $T'_s = \{n_0, n_1\}$, $P'_s = \{c_1, init\}$, $F'_s = \{(init, n_0), (n_0, c_1), (c_1, n_1)\}$. Hence, $N'_s = N_s$, since $P_s = P'_s, T_s = T'_s$ and $F_s = F'_s$. Therefore, it satisfies structural equivalence after the roll-back of one-state, requiring a repeat of non-bypassable activity n_2.

It is noted that, this old instance which is thus migratable in Qiu et al. approach is non-migratable in WASA_2 approach discussed earlier, due to absence of roll-back.

Case Adaptation in the CAKE Approach. The workflow adaptation approach in the CAKE project [33] also aligns with the notion of structural equivalence. The CAKE modeling approach uses block-structured workflow models in UML activity notation. The difference between this approach and the previously discussed two approaches is that in the CAKE system, a change is done to a *specific instance* as opposed to all instances based on what is called a "change request". For a change request, the elements to be deleted and added are determined by consulting prior adaptation cases and manual revisions for confirmation.

Anchors are the points in the running workflow where new chains of activities can be added. The approach considers the already-executed part of the workflow as *non-editable*, and hence, only the remaining part of the workflow is assessed to determine anchor position. In other words, any structural elements in the past are carried forward, and the change request is satisfied by editing the remaining part of the workflow if it can be done, which aligns with the structural equivalence model.

Fig. 12 constructs an example of a running case with an anchor-point, and its adaptation to a new workflow in CAKE notation. Completed portion is shown shaded. The equivalent WF-net representation is also constructed. A new task e is added downstream at the anchor point. From the WF-net we can deduce the structural equivalence as follows: $M = \{andEb, andEc\}$, $T_s = \{a, b, c\}$, $P_s = \{andBb, andBc, init\}$, $F_s = \{(init, a), (a, andBb), (a, andBc), (andBb, b), (andBc, c)\}$. We have $N_s = (P_s, T_s, F_s)$. For new marking $M' = \{andEb, andEc\}$, we have P'_s, T'_s, and F'_s which are the same as P_s, T_s and F_s respectively, satisfying condition $N_s = N'_s$ for structural equivalence.

5 Trail-Based Consistency

A *trail* is the exact trace of execution of an instance prior to the current state. These models establish state equivalence based on the past transition firings that have reached the current state. Therefore, in terms of structure, these models are weaker than the structural equivalence as all the preceding control-flows need not be considered to establish state equivalences. We enlist four variants of the trail-based models in the following, based on whether the ordering of transitions in the trace are taken into account and whether all of the transitions in the trace are taken into account.

5.1 Formulation

- Let σ be the trace, the actual transition firing sequence recovered from the logs from initial marking M_0 to current marking M in the old net N. Trace σ is represented as $M_0 \xrightarrow{\sigma} M$.
- Let some trace σ' be a trace in the new net N'. Trace σ' is represented as $M'_0 \xrightarrow{\sigma'} M'$ from initial marking M'_0 to a marking M' in the new net N'.
- Let T and T' be the full sets of transitions respectively in nets N and N'. $T \cap T'$ gives the set of transitions from the old net that are also present in the new net.
- $\sigma|_{T \cap T'}$ denotes restriction of trace σ, to set $T \cap T'$, keeping only those transition appearing in the intersection.
- $s(\sigma)$ and $s(\sigma')$ are the sets of transitions in traces σ and σ' respectively.

Definition 2. *Markings M and M' are trail-consistent if any of the following conditions is satisfied: (i) $\sigma = \sigma'$* (trace equivalence) *(ii) $s(\sigma) = s(\sigma')$* (history equivalence) *(iii) $\sigma|_{T \cap T'} = \sigma'|_{T \cap T'}$*, (purged trace equivalence) *(iv) $s(\sigma|_{T \cap T'}) = s(\sigma'|_{T \cap T'})$*, (purged history equivalence).

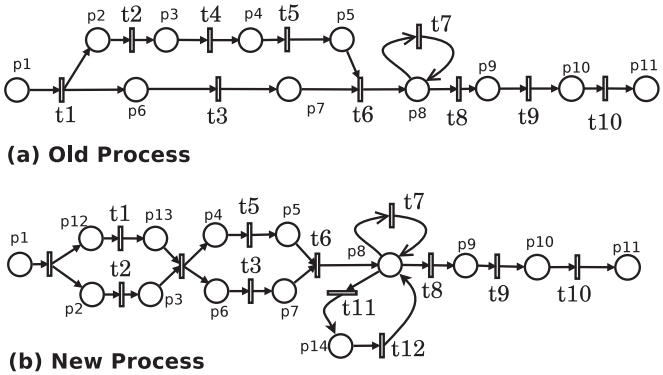


Fig. 13. Trail-based models: Example of an accommodation process.

When order in the trace is considered, the term *trace* is used, and the term *history* is used when the exact order of occurrence is not of interest.

As long as the respective conditions are preserved, structural changes are allowed by the above four trail-based consistency models. In purged-trace equivalence, the transitions that are carried over into the new net must appear in the same order. As a consequence, trivially if all transitions are new, the two nets are purged-trace equivalent. In other words, either everything is changed, or the unchanged parts are order-preserving. If the transitions in the old trace are carried over in the new net, they must appear in the new trace. In purged-history equivalence, the only difference is that the order is not important, and hence set equality is used for comparison. Since purged-history equivalence criteria is set based which tolerates changes to order in addition to tolerating deletions, several traces may become migratable. This is the most relaxed trail-based criterion.

5.2 Process P_3: An Accommodation Process

5.2.1 The Old process

Fig. 13a depicts an accommodation process in a hotel. Once a room for a guest is selected (t_1), check-in registration (t_2) is completed to record her identity and the alloted room details. Thereafter the guest makes an advanced payment of certain fraction amount towards the charges (t_4). A key-card is generated for the guest subsequently (t_5). The hotel staff prepares the room in the meantime (t_3). Daily room-service is provided on regular basis and whenever asked by the guest (t_7). The checkout step is performed after inspecting the room (t_8) when the guest vacates, generating the invoice (t_9) and clearing the bill (t_{10}).

5.2.2 The New Process

The new process is shown in Fig. 13b. The hotel introduces flexibility in the process by exploiting concurrency between check-in registration (t_2) and room selection (t_1). Second, the advanced payment step (t_4) is deleted. Third, the purchases (t_{11}) made from the hotel gift-shop/restaurant during the stay period are also added into the bill (t_{12}) without having the guest to make immediate payments.

5.2.3 Migration Mappings

The migration problem investigates whether the current guests who have checked-in by the old process can benefit immediately from the new process when the process upgrades during their stay. Since the process is changed structurally starting from its initial part, *structural equivalence* model does not allow any instance to migrate except the initial marking, making the model *unsuitable* for the process. Table 3 enlists migratable markings as per the four trail-based consistency models. It can be noted that, in this case, the mappings by trace equivalence and history equivalence happen to be the same. In purged-trace and purged-history equivalence models, though the mapping sets are the same, the set of acceptable traces is bigger in the latter.

- *Examples of Trace equivalence*: It can be observed that besides empty trace, only the traces (i) $\sigma = t_1$, (ii) $\sigma = t_1t_2$, or (iii) $\sigma = t_1t_2t_3$ can be mapped into the new process by trace equivalence. In Table 3, these old traces listed in the row *Trace*. Equivalent traces (i.e., $\sigma = \sigma'$) $\sigma' = t_1$, $\sigma' = t_1t_2$ and $\sigma' = t_1t_2t_3$ are available in the new net, for which the respective trace equivalent markings are M and M'.

  Deletion of t_4 makes the *trace equivalence* criteria to prohibit migration of traces such as (i) $t_1t_3t_2t_4$, (ii) $t_1t_2t_3t_4$, and (iii) $t_1t_2t_4t_3$, and the traces thereafter, since these traces are not available in the new net.
- *Examples of History equivalence*: Old marking $M = \{p_3, p_7\}$ has a mapping $M' = \{p_4, p_7\}$ in the new net by trace equivalence, for which, $\sigma = \sigma' = t_1t_2t_3$. It can be noted that, the history equivalence model permits mapping of M to M' through one more trace $\sigma = t_1t_3t_2$ that cannot migrate by trace equivalence. For trace $\sigma = t_1t_3t_2$, we have $s(\sigma) = \{t_1, t_2, t_3\}$. With both the new traces $\sigma' = t_1t_2t_3$ and $\sigma' = t_2t_1t_3$, set $s(\sigma') = \{t_1, t_2, t_3\}$, which is the same as set $s(\sigma)$, proving history equivalence.
- *Examples of Purged Trace equivalence*: This equivalence is more suitable for handling the deletion situation since the business model is such that deletions from the traces are tolerated. It can be observed from the table that traces such as $\sigma = t_1t_2t_4t_3$, $\sigma = t_1t_2t_4t_3t_5t_6$ have mappings into the new process as per purged-trace equivalence in spite of deletion of t_4. The set of common transitions in the two nets is $T \cap T' = \{t_1, t_2, t_3, t_5, t_6, t_7, t_8, t_9, t_{10}\}$. For old trace $\sigma = t_1t_2t_4t_3$, we obtain $\sigma_{|T\cap T'} = t_1t_2t_3$. It can be observed that for a new trace $\sigma' = t_1t_2t_3$, we obtain $\sigma'_{|T\cap T'} = t_1t_2t_3$, which satisfies the condition $\sigma_{|T\cap T'} = \sigma'_{|T\cap T'}$ for purged-trace equivalence between the corresponding markings $M = \{p_5, p_7\}$ and $M' = \{p_5, p_7\}$. Similarly, for old trace $\sigma = t_1t_2t_4t_3t_5t_6$ and new trace $\sigma' = t_1t_2t_3t_5t_6$, we have $\sigma_{|T\cap T'} = \sigma'_{|T\cap T'}$ establishing consistency by purged-trace equivalence between the corresponding markings $M = \{p_8\}$ and $M' = \{p_8\}$.
- *Examples of Purged History equivalence*: For old trace $\sigma = t_1t_3t_2t_4$ corresponding to marking $M = \{p_4, p_7\}$, we obtain $s(\sigma_{|T\cap T'}) = \{t_1, t_2, t_3\}$. For new trace $\sigma' = t_1t_2t_3$ we obtain $s(\sigma'_{|T\cap T'}) = \{t_1, t_2, t_3\}$ corresponding to new marking $M' = \{p_4, p_7\}$. Since the condition $s(\sigma_{|T\cap T'}) = s(\sigma'_{|T\cap T'})$ is met, the two

TABLE 3
Migration According to Four Trail-Based Models for Process in Fig. 13

Model	Old Trace (σ)	Old Marking (M)	New Consistent Marking (M')
Trace	ϵ	$\{p_1\}$	$\{p_1\}, \{p_2, p_{12}\}$
	t_1	$\{p_2, p_6\}$	$\{p_2, p_{13}\}$
	t_1t_2	$\{p_3, p_6\}$	$\{p_3, p_{13}\}, \{p_4, p_6\}$
	$t_1t_2t_3$	$\{p_3, p_7\}$	$\{p_4, p_7\}$
History	ϵ	$\{p_1\}$	$\{p_1\}, \{p_2, p_{12}\}$
	t_1	$\{p_2, p_6\}$	$\{p_2, p_{13}\}$
	t_1t_2	$\{p_3, p_6\}$	$\{p_3, p_{13}\}, \{p_4, p_6\}$
	$t_1t_2t_3$, $t_1t_3t_2$	$\{p_3, p_7\}$	$\{p_4, p_7\}$
Purged-trace	ϵ	$\{p_1\}$	$\{p_1\}, \{p_2, p_{12}\}$
	t_1	$\{p_2, p_6\}$	$\{p_2, p_{13}\}$
	t_1t_2	$\{p_3, p_6\}$	$\{p_3, p_{13}\}, \{p_4, p_6\}$
	$t_1t_2t_3$	$\{p_3, p_7\}$	$\{p_4, p_7\}$
	$t_1t_2t_3t_4$	$\{p_4, p_7\}$	$\{p_4, p_7\}$
	$t_1t_2t_3t_4t_5$	$\{p_5, p_7\}$	$\{p_5, p_7\}$
	$t_1t_2t_3t_4t_5t_6$, $t_1t_2t_3t_4t_5t_6t_7$, $t_1t_2t_3t_4t_5t_6t_7t_7$, ... [expression: $t_1t_2t_3t_4t_5t_6(t_7)^*$]	$\{p_8\}$	$\{p_8\}, \{p_{14}\}$
	$t_1t_2t_3t_4t_5t_6(t_7)^*t_8$	$\{p_9\}$	$\{p_9\}$
	$t_1t_2t_3t_4t_5t_6(t_7)^*t_8t_9$	$\{p_{10}\}$	$\{p_{10}\}$
	$t_1t_2t_3t_4t_5t_6(t_7)^*t_8t_9t_{10}$	$\{p_{11}\}$	$\{p_{11}\}$
Purged-history	ϵ	$\{p_1\}$	$\{p_1\}, \{p_2, p_{12}\}$
	t_1	$\{p_2, p_6\}$	$\{p_2, p_{13}\}$
	t_1t_2	$\{p_3, p_6\}$	$\{p_3, p_{13}\}, \{p_4, p_6\}$
	$t_1t_2t_3$, $t_1t_3t_2$	$\{p_3, p_7\}$	$\{p_4, p_7\}$
	$t_1t_2t_3t_4$, $t_1t_2t_4t_3$, $t_1t_3t_2t_4$ [expression: $t_1(t_2t_4\|\|t_3)$]	$\{p_4, p_7\}$	$\{p_4, p_7\}$
	$t_1(t_2t_4t_5\|\|t_3)$	$\{p_5, p_7\}$	$\{p_5, p_7\}$
	$t_1(t_2t_4t_5\|\|t_3)(t_7)^*$	$\{p_8\}$	$\{p_8\}, \{p_{14}\}$
	$t_1(t_2t_4t_5\|\|t_3)(t_7)^*t_8$	$\{p_9\}$	$\{p_9\}$
	$t_1(t_2t_4t_5\|\|t_3)(t_7)^*t_8t_9$	$\{p_{10}\}$	$\{p_{10}\}$
	$t_1(t_2t_4t_5\|\|t_3)(t_7)^*t_8t_9t_{10}$	$\{p_{11}\}$	$\{p_{11}\}$

markings are consistent by purged-history equivalence. In this process, the traces thereafter also have mappings in the new process by purged-history equivalence.

The following sections draw parallels between models in the literature and trail-based models. For non-Petri net based approaches, example cases are constructed, whereas for Petri net based approaches, examples are omitted.

5.3 Trail-Based Approaches in the Literature

5.3.1 The Historic Approach of Ellis et al.

One of the earliest works in this area by Ellis et al. [1] explains the notion of *consistency* on Petri net models of workflows. According to their definition, the *pre-change sequence* ω (i.e., firing sequence) can be used to deal with a change as follows: In the new process, any new firing sequence ω' is carried out only after considering the original firing sequence ω of the old schema as valid. This generates new trace $\omega\omega'$. Such cases are migratable. If such a prefix sequence is not found, the case has no consistent migration. Such cases are continued in the old process instead of migration into the new process. Hence, the notion of consistency relies on the ability to generate the *same trace* till the point of migration. In this case, condition (i) in Definition 2 of trace equivalence is satisfied as follows: In the old net, $\sigma = \omega$, and the new full trace is $\omega\omega'$, implying tokens to be placed just after ω in the new net, resulting in $\sigma = \sigma' = \omega$.

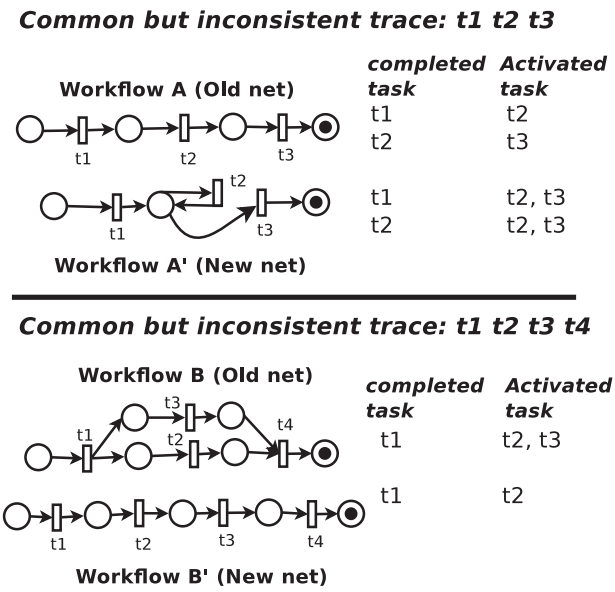


Fig. 14. Examples of trace equivalence violating WIDE consistency.

5.3.2 The WIDE Approach

A workflow schema in the WIDE model [3], [34] consists of tasks, data variables and flow-structuring elements. Their consistency criteria is defined by a scheduler function, which ensures that given an instance, upon completion of each task in it, the set of tasks that are activated must be the same even in the new schema. We observe that this criteria ensures for migratability such that at least one new trace σ' is available that matches with σ, the old trace. However, additionally, the condition requires that the set of enabled tasks at each step is the same in both the nets. Therefore, a drawback of this model is that, even though the two traces are exactly the same (i.e., $\sigma = \sigma'$), structural changes may violate the criteria of their scheduler function, not matching the sets of activated tasks at every step. Two examples of such situations are depicted in Fig. 14. In pair A and A', the common trace is $t_1t_2t_3$, and in pair B and B', the common trace is $t_1t_2t_3t_4$. However, they cannot be considered to be equivalent, since, as noted in the figures, the activation sets do not match in every state prior to the markings. However, such cases are migratable in pure trace-based criteria if we omit the condition of matching the activation sets along the trace. It may be noted that, since activation set and the trace are both involved, the criteria is not a pure trace-based one. It is instead partly structural and partly trace-based.

5.3.3 The ADEPT Approach

Well-structured Marking Nets (WSM-nets) are block structured nets in which explicit control-synchronization between AND-tasks is included. WSM-nets were introduced in the course of ADEPT$_{flex}$ project for modeling workflows [2]. Fig. 15 constructs an example process in their visual notation, and it also provides the corresponding WF-net. Since their notation models task states (completed, activated, running

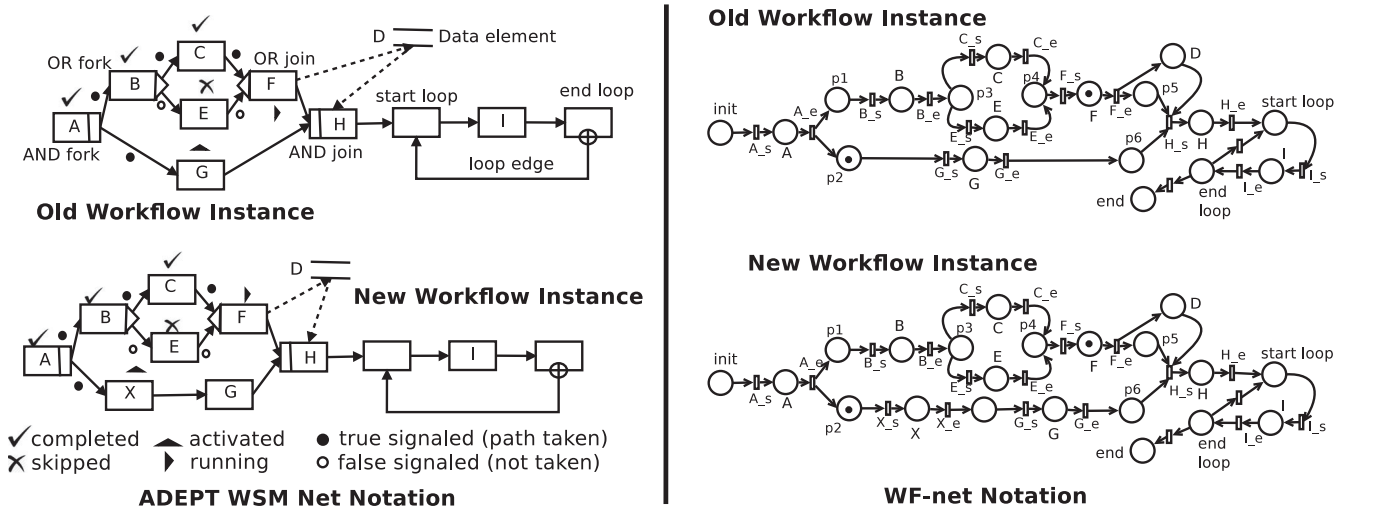


Fig. 15. Notation followed in ADEPT approach.

etc.), the corresponding WF-nets uses the scheme *tasks-as-places* as described in Section 2.2.2. Their approach develops three consistency criteria called traditional compliance, loop-tolerant compliance, and relaxed loop-tolerant compliance, which are discussed below.

Traditional compliance was defined by Rinderle et al. [35], and it is referred to as *compliance* in the work of Reichert et al. [36], [37]. By this criterion, an old instance is consistent with a new schema if the old trace σ can be exactly replayed in the new schema, which is the same as trace equivalence condition ($\sigma = \sigma'$). For old marking $M = \{p_2, F\}$ and new marking $M' = \{p_2, F\}$ in Fig. 15, $\sigma = \sigma' = A_s A_e B_s B_e C_s C_e F_s$, which gives a case of trace equivalence. It can be noted that the exact traces are to be obtained from the WSM-net notation, which includes the logs of tasks completed and paths traversed. This information is extracted from logs, and is not visible in WF-nets.

Loop-purged trace [9] is obtained by discarding earlier loop iterations and keeping only the last loop iteration. In *loop-tolerant compliance*, the loop-purged trace is used. It is an example of a special case of *trace equivalence*, where the shortest old trace from among many possible traces in the case of loops is considered as the old trace σ that is to be compared with the new trace σ'.

A trace becomes *delete-purged* by discarding activities deleted from the old schema. For old trace σ, $\sigma_{|T \cap T'}$ obtains the delete-purged trace if T is the set of tasks in the old net and T' is the set of tasks in the new net. The intersection removes the deleted (and also added) tasks. *Relaxed loop-tolerant compliance* uses both delete-purged and loop-purged traces, which gives a special case of *purged-trace equivalence*, where σ is the shortest trace in the case of loops, and the traces are also purged of the non-common transitions. In Fig. 15, the new instance is also consistent by both loop-tolerant and relaxed loop-tolerant compliance, since there are neither loops nor deleted tasks in the old trace.

5.3.4 The TRAM Approach

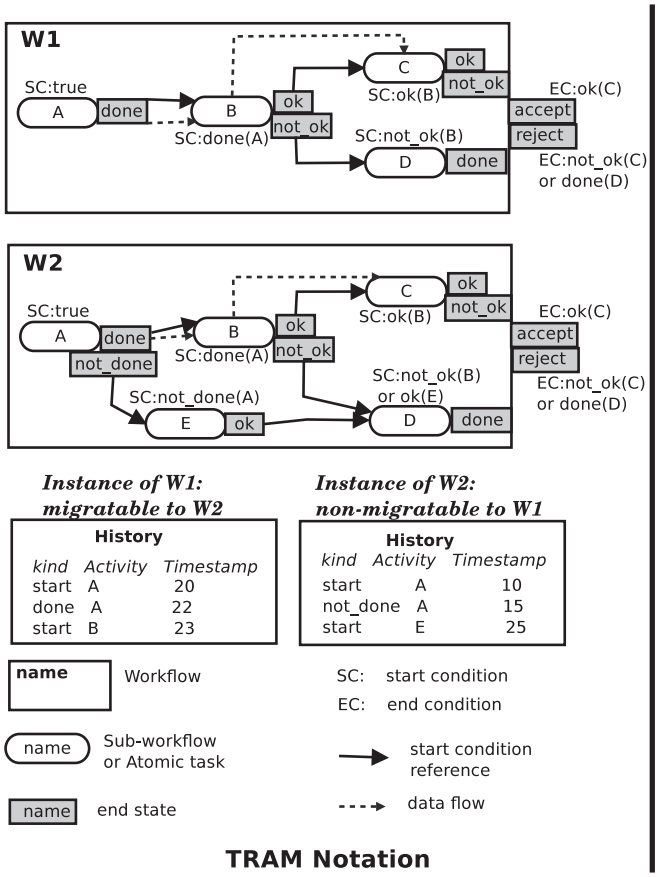
The work by Kradolfer and Geppert [4] presents a workflow schema evolution model using the TRAM workflow approach. Fig. 16 constructs a schema change situation following the notations in this approach. It can be observed in the figure that a workflow is composed of sub-workflows. These are atomic activities. Each of the activities has input-output data parameters, a start condition and one or more end states. The control flow gets represented through logical connectives among the end states of the sub-workflows. A migratable and a non-migratable instance of the old workflow are shown in boxes. The instances are represented in terms of their history. As shown in the figure, the history includes time-stamps, making it a *trace*. The figure also shows WF-nets with markings corresponding to the instances.

The consistency criteria focuses on migrating instances without undoing the already completed tasks from the history, which highlights the conformance to the trace of the running instance. In other words, modifications on the schema are permitted if the existing trace remains a valid trace in the modified schema. Their approach develops a fixed set of change operations and their corresponding *migration conditions* to ensure consistency during dynamic change of the running instance.

For example, the instance of W_1 shown in Fig. 16 is migratable to W_2 in spite of adding activity E and an end-state to existing activity A, since the trace $start\ A$, $done\ A$, $start\ B$ can be replayed (σ in $W_1 = \sigma'$ in $W_2 = A_s A_e B_s$). However, an instance of W_2 shown in the figure is not migratable to W_1 since its trace $start\ A$, $not\_done\ A$, $start\ E$ cannot be replayed in W_1 (σ in $W_2 = A_s A_{eno} E_s$, is not a valid σ' in W_1). It can be observed that task deletion affects trace equivalence if the task is traversed, whereas, adding an alternative-task does not affect trace equivalence.

5.3.5 The Approach of Sun and Jiang

The consistency criteria given by Sun and Jiang [10] for their work on WF-nets defines a compliance and a consistency condition. The new net is *compliant* with the old one when the set of its traces contains all the old traces after purging out from the traces the transitions that are not in both the nets. The consistency condition is given for mapping the markings for migration. A new marking is *consistent* with the old in complaint nets provided that the transitions common to the two nets and appearing in the old trace also appear in the new trace equal number of times, and the



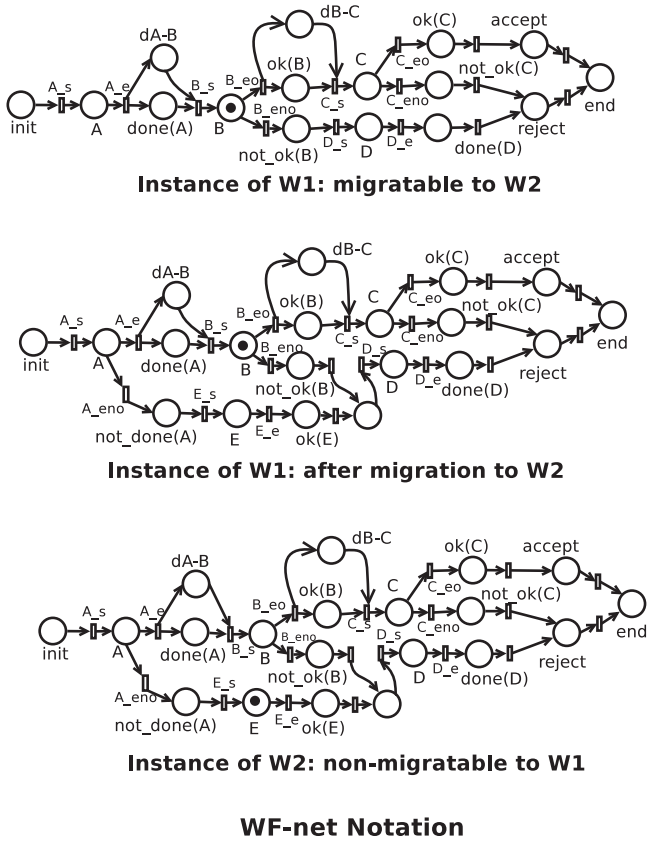


Fig. 16. TRAM workflow model.

vice-versa. As a result, combining the compliance and the consistency conditions generates purged-trace equivalence. They use loop-purged traces for consistency.

5.3.6 The MILANO Approach

The work by Agostini et al. [38] describes the MILANO workflow system, which follows the elementary Petri net model. The workflow system implements computation of state space, which is used for migration mappings. The system supports change operations *sequentialize*, *parallelize* and *swap* as outlined in state-space changes shown in Fig. 17. Sequentialization converts two concurrent tasks into a sequence, parallelization does the opposite, and swapping swaps the order of two tasks in a sequence. For each of the operation, the old state space is shown in the left and the new on the right. Consistent state mappings from the old to the new are shown with dotted arrows. Also, the *unsafe states* in the old, which do not have any consistent mapping into the new, are shown shaded. Consistent mapping is referred to by them as the *univocally* determined image.

The underlying notion of consistency that can be observed from the state mappings is the equal sets of tasks (or state transitions in state-space) from the past. The model therefore aligns with *history equivalence* in the taxonomy. In state-space shown in Fig. 17a, if only t_2 is completed being in state S_3, the instance cannot be mapped, since by history equivalence, history set $\{t_2\}$ is not available in any state in the new net. Therefore, S_3 is shown shaded.

On the other hand, in Fig. 17a, for old state S_1 and new state S_1', $\sigma = \sigma' = \epsilon$. Therefore, we have $s(\sigma) = \phi$ and $s(\sigma') = \phi$ respectively. For states S_2 and S_2', traces $\sigma = \sigma' = t_1$ obtain history sets $s(\sigma) = s(\sigma') = \{t_1\}$. For old state S_4, we have traces $\sigma = t_1 t_2$ or $\sigma = t_2 t_1$ produced by concurrent interleavings reaching the same state, both of which obtain $s(\sigma) = \{t_1, t_2\}$. For new state S_4', the trace is $\sigma' = t_1 t_2$, which obtains $s(\sigma') = \{t_1, t_2\}$. Thus, the condition $s(\sigma) = s(\sigma')$ is met for state-pairs $S_1 - S_1', S_2 - S_2'$ and $S_4 - S_4'$ satisfying *history equivalence* established by set equality. Similar observations can be made for cases (b), (c).

5.3.7 More Examples

The Dilepis approach on dynamic migration validity of WS-BPEL processes by Song et al. [39] allows changes in order of task execution, which aligns with the *history equivalence* model. Sadiq et al. [25] describe the notion of consistency as obtaining the same execution trace (*trace equivalence*) in the context of dynamic workflow change. The work presented by Dias et al. [40] also adopts the same trace equivalence notion of consistency for handling dynamic process changes

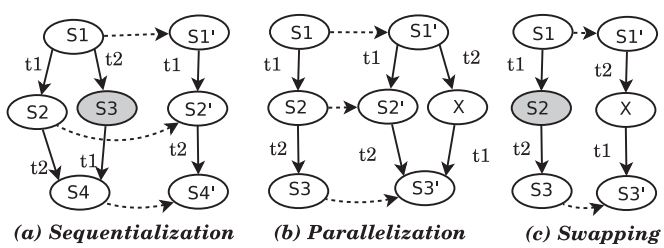


Fig. 17. Mapping from old State-space to new in MILANO approach.

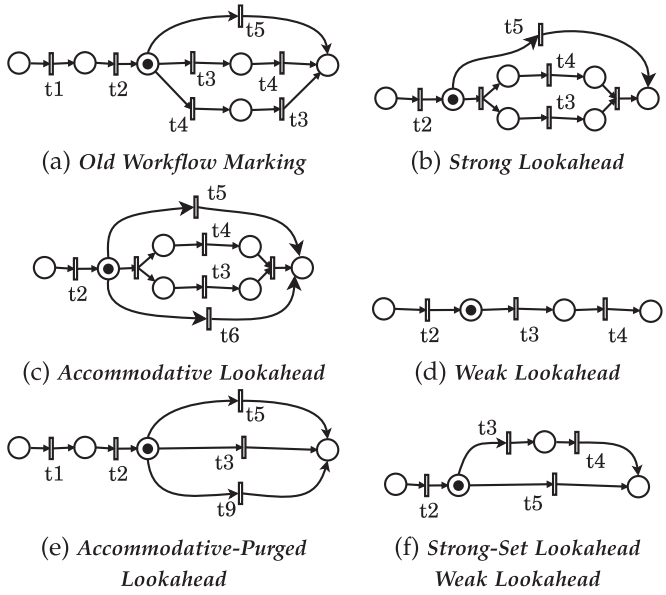


Fig. 18. Consistency by lookahead models.

by pointing out the independence of the consistency notion from the modeling notation. The approach of declarative workflow, developed in DECLARE [13] by Aalst et al. also adopts trace equivalence as their consistency model. The consistency model followed in the pattern-based *Yo-Yo* token transportation approach [41] [11] for block-structured WF-nets uses history equivalence, in which, the set of done tasks in both the nets required to be the same for token migration.

6 Lookahead Consistency

Lookahead models of consistency are conceptually the dual of trail-based models. These models are based on how the remaining part of the workflow is treated in the new process. Past-based consistency derives its motivation in the need to consider as done what is already accomplished, proceeding thereafter by resuming in the new logic. However, as pointed out in [42], besides the models of trail-based consistency, workflow migration in the context of other business goals such as resource optimization, incidental migration, and future compatibilities may require lookahead criteria requiring to consider the future instead of the past.

Lookahead consistency models were formally identified in the workflow migration domain by us in [42]. These models are formulated in terms of set relations among lookahead traces. Manifestations of the lookahead trace-based consistency models have been found in the literature, such as in schema compliance models [8], [10].

6.1 Formulation

The symbols defined earlier in Section 3 are recalled. Let M_E and M'_E be the final markings in old net N and new net N' respectively. Let M be the current marking in N, and M' be a marking in N' that is consistent with M. Let $F_M = \{\sigma | M \xrightarrow{\sigma} M_E\}$, i.e., be the set of all firing sequences starting from marking M and reaching the final marking M_E in N. Similarly, let set $F'_M = \{\sigma' | M' \xrightarrow{\sigma'} M'_E\}$, i.e., be the set of all firing sequences starting from marking M' and reaching the final marking M'_E in N'.

The difference between lookahead sets and the sets considered in history-based models is as follows: A lookahead set is formed as a *set of all lookahead traces*, whereas the history-set in history-based models is the *set of transitions* that actually fired preceding the current marking.

Definition 3. Strong Lookahead consistency *between M and M' is defined by condition $F_M = F'_M$.*

Definition 4. Accommodative Lookahead Consistency *between M and M' is defined by condition $F_M \subseteq F'_M$.*

Definition 5. Weak Lookahead Consistency *between M and M' is defined by the following condition: For non-empty F_M and F'_M, $F_M \cap F'_M \neq \phi$; $F_M = F'_M = \phi$ otherwise.*

6.1.1 Examples

Fig. 18 illustrates each of the lookahead models. The old workflow instance is given in Fig. 18a with lookahead trace set $F_M = \{t_5, t_3t_4, t_4t_3\}$. Fig. 18b gives a *strongly consistent* new marking in a different schema since the lookahead trace set $F'_M = \{t_5, t_4t_3, t_3t_4\}$ obtains $F_M = F'_M$.

Fig. 18c shows a marking in another schema which is consistent with Fig. 18a as per the *accommodative lookahead* consistency model. This marking provides lookahead traces t_5, t_3t_4, t_4t_3 and t_6, which make a superset of the old lookahead traces, i.e., $F'_M = \{t_5, t_3t_4, t_4t_3, t_6\} \supset F_M = \{t_5, t_3t_4, t_4t_3\}$. The marking shown in Fig. 18d has a lone lookahead trace t_3t_4, resulting in $F'_M = \{t_3t_4\}$. It is *weakly consistent* with the marking in Fig. 18a, since $F'_M \subset F_M$.

6.1.2 Two Extended Lookahead Models

Extended lookahead models can be constructed by importing from trail-based models the features of purging and allowing order violation in traces. Two such models that can be created are *accommodative purged lookahead* and *Strong-set based lookahead* models. The first one permits transitions to be removed from traces (purging) in accommodative lookahead model. The second one permits order swapping in traces in strong lookahead model. Examples of these models are now discussed through Figs. 18e and 18f.

Accommodative Purged Lookahead: To reiterate, Fig. 18a shows the old marking. In Fig. 18e, the marking has three lookahead traces each with one transition, i.e., trace t_5, trace t_3 and trace t_9. The first trace qualifies for being a purged form of old lookahead trace t_5. The second trace is a purged form of old lookahead traces t_4t_3 and t_3t_4. Trace t_9 is a newly added trace. The situation satisfies *accommodative purged lookahead* model in which all old traces are facilitated at least in a purged-form in the new net.

Strong-Set Based Lookahead: Fig. 18f depicts a marking with lookahead traces t_3t_4 and t_5. The marking is consistent with the old marking shown in Fig. 18a by both weak lookahead and strong-set based lookahead models. Weak lookahead is satisfied since old traces t_3t_4 and t_5 are available in the new net. *Strong-set based lookahead* is satisfied since all lookahead trace-sets, i.e., $\{t_3, t_4\}$, and $\{t_5\}$ in the old net form the lookahead trace-sets in the new net.

6.2 Situations Suitable for Lookahead Models

Lookahead concepts have been earlier used in the context of web-service interactions [44]. However, in the area of

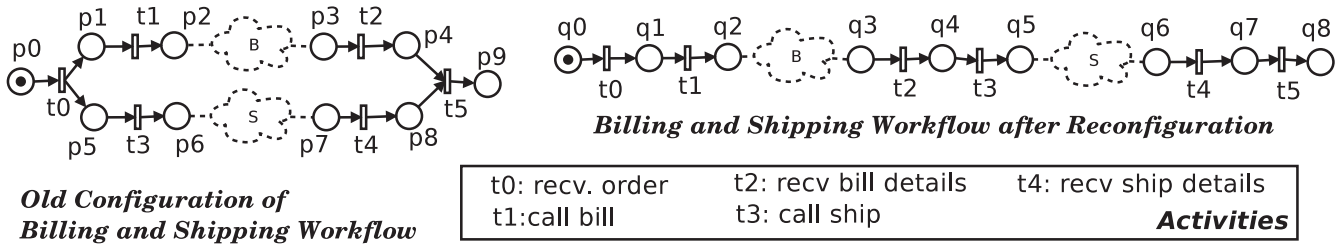


Fig. 19. Workflow reconfiguration requiring weak lookahead consistency [43].

workflow migration, future-based consistency has not been explicitly modeled to our knowledge till the formulation presented by us in [42]. The examples elaborated therein point out that situations such as maintenance of assembly line process of packaging workflow, change in course curriculum, academic transfer process and process re-engineering cases require trace-based lookahead models.

Though the following works on workflow migration do not explicitly mention a formulation of consistency criteria in terms of *lookahead parameters*, we observe that the lookahead models fit into a few situations and definitions discussed therein. This section discusses the appropriateness of applying lookahead models for such examples.

6.2.1 Approach of Mazzara et al.

Fig. 19 depicts a snapshot of the office workflow example discussed by Mazzara et al. [43]. Their workflow is modeled using BPMN, for which, the figure shows an equivalent WF-net model. Clouds labeled B and S abstract billing and shipping sub-workflows. The billing and shipping sub-workflows are executed when an order is to be processed. Both the execution branches are synchronized by activity `send bill&ship details`, which finishes this workflow and transfers control further downstream.

A dynamic workflow reconfiguration changes the process logic from AND to sequence as depicted in the same figure. As further discussed by the authors, the motivation for this change comes from the lack of synchronization (i.e., independence) between the billing and shipping activities in the old process, which often causes delays between the deliveries of the bill and the corresponding goods. To enhance customer satisfaction, the manager then decides to arrange these two parts of the workflow in a sequence. Accordingly, shipping shall be performed after billing.

We observe that given the motivation, the solution to this workflow reconfiguration can be formulated in terms of possible lookahead execution traces, where a particular trace is intended among all possible lookahead traces in the original workflow. For old marking $M = \{p_0\}$ and new marking $M' = \{q_0\}$, we have old set of lookahead traces $F_M = \{t_0t_1Bt_2t_3St_4t_5, t_0t_3St_4t_1Bt_2t_5, \ldots\}$, and new set of lookahead traces $F'_M = \{t_0t_1Bt_2t_3St_4t_5\}$ obtaining $F_M \supset F'_M$, making M migratable by weak lookahead consistency to M'. For old marking $M = \{p_4, p_6\}$ and new marking $M' = \{q_5\}$, we have another migratable mapping $F_M = F'_M = \{St_4t_5\}$. On the other hand, marking $M = \{p_2, p_6\}$ does not have a corresponding new marking with a common lookahead trace, making it non-migratable. Similar observations have been made by them.

The structural change conforming to the new requirement in this example is that of *downsizing*, i.e., the new process can do a lesser interleaving than the old. Therefore, the migration criteria aligns to *weak lookahead consistency*.

6.2.2 Trace-Based Compliance Combined with Consistency

This section discusses how reformulating compliance into consistency reveals a form of lookahead based criteria.

Observable Behavioral Equivalence: Compliance in the approach of Van der Aalst and Basten [8] is defined in terms of traces from initial marking to terminal marking. In addition to compliance, they also define consistency in terms of instance markings. In their approach, compliance between two WF-nets is defined in terms of *observable behavioral equivalence* [45] by applying *branching bisimilarity* [46] between the *initial markings* of the nets. As opposed to trace equivalence, branching bisimilarity is a formulation that establishes state to state equivalence in terms of the succeeding observable transitions from the states. In their approach, branching bisimilarity is applied after blocking (encapsulation) or hiding (abstraction: rewriting them as unobservable τ transitions) additional transitions. Consistency is still required on top of the compliance criteria for achieving migration of markings. Two markings are consistent when they have the same set of marked places (which is the live consistency model discussed in Section 7). The above compliance criteria can be seen as a lookahead policy based on branching bisimilarity that is applied on the initial markings. The same observation holds true for the next example.

Upward Compatibility: The compliance criteria defined by Sun and Jiang [10] specifies schema compliance between two WF-nets through a relation over the net-languages as follows: every single full trace (from the initial marking to terminal marking) in the old net must be mapped to an equivalent full trace in the new net, where the equivalence between two full traces permits purging of transitions that are present in only one of the two nets. After purging the traces, the relation reduces to relation ($Set\ of\ old\ purged\ traces \subseteq Set\ of\ new\ purged\ traces$). It can be noted that an instance level consistency criteria is defined in this approach in terms of purged-trace equivalence.

7 Live Consistency

The most straight-forward migration equivalence notion is the live consistency model, since it solely uses the explicit state representations (marking labels in Petri nets). Consistency mapping by marked places in Petri net models has been discussed frequently in the literature. This notion of consistency relies on producing the *same marking* in the new net as that of the old one. Hence, the *place labels* are the primary parameters to decide consistency. Unlike the

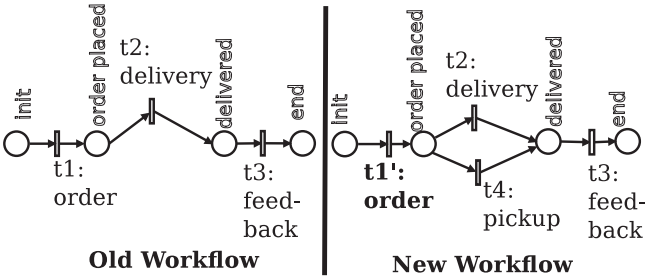


Fig. 20. Old and new workflows for online food purchase.

previously discussed models, live consistency models do not rely on execution and structural information of past or future. On the basis of only the present runtime state attributes, such as place labels the live consistency model is defined. It can be noted that, the live consistency model is not found in non-Petri net formalisms due to absence of an equivalent notion of local states (places).

7.1 Formulation

Using the notations described in Section 3, the WF-net based live consistency model is stated as follows:

Definition 6. Live consistency *between old marking M and new marking M' is defined by the equality $M = M'$.*

It may be recalled from Section 2.2.3 that the equality relation holds when a marking is represented as a set of places such as $\{p_1, p_2, p_3\}$, and not when it is represented as a vector such as $(111)^T$ since the latter uses an implicit labeling which may not be the same in both the nets. As observed in change region based approaches [7], [47], there may be non-transferable markings if they become invalid in the new net. In such cases, the required M' satisfying the above condition cannot be found.

7.2 Process P_4: An Online Food Purchase Process

An online food purchase workflow is described in Fig. 20. In the old workflow, first a purchase order is placed mentioning the delivery address. After the order is placed, the food item is delivered and the customer is required to fill a feedback form to complete the process. The new process introduces an alternative provision for picking up one's order from a pickup point. It also refines the order activity t_1 to t_1' to include the pickup address.

It can be observed from the figure that, every old marking is consistent in the new net as per the live consistency model. For example, for marking $M = \{order\ placed\}$, $M' = \{order\ placed\}$ is consistent since the state is the same in the new net even though their respective traces are different (i.e., old trace is t_1, new trace is t_1'). In this net, all markings are transferable *as-it-is*. This model is suitable for the situations where migration to the same marking carries the same meaning from the user's perspective.

7.3 Appearances of Live Model in the Literature

7.3.1 Approach of Van der Aalst

The earliest adoption of the live consistency model to our knowledge is that of *transfer validity* [7]. Upon migration, the migrated instance marking is consistent. It ensures that the workflow can be completed after migration without a runtime error, if the marking is *valid* in the new net. In process change situation, their approach isolates certain regions (subnets) in the old net, from where, no marking is migratable as per the consistency criteria. Such regions are called *change regions*.

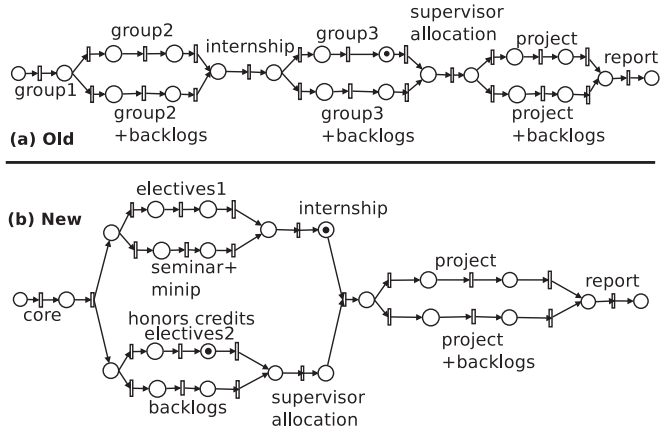


Fig. 21. Change in an academic curriculum process.

7.3.2 Approach of Van der Aalst and Basten

The state transfer rules defined by Van der Aalst and Basten [8] also follow the live model. However, this approach differs from the previous one by relaxing the notion of consistency in terms of equal marking. Two markings M and M' are consistent if $M \cap M' \neq \emptyset$. If the same marking ($M = M'$) is available, it is chosen. Otherwise, state-transfer to a marking $M' \subset M$ or $M' \supset M$ is carried out. In the case of multiple such mappings, the actual choice is left to the user. This consistency model is a relaxed form of the basic definition of live consistency model since it allows migration to a marking that is either subset or superset marking when the old marking is absent in the new net.

7.3.3 Other Approaches

Cicirelli et al. [24] adopted the notion of change region defined over the transfer validity condition for migration of workflows in systems where the execution is decentralized and the execution control is centralized. We have adopted the live consistency model in our work on change region computation for fully distributed processes [47].

8 Business Consistency

The business consistency model is relevant for a migration situation where business knowledge beyond the control-flow needs to be used to define schema compliance. In such a case, state equivalences are established based on parameters external to the workflow. The model of consistency in this case may not obey a well-defined control-flow based or a structure-based rule. An example illustrating such a situation is provided next.

8.1 Process P_5: An Academic Curriculum

The old process in Fig. 21a represents a four-semester Master's program. There are three course groups to be completed sequentially in the first three semesters before

starting the project work. An internship must be completed before joining the third semester. The process ends with submission of the project report. Alternate paths for backlog students are also specified. The revised curriculum emphasizes project, reduces the compulsory course credits and opens an honors alternative. The changed process is shown in Fig. 21b. A migration circular is issued to the students as a part of the change scheme as shown in the box. The migration circular essentially identifies manually the business equivalence between states. In this way, a change can also be handled by relating external parameters.

"Migration Circular"

Credit substitution rules for immediate migration of all Master's students into the new academic curriculum are given as follows:

1) *Group 1 credits are treated as Core credits.*
2) *Group 2 credits are treated as Elective-1 credits.*
3) *Backlog students registered for Group 2 are inducted into Seminar+Mini-project path.*
4) *Group 3 courses are made optional and converted into honors credits.*
5) *Backlog students who do not complete Group 3 credits can not register for the honors credits, instead they will need to complete the backlog credits.*

It can be observed that equivalences between groups of courses are not apparent in terms of the net structure or the control-transitions. Moreover, the activities themselves have been re-engineered, and their equivalences are defined manually outside the workflow structure. The two markings shown in Figs. 21a and 21b are consistent by the business rules formulated according to the circular. The old instance depicts completion of group-1, group-2 (with or without backlogs), internship and group-3. The new instance depicts completion of core courses, electives-1 or seminar+mini-project, internship and honors credits of electives-2, which obtains an equivalent state in the new curriculum.

9 Conclusion

Though the notion of consistency has been used widely in the literature on business process migration, consistency models themselves had remained to be unified. This problem was addressed in this paper through a taxonomy based approach. A taxonomy framework is developed identifying control-flow parameters to obtain variations in the models of consistency. The parameters are *time-frame*, *net structure* and *comparison operators*. The resultant classification is formalized and elaborated over WF-nets.

Structural equivalence is a stricter model, which allows migration only when the changes are in the unvisited part of the process. Trail-based consistency models allow changes anywhere in the net provided that the execution trace till the current state satisfies a past-based criteria. Lookahead models are suitable when the business goals are concerned with future executions. Live consistency model establishes equivalence irrespective of past and future as long as the present state attributes match in both the processes. Business consistency captures business knowledge that cannot be expressed in terms of the above formulations.

Choosing a consistency model for a given context is not merely a computational decision, since it is often driven by business goals of migration. Similarly, the choice of modeling notation can have an impact on the applicability of consistency models, since the information available about a process depends on what is captured in the model. Various modeling notations differ from each other on their expressiveness. To apply a particular consistency model, independent of the modeling notation used to model the process, the information that is required to apply it needs to be made available.

Also, the context of the consistency model influences the algorithms that are needed for performing the actual migration in a dynamic evolution scenario, which requires computation of the new consistent states for every possible old state. A brute-force approach such as exploration through the reachability can be computationally expensive. On the other hand, as and when required, efficient algorithms for the same need to be developed based on the needs of the consistency criteria to be applied and the information that is available from the models and logs.

The research may be seen as a step towards automation of widespread dynamic migration in workflow engines, by enabling formal rules of consistent mapping, algorithms for actual migration and for checking correctness properties. The framework can also be used to extend the classification through combined models and additional parameters.

References

[1] C. Ellis, K. Keddara, and G. Rozenberg, "Dynamic change within workflow systems," in *Proc. Conf. Organizational Comput. Syst.*, 1995, pp. 10–21.
[2] M. Reichert and P. Dadam, "ADEPTflex—Supporting dynamic changes of workflows without losing control," *J. Intell. Inf. Syst.*, vol. 10, no. 2, pp. 93–129, 1998.
[3] F. Casati, S. Ceri, B. Pernici, and G. Pozzi, "Workflow evolution," *Data Knowl. Eng.*, vol. 24, no. 3, pp. 211–238, 1998.
[4] M. Kradolfer and A. Geppert, "Dynamic workflow schema evolution based on workflow type versioning and workflow migration," in *Proc.. IFCIS Int. Conf. Cooperative Inf. Syst. (Cat. No.PR00384)*, 1999, pp. 104–114.
[5] H. A. Reijers, J. Rigter, and W. M. Van der Aalst, "The case handling case," *Int. J. Cooperative Inf. Syst.*, vol. 12, no. 03, pp. 365–391, 2003.
[6] W. M. Van der Aalst, M. Weske, and D. Grünbauer, "Case handling: A new paradigm for business process support," *Data Knowl. Eng.*, vol. 53, no. 2, pp. 129–162, 2005.
[7] W. M. Van der Aalst, "Exterminating the dynamic change bug: A concrete approach to support workflow change," *Inf. Syst. Frontiers*, vol. 3, no. 3, pp. 297–317, 2001.
[8] W. M. Van der Aalst and T. Basten, "Inheritance of workflows: An approach to tackling problems related to change," *Theoretical Comput. Sci.*, vol. 270, no. 1, pp. 125–203, 2002.
[9] M. Reichert, S. Rinderle, and P. Dadam, "On the common support of workflow type and instance changes under correctness constraints," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*. Berlin, Germany: Springer, 2003, pp. 407–425.
[10] P. Sun and C. Jiang, "Analysis of workflow dynamic changes based on petri net," *Inf. Softw. Technol.*, vol. 51, no. 2, pp. 284–292, 2009.
[11] A. Pradhan and R. K. Joshi, "Catalog-based token transportation in acyclic block-structured wf-nets," in *Proc. Int. Workshop Petri Nets Softw. Eng.*, 2015, pp. 287–307.
[12] M. Adams, A. H. Ter Hofstede, D. Edmond, and W. M. Van der Aalst, "Worklets: A service-oriented implementation of dynamic flexibility in workflows," in *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*. Berlin, Germany: Springer, pp. 291–308.

[13] W. M. Van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," *Comput. Sci.-Res. Develop.*, vol. 23, no. 2, pp. 99–113, 2009.
[14] S. Rinderle-Ma and M. Reichert, "Advanced migration strategies for adaptive process management systems," in *Proc. IEEE 12th Conf. Commerce Enterprise Comput.*, 2010, pp. 56–63.
[15] L. Máčel and T. Hruška, "Bringing flexibility into dynamic process change: The process re-execution approach," in *Proc. 4th Int. Conf. Bus. Intell. Technol.*, 2014, pp. 31–38.
[16] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Berlin, Germany: Springer, 2012.
[17] "Business process model and notation version 2.0," 2011. [Online]. Available: www.omg.org/spec/BPMN/2.0/PDF/. Accessed on: Apr. 04, 2015.
[18] W. Van der Aalst, "Business process management as the "Killer App" for Petri nets," *Softw. Syst. Model.*, vol. 14, no. 2, pp. 685–691, 2015.
[19] W. M. Van der Aalst, "The application of Petri nets to workflow management," *J. Circuits Syst. Comput.*, vol. 8, no. 01, pp. 21–66, 1998.
[20] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[21] J. Desel and J. Esparza, *Free Choice Petri Nets*, New York, NY, USA: Cambridge Univ. Press, 1995.
[22] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.
[23] N. Russell, A. H. Ter Hofstede, and N. Mulyar, "Workflow controlflow patterns: A revised view," BPM Center, Brisbane QLD, Australia, Rep. BPM-06–22, 2006.
[24] F. Cicirelli, A. Furfaro, and L. Nigro, "A service-based architecture for dynamically reconfigurable workflows," *J. Syst. Softw.*, vol. 83, no. 7, pp. 1148–1164, 2010.
[25] S. W. Sadiq, "Workflows in dynamic environments-can they be managed?" in *Proc. 2nd Int. Symp. Cooperative Database Syst. Adv. Appl.*, 1999, pp. 27–28.
[26] W. Song and H.-A. Jacobsen, "Static and dynamic process change," *IEEE Trans. Service Comput.*, 2016, doi: 10.1109/TSC.2016.2536025.
[27] S. Rinderle, M. Reichert, and P. Dadam, "Evaluation of correctness criteria for dynamic workflow changes," in *Business Process Management*. Berlin, Germany: Springer, 2003, pp. 41–57.
[28] R. J. van Glabbeek and W. P. Weijland, *Refinement in Branching Time Semantics*. Amsterdam, The Netherlands: Centre for Mathematics and Computer Science, 1989.
[29] S. Rinderle, M. Reichert, and P. Dadam, "Correctness criteria for dynamic changes in workflow systems—a survey," *Data Knowl. Eng.*, vol. 50, no. 1, pp. 9–34, 2004.
[30] G. Vossen and M. Weske, "The WASA2 object-oriented workflow management system," *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 587–589, 1999.
[31] M. Weske, "Formal foundation and conceptual design of dynamic adaptations in a workflow management system," in *Proc. 34th Annu. Hawaii Int. Conf. Syst. Sci.*, 2001, Art. no. 7051.
[32] Z. Qiu and Y. Wong, "Dynamic workflow change in PDM systems," *Comput. Ind.*, vol. 58, pp. 453–463, 2007.
[33] M. Minor, R. Bergmann, S. Görg, and K. Walter, "Towards case-based adaptation of workflows," in *Proc. 18th Int. Conf. Case-Based Reasoning Res. Develop.*, 2010, pp. 421–435.
[34] F. Casati, P. Grefen, B. Pernici, G. Pozzi, and G. Sánchez, *WIDE: Workflow Model and Architecture*. Citeseer, 1996.
[35] S. Rinderle-Ma, M. Reichert, and B. Weber, "Relaxed compliance notions in adaptive process management systems," in *Conceptual Modeling-ER*. Berlin, Germany: Springer, 2008, pp. 232–247.
[36] M. Reichert, S. Rinderle-Ma, and P. Dadam, "Flexibility in process-aware information systems," in *Transactions on Petri Nets and Other Models of Concurrency II*. Berlin, Germany: Springer, 2009, pp. 115–135.
[37] M. Reichert and P. Dadam, "Enabling adaptive process-aware information systems with ADEPT2," *Handbook of Research on Business Process Modeling*, Univ. Ulm, Germany, 2009.
[38] A. Agostini and G. De Michelis, "Simple workflow models," in *Proc. Workflow Manag.: Net-Based Concepts Models Techn. Tools*, Lisbona, Portugal, 1998.
[39] W. Song, X. Ma, H. Hu, Y. Zou, and G. Zhang, "Migration validity of WS-BPEL instances revisited," in *Proc. 16th IEEE Int. Conf. Comput. Sci. Eng.*, 2013, pp. 1013–1020.
[40] P. Dias, P. Vieira, and A. Rito-Silva, "Dynamic evolution in workflow management systems," in *Proc. 14th Int. Workshop Database Expert Syst. Appl.*, 2003, pp. 254–260.
[41] A. Pradhan and R. K. Joshi, "Token transportation in Petri net models of workflow patterns," in *Proc. 7th India Softw. Eng. Conf.*, 2014, pp. 17:1–17:6.
[42] A. Pradhan and R. K. Joshi, "Lookahead consistency models for dynamic migration in workflow processes," in *Proc. Int. Workshop Petri Nets Softw. Eng.*, 2015, pp. 267–286.
[43] M. Mazzara, F. Abouzaid, N. Dragoni, and A. Bhattacharyya, "Design, modelling and analysis of a workflow reconfiguration," in *Proc. Int. Workshop Petri Nets Softw. Eng.*, 2011, pp. 10–24.
[44] S. H. Ryu, F. Casati, H. Skogsrud, B. Benatallah, and R. Saint-Paul, "Supporting the dynamic evolution of web service protocols in service-oriented architectures," *ACM Trans. Web*, vol. 2, no. 2, 2008, Art. no. 13.
[45] R. Milner, *A Calculus of Communicating Systems*. Berlin, Germany: Springer, 1980, vol. 92.
[46] R. J. Van Glabbeek and W. P. Weijland, "Branching time and abstraction in bisimulation semantics," *J. ACM*, vol. 43, no. 3, pp. 555–600, 1996.
[47] A. Pradhan and R. K. Joshi, "Distributed change region detection in dynamic evolution of fragmented processes," in *Proc. Int. Workshop Petri Nets Softw. Eng.*, 2016, pp. 153–172.

Ahana Pradhan is the PhD research scholar in the Department of Computer Science & Engineering, Indian Institute of Technology Bombay. She works in the area of Petri Net models of workflow processes, specializing in dynamic workflow migration algorithms. She has recently joined the Database group in Huawei Technologies, India R&D Center.

Rushikesh Krishnakant Joshi is a professor of computer science & engineering in the Indian Institute of Technology Bombay, Mumbai. He works in the area of software architecture. In specific, he has recently worked on re-engineering of programs for design quality, process modeling approaches for smart grid, approaches for dynamic process migration, and descriptions of interactions in software component architectures.