The Adaptation of a Model of an Artifact-Centric Business Process Instance and Its Validation

Junbao Zhang * and Guohua Liu

School of Computer Science and Technology, Donghua University, Shanghai 201600, China; ghliu@dhu.edu.cn

* Correspondence: junbaozb0451@163.com

Received: 29 November 2018; Accepted: 30 January 2019; Published: 11 February 2019

Abstract: The adaptability of an in-progress business process is an essential requirement for any business process management system in dynamic business process environments. Over the last two decades, the artifact-centric approach for business process management has been evidenced to have higher level of flexibility. However, the adaptation of a model of an artifact-centric business process instance is still inevitable and pervasive due to the complex and ever-changing business environments. Almost all works of artifact-centric business process neglect this issue. To fill this gap, we propose a special business rule called adaptation rule to address the dynamic adaptation problem and describe the adaptation by a global adaptation model. Moreover, we provide a validation mechanism over our proposed adaptation rule of the global adaptation model to guarantee the behavior correctness of the adaptation. Through this validation approach, computing the lifecycle of the global adaptation model can be avoided.

Keywords: business process change; artifact-centric business process; process adaptation; adaptation rule; adaptation correctness

1. Introduction

A critical challenge for the competitive enterprise is its ability to quickly react to business process changes and adequately address them. The adaptability of in-progress business process is an essential requirement for any business process management system in dynamic business process environments [1]. Adaptation of a business process (also referred to as process adaptation) refers to the action of customizing a process instance to make it applicable to a particular situation [2].

In the past two decades, a new business process modeling paradigm has emerged, i.e., artifact-centric business process modeling [3,4]. It has been proved that this approach provides a higher level of flexibility of workflow enactment and evolution [5]. However, few works have developed artifact-centric business process paradigms for the adaptation of business processes. Wang et al. [6] proposed a taxonomy of changes of artifact-centric business processes based on a three-level workflow implementation and an analysis of the impact of changes. Yongchareon et al. [5] addressed the requirements of private artifact-centric business processes change and provided a verification mechanism to preserve the correctness of public business processes in interorganizational business process circumstances based on a view framework. Li and Yu [7] proposed three primitive changes (add, delete and modify) of activities of artifact-centric workflow model in the execution level and proposed guidelines to guarantee the execution of business process instances that are adapted. However, the change of artifact information model is not discussed and theoretical validation of the adaptation is not presented. Eshuis et al. [8] formally reasoned about properties of a process model that are preserved after the model is changed and defined several change operations that preserve certain properties based on Case Management Model and Notation (CMMN), which is an artifact-centric business process model based on the Guard–Stage–Milestone (GSM) model [9,10]. Although flexibility
is naturally considered by artifact-centric business process modeling approach, dynamic adaptation of a model of a business process instance is still inevitable because of the turbulent business process environments [11–13]. However, almost all works in artifact-centric business process community neglect this issue. It is difficult to address this issue, mainly for the following two reasons. First, both data and process should be considered during the adaptation of business process instances. Second, due to the flexible and the declarative properties of artifact-centric business processes, it is hard to guarantee the correctness of the adaptation, i.e., a business process instance that is adapted can still reach an acceptable business goal without deadlock.

Many efforts in activity-centric business process community have addressed this issue. Vasilecas et al. [14] proposed a rule- and context-based approach to dynamically define a business process model. The adaptations of business process instances are implemented by changing contexts, rules and activities. Hu et al. [15] proposed a rule-, context- and goal-based approach for the adaptation of business process instances. Ayora et al. [16] proposed nine change patterns to enable the modeling and evolution of process families at a high level of abstraction. They extended these change patterns to 10 change patterns in [17] and verified that these change patterns can guarantee the structure and the correct behavior of the variants by case study. Policies are used to decide whether a business process instance should be adapted. Nunes et al. [18] proposed the adaptation of business process instances based on context. Patiniotakis et al. [19] used a multi-criteria decision-making approach for the adaptation of event-driven business process instances. Oberhauser [20] proposed adaptation processes to adapt business process instances. Artifact-centric business processes focus on what can be done (goals and progresses) instead of what should be done (tasks or corresponding services) that activity-centric business processes focus on [21]. Nevertheless, due to this difference between activity-centric and artifact-centric modeling approaches, the adaptation methods in activity-centric modeling paradigm are not applicable for artifact-centric business processes.

To solve the issue, we present a taxonomy of changes of artifact-centric business processes and analysis characteristics of these changes. Considering these characteristics, we propose a special business rule called adaptation rule to implement the dynamic adaptation of a model of an artifact-centric business process instance and describe the adaptation by a global adaptation model. Although a global adaptation model can be converted into an artifact-centric business process model, we provide an incremental validation method to verify behavior properties of the business process instances that are adapted by an adaptation rule. As such, the traditional lifecycle composition process of model validation can be avoided. Note that our proposed solution partially solves the problem of adaptation of an artifact-centric business process. How to adapt a business process model to create a new business process model is not discussed in this paper.

2. Artifact-Centric Business Process Model

There are abundant artifact-centric business process models defined formally or informally, but they reach a consensus that business processes are dominated by business artifacts [22]. In this study, we adopt the ACP model with some modifications as it reflects the original idea of artifact-centric approach faithfully [22–24].

**Definition 1** (modified from [23]). An artifact-centric business process model is a triple \( M = (Z, V, R) \), where \( Z \) is an artifact schema. An artifact schema is a set of artifact classes \( Z = \{C_1, C_2, \ldots, C_n\} \), \( C_i \) is an artifact class that defined by a tetrad \((A, s_0, S, S_f)\), where \( A \) is a set of name-value paired attributes; \( s_0 \) is the default start state; \( S \) is a set of states; and \( S_f \subset S \) is the set of final states. \( V \) is a set of services. \( R \) is a set of business rules over \( Z \). A business rule is a triple \((\lambda, \beta, v)\), where \( \lambda \) is the precondition; \( \beta \) is the postcondition; and \( v \in V \) is a service that operates one or more artifacts \( c_1, c_2, \ldots, c_n \) of artifact classes \( C_1, C_2, \ldots, C_n \), respectively.

Artifacts are business-relevant entities that are created and evolved through business processes. An artifact class (hereafter artifact) defines an information model and its lifecycle [3]. A business
rule associates service(s) with artifact(s) in a Condition–Action style. Each business rule describes which service is invoked and which state of an artifact is changed under what precondition and postcondition [23]. λ and β are defined by quantifier free first order logic formula. In the formula, there are two types of proposition over schema Ζ: (1) state proposition (the instate predicate); and (2) attribute proposition (the defined predicate and scalar comparison).

Example 1.

We use a purchase business process [23] to illustrate the concept of the artifact-centric business processes. Its artifact-centric representation is described in Figure 1.

A purchase business process consists of an information model (Figure 1, top left) and a lifecycle model (Figure 1, top right). There are three artifacts, namely Order, Payment and Shipment, in the process. A customer can instantiate the process by selecting some products and creating an order artifact instance, and follows the process to prepay the order. At this moment, a payment artifact instance is created and move to the prepaid state. A seller can accept the order and get ready to ship the order after the order has been packed, and follow the process to ship the order. Now, a shipment artifact instance is created. After the seller ships the order, the customer can receive the order and follow the process to finish the payment. Eight business rules and eight services work together to update the information model of the process and move three artifacts from one state to another along their lifecycles.

There is an example of business rules in the purchase process (Table 1). In Table 1, state proposition instate(O, order_created) holds if state of O is order_created, i.e., O.curState = order_created; and attribute proposition defined(O.custName) holds if custName of O has been defined.

![Figure 1. Artifact-centric representation of a purchase business process.](image-url)

**Table 1.** An example of business rules in the purchase process (Figure 1).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Precondition</th>
<th>Service</th>
<th>Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1: Customer C requests to make an Order O with order items OIS.</td>
<td>instate(O, init)</td>
<td>create(O, C, OIS)</td>
<td>instate(O,order_created) ∧ defined(O.custName) ∧ defined(O.address) ∧ defined(O.grandTotal) ∧ defined(O.orderItem)</td>
</tr>
</tbody>
</table>
Attribute change may add an attribute to an artifact, delete an attribute from an artifact and modify an attribute of conditions. A sub condition could be added to a business rule. A business rule may be modified. Business rule change includes rule action change, rule condition change and rule existence change. Rule action change refers to replacement of an action. Rule condition change refers to modification of conditions. A sub condition could be added to λ or β of a business rule. A sub condition may be deleted from λ or β of a business rule. A sub condition of λ or β of a business rule may be modified.

3. Adaptation of a Model of an Artifact-Centric Business Process Instance

In this section, we present a taxonomy of changes of an artifact-centric business process and discuss characteristics of these changes. Considering characteristics of these changes, we propose a global adaptation model for the adaptation of a model of an artifact-centric business process instance.

3.1. Taxonomy of Changes of an Artifact-Centric Business Process

According to key components of an artifact-centric business process, we present the taxonomy of changes of an artifact-centric business process in Figure 2. These changes are inspired from the work of Wang et al [6]. However, we focus on model level changes.

![Figure 2. The taxonomy of changes of an artifact-centric business process.](image-url)

**Table 1. Cont.**

<table>
<thead>
<tr>
<th>r2: Customer C prepays an Order O through payment P.</th>
</tr>
</thead>
<tbody>
<tr>
<td>precondition</td>
</tr>
<tr>
<td>service</td>
</tr>
<tr>
<td>postcondition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r3: A seller collects order items of an Order O and pack the Order.</th>
</tr>
</thead>
<tbody>
<tr>
<td>precondition</td>
</tr>
<tr>
<td>service</td>
</tr>
<tr>
<td>postcondition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r4: Create Shipment S according to Order O.</th>
</tr>
</thead>
<tbody>
<tr>
<td>precondition</td>
</tr>
<tr>
<td>service</td>
</tr>
<tr>
<td>postcondition</td>
</tr>
</tbody>
</table>
Business rule existence change includes add a business rule, delete a business rule, merge business
rules and split a business rule.

Service change includes: add a service, delete a service and modify a service.

Suppose we want to adapt a business process instance from model M to model M’. According to
the taxonomy, we can see that artifacts of M and M’ may have following differences.

From view of information models of artifacts:
(1) An artifact of M may map to one artifact of M’ if attribute changes are used alone. Nevertheless,
   the cardinality of attributes may be different if changes of add an attribute or delete an attribute
   are used.
(2) An artifact of M may map to multiple artifacts of M’ if changes of split an artifact are used alone.
(3) Multiple artifacts of M may map to one artifact of M’ if changes of merge artifacts are
   used alone.
(4) Multiple artifacts of M may map to multiple artifacts of M’ if combination of changes is used.
(5) Artifacts of M may map partially to artifacts of M’ if change of add an artifact or delete an
   artifact is used.
(6) Attributes of artifacts of M map partially to attributes of artifacts of M’ while considering
   Characteristics (1)–(5).

Similarly, from view of lifecycles of artifacts, relationships of artifacts between M and M’ may
be one-to-one, one-to-many, many-to-one, many-to-many and partial mapping. Usually, mappings
between states do not exist if changes of merge artifacts and split an artifact are used because both the
cardinality of states and the names of states may be changed due to these changes. As a result, states
of artifacts of M map partially to states of artifacts of M’.

From former analysis, we can see that relationships of artifacts between two models are complex
from perspectives of both information models of artifacts and lifecycles of artifacts. Because data and
states of artifacts of business process instances that should be adapted cannot be decided according to
relationships of artifacts between M and M’, it is hard to decide when and how to adapt a model of a
business process instance automatically. This circumstance is common for most changes in Figure 2,
for example, add an attribute, add an artifact, merge artifacts, and split an artifact.

Note that these changes are interleaving [6]; coordination of changes of artifacts, business rules
and services should be done to generate a new version of business process that is modeled as M’. How
to generate M’ by these changes is omitted, as it is not the interests of this paper. Next, we present our
method for the adaptation of a model of an artifact-centric business process instance semi-automatically.

3.2. The Proposed Global Adaptation Model

There are two operations while implementing the adaptation of a model of an artifact-centric
business process instance when considering perspectives of data and process of an artifact-centric
business process. Suppose the instance is adapted from model M to model M’. One operation is to
convert the information models of artifacts of the instance from M to M’. The other operation is to set
the converted artifact instances with proper data (attributes with proper values and artifact instances
with proper states).

Considering complex relationships of artifacts of M and M’ while adapting a business process
instance from M to M’, we propose special user-defined business rules called adaptation rule for
deciding when and how to implement the adaptation. After an adaptation rule is defined and
deployed to the business process system, a business process instance will be adapted dynamically by
the adaptation rule if the precondition and the postcondition of the adaptation rule are satisfied. We
use a global adaptation model to describe the adaptation.

**Definition 2.** A global adaptation model is a tetrad $\Gamma = (M, M', v_a, r_a)$, where M and M’ are two artifact-centric
business process models; $v_a$ is a service that implement the two operations while adapting a business process
instance of M to M’; and $r_a$ is a special business rule called adaptation rule that is a tuple $(\lambda_M, \lambda_M', \beta_M, \beta_{M'})$.
\( \nu_0 \), where \( \lambda_M \) is the sub condition of \( r_0 \) on \( M \) in the precondition of \( r_0 \); \( \lambda_{M'} \) is the sub condition of \( r_0 \) on \( M' \) in the precondition of \( r_0 \); \( \beta_M \) is the sub condition of \( r_0 \) on \( M \) in the postcondition of \( r_0 \); and \( \beta_{M'} \) is the sub condition of \( r_0 \) on \( M' \) in the postcondition of \( r_0 \). \( \lambda_M \land \lambda_{M'} \) formulates the precondition of \( r_0 \); and \( \beta_M \land \beta_{M'} \) formulates the postcondition of \( r_0 \).

In Definition 2, \( \nu_0 \) is a service that implements a partial function \( f: M.Z.C_1 \times \ldots \times M.Z.C_m \rightarrow M'.Z.C_1 \times \ldots \times M'.Z.C_m, m = |M.Z| \) and \( n = |M'.Z| \). \( \lambda_M \) takes formula \( \text{instate}(M'.Z.C_1, s_{0f}) \land \ldots \land \text{instate}(M'.Z.C_m, s_{0f}) \) (\( n = |M'.Z| \)) as default. Note that there is a special final state \( s_{gf} \) in \( C_i.S_f \), where \( C_i \in M.Z \). State proposition in \( \beta_M \) takes formula \( \text{instate}(M.Z.C_1, s_{gf}) \land \ldots \land \text{instate}(M.Z.C_m, s_{gf}) \) (\( m = |M.Z| \)) as default. If state of an artifact instance was logged as \( s_{gf} \), it means that the corresponding process instance of this artifact instance has been adapted.

Given a global adaptation model \( \Gamma \), it can be mapped to an artifact-centric business process model \( M_{\Gamma} \), where \( M_{\Gamma}.Z = \Gamma.M.Z \cup \Gamma.M'.Z, M_{\Gamma}.V = \Gamma.M.V \cup \Gamma.M'.V \cup \{ \text{instate}(\Gamma.r_0) \} \) and \( M_{\Gamma}.R = \Gamma.M.R \cup \Gamma.M'.R \cup \{ \text{instate}(\Gamma.r_0) \} \). Compare \( M_{\Gamma} \) with \( M \) and \( M' \), we can see that only \( \Gamma.r_0 \) needs to be defined as we suppose \( \Gamma.r_0 \) has been provided. From definition of an adaptation rule, we can see that an adaptation rule dominates artifacts of two business process models, instead of artifacts of the same business process model as compared with general business rules. Another difference is that an adaptation rule synchronizes all artifacts of both business process models as contrast to part of artifacts within one business process model. In this paper, we suppose \( \Gamma.M.Z \cap \Gamma.M'.Z = \emptyset \).

Example 2.

It is hard to find a real-world business process to illustrate all circumstances. In this example, we use two domain independent process models to illustrate the concept of our global adaptation model (Figure 3).

Compare two business process models \( M \) and \( M' \) in Figure 3. two artifacts \( \text{Arti}A \) and \( \text{Arti}B \) are merged as artifact \( \text{Arti}AB \). Further attribute changes are used on \( \text{Arti}AB \), where the changes are delete attributes \( \text{Attric} \) and \( \text{Attrig} \) and add attribute \( \text{Attril} \). Artifact \( \text{Arti}C \) splits as two artifacts \( \text{Arti}C1 \) and \( \text{Arti}C2 \). Artifact \( \text{Arti}D \) is deleted and a new artifact \( \text{Arti}E \) is added in model \( M' \).

Figure 3. Artifact-centric representation of a global adaptation model.
In Figure 3, we can see that, if adaptation rule $r_{21}$ is evaluated, a business process instance of $M$ will be adapted to model $M'$ and continues to run according to definition of $M'$. Note that some complementary sub services may be incorporated in $v_{21}$ to remedy losses and supplement gains of the business process instance due to the adaptation. For instance, a seller returns postage fee if type of shipment of an order is changed from express to self-pick-up in a purchasing process. On the contrary, a customer pays postage fee if type of shipment of an order is changed from self-pick-up to express.

From Example 2, we can see that an adaptation rule combines two business process models as one brand new business process model $M'$. Next, we validate a global adaptation model through its corresponding artifact-centric business process model.

The adaptation rule of Example 2 is defined in Table 2.

### Table 2. The adaptation rule of Example 2.

<table>
<thead>
<tr>
<th>Adaptation Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_M$</td>
<td>$\text{instate}($ArtiA, $s_3$) $\land$ $\text{instate}($ArtiB, $s_6$) $\land$ $\text{instate}($ArtiC, $s_9$) $\land$ $\text{instate}($ArtiD, $s_{13}$) $\ldots$</td>
</tr>
<tr>
<td>$\lambda_{M'}$</td>
<td>$\text{instate}($ArtiAB, init) $\land$ $\text{instate}($ArtiC1, init) $\land$ $\text{instate}($ArtiC2, init) $\land$ $\text{instate}($ArtiE, init)</td>
</tr>
<tr>
<td>$\nu_a$</td>
<td>adapt($\text{Arti}A\text{a}$, $\text{Arti}B\text{b}$, $\text{Arti}C\text{c}$, $\text{Arti}D\text{d}$, $\text{Arti}AB\text{ab}$, $\text{Arti}C\text{1}$, $\text{Arti}C\text{2}$, $\text{Arti}E\text{e}$, $\ldots$)</td>
</tr>
<tr>
<td>$\beta_M$</td>
<td>$\text{instate}($ArtiA, $s_9$) $\land$ $\text{instate}($ArtiB, $s_6$) $\land$ $\text{instate}($ArtiC, $s_9$) $\land$ $\text{instate}($ArtiD, $s_{13}$) $\ldots$</td>
</tr>
<tr>
<td>$\beta_{M'}$</td>
<td>$\text{instate}($ArtiAB, $s_{17}$) $\land$ $\text{instate}($ArtiC1, $s_{20}$) $\land$ $\text{instate}($ArtiC2, $s_{25}$) $\land$ $\text{instate}($ArtiE, $s_{27}$) $\ldots$</td>
</tr>
</tbody>
</table>

### 4. Verification of a Global Adaptation Model

Verification of a global adaptation model $\Gamma$ is crucial to guarantee the correctness of execution of a business process [24]. In this section, we use the soundness of a global adaptation model $\Gamma$ to guarantee that business process instances of $\Gamma.M$ that are adapted by $\Gamma.x$ can reach business goals without deadlock. Considering flexible and declarative properties of artifact-centric business process, we use the correctness of an adaptation rule to guarantee that a business process instance that is adapted by the adaptation rule can reach an acceptable business goal.

#### 4.1. Soundness of an Artifact-Centric Business Process Model

Before presenting the soundness of a global adaptation model, we introduce the soundness of an artifact-centric business process model as a foundation in this subsection.

It is known that verifying artifact behaviors is undecidable when the ability of creating new artifacts is considered [24]. However, the verification is decidable when artifact behaviors, i.e., the lifecycle of an artifact, are represented by finite states [5]. Differently, we generate the lifecycle of an artifact-centric business process directly as opposed to pair-wise composition of the lifecycles of two artifacts.

**Definition 3** (modified from [5]). Given an artifact-centric business process model $M$, the lifecycle of an artifact $C$ ($C \in M.Z$), denoted as $L_C$, is a triple $(s_0, V_s, E_t)$, where $s_0$ is the default start state of $C$; $V_s = C.S \cup \{s_0\}$; and $E_t \subset V_s \times M.R \times V_s$ is the set of transitions.

Given a transition $(s_x, r, s_y) \in E_t$, it means that state of an artifact will be updated from $s_x$ to $s_y$ if business rule $r$ is evaluated.

**Definition 4.** Given an artifact-centric business process model $M$, the lifecycle of $M$, marked as $L_M$, is a triple $(s_0, V_s, E_t)$, where $s_0 = (L_{C_1}, s_{01}, \ldots, L_{C_n}, s_{0n})$, $n = |M.Z|$, and $C_i \neq C_j$ ($i \neq j$). $V_s \subset L_{C_1}, V_s \times \ldots, \times L_{C_n}, V_s$. Given a state $s \in V_s$, $s$ is a final state of $L_M$ if $s[i]$ is a final state of $L_{C_i}$ for $i$ in $[1,n]$. $E_t \subset V_s \times R \times V_s$ is a set of transitions, where $R$ is a set of business rules that transformed from $M.R$.

Given a lifecycle $L_M$ and a state $s \in L_M.V_s$, we call $s$ as a business goal of $L_M$ if there is a business process instance of $M$ that is completed, where $C_i \in M.Z$ and $\text{instate}(C_i,s[i])$ ($1 \leq i \leq |s|)$ holds for the
process instance. If not specified clearly, the sequence of states in \( s \) are determined by lexicography order of artifact names, which take marks of artifacts as default. Note that a business goal of \( L_M \) is equivalent to a final state of \( L_M \) if \( M \) is a general artifact-centric business process model; and not equivalent if \( M \) is an artifact-centric business process model that derives from a global adaptation model. In the second circumstance, a final state of \( L_M \) must be a business goal of \( L_M \) but a business goal of \( L_M \) may not be a final state. For instance, state \( \langle r_0, s_0, s_0, s_0, s_0, s_0, s_0 \rangle \) is a business goal, not a final state of \( L_{M_1} \) (\( \Gamma \) is the global adaptation model of Figure 3). We call lifecycle \( L_M \) as the lifecycle of \( \Gamma \).

Let \( cv(r, s_x, s_y) \) be a transformation function. \( cv(r, s_x, s_y) \) returns a new business rule denoted as \( r[s_x, s_y] \), where sub condition of state proposition in the precondition of \( r \) is substituted by \( \text{instate}(C_M, s_x) \) and sub condition of state proposition in the postcondition of \( r \) is substituted by \( \text{instate}(C_M, s_y) \). \( C_M \) is a virtual artifact that composites all the artifacts of model \( M \), where their states appear in \( s_x \). Given an artifact-centric business process model \( M \), \( L_M \) can be generated iteratively from \( L_M.s_0 \).

**Example 3.**

In this example, we use aforementioned purchase business process that is modeled as \( M_P \) to illustrate the process of generating the lifecycle of \( M_P \), which is marked as \( L_{M_P} \). The initial state of the lifecycle is \( \langle \text{instate}(M_P, s_x) \rangle \) for there are three artifacts in the model. At present, only precondition of \( r_1 \) holds. Because \( r_1 \) induces \( \text{Order} \) from \( \text{init} \) to \( \text{order\_created} \), state of a process instance will change from \( \langle \text{init}, \text{init}, \text{init} \rangle \) to \( \langle \text{order\_created}, \text{init}, \text{init} \rangle \) if \( r_1 \) is evaluated. To make \( r_1 \) suitable for the lifecycle, the sub condition of state propositions of the precondition and the postcondition of \( r_1 \) are substituted by \( \text{instate}(C_{M_1}, \text{init}, \text{init}, \text{init}) \) and \( \text{instate}(C_{M_P}, \text{order\_created}, \text{init}, \text{init}) \), respectively. As such, it generates a new business rule, i.e., \( cv(r_1, \text{init}, \text{init}, \text{init}) \), \( \langle \text{order\_created}, \text{init}, \text{init} \rangle \). Consequently, transition \( \langle \text{init}, \text{init}, \text{init} \rangle \rightarrow \langle \text{order\_created}, \text{init}, \text{init} \rangle \) belongs to \( L_{M_P}.Et \) and state \( \langle \text{order\_created}, \text{init}, \text{init} \rangle \) belongs to \( L_{M_P}.Vs \). Similarly, the lifecycle of \( M_P \) is derived iteratively until \( L_{M_P}.Vs \) and \( L_{M_P}.Et \) are fixed. Finally, \( L_{M_P} \) is generated as Figure 4.

**Figure 4.** The lifecycle of the purchase business process model \( M_P \) (Figure 1).

Next, we introduce the soundness of a lifecycle with some modifications [5]. Given a lifecycle \( L \), \( L \) is sound if and only if following two conditions are satisfied: (1) business rule \( r \in \text{rule}(L) \) such that \( r \) induces one and only one transition, i.e.,

\[
\forall r \in \text{rule}(L), (s_x, r, s_y) \in L.E_t \land (s_m, r, s_n) \notin L.E_t;
\]

and (2) for every none final state \( s \) in \( L.V_s \), \( s \) can be reached from \( L.\text{so} \) and \( s \) can reach a final state, i.e.,

\[
\forall s \in L.V_s \land \neg f(s), \exists s_f, s_f \in L.V_s \land f(s_f) \land L.s_0 \Rightarrow^* s \land s \Rightarrow^* s_f.
\]
Theorem 1. Given a global adaptation model \( M \), prove these two parts separately. Suppose the lifecycle of an artifact-centric business process model, there are three types of transitions in \( \Gamma \) of a lifecycle, \( M \) is sound if and only if Formulas (1) and (2) hold for the lifecycle of \( M \); and (2) \( s_f \) is a reachable state from \( s_L \). Condition (2) implies connectivity and goal-reachable of the lifecycle of an artifact or a business process model.

Definition 5. Given an artifact-centric business process model \( M \), \( M \) is sound if and only if \( L_M \) is sound.

If an artifact-centric business process model \( M \) is sound, all business process instances that are regulated by \( M \) can reach some final states of \( L_M \) without deadlock [5].

4.2. Verification

In this sub-section, we define the soundness of a global adaptation model \( \Gamma \) and verify it. We prove that the global adaptation model is sound if the adaptation rule in the model is sound. Here, we suppose \( \Gamma M \) and \( \Gamma M' \) are sound. Furthermore, we use an adaptation goal to define the acceptable business goals for business process instances that are adapted by an adaptation rule. Then, we use the correctness of the adaptation rule to guarantee that the business process instances can reach some acceptable business goals.

Definition 6. Given a global adaptation model \( \Gamma \), we say that \( \Gamma_rQ \) is sound if and only if: (1) \( s_y \) is a reachable state from \( s_L, s_0 \) in the lifecycle of \( \Gamma \); and (2) \( s_f \) is a reachable state from \( s_L, s_0 \) in the lifecycle of \( \Gamma \). Given a global adaptation model \( \Gamma \), let the artifact-centric business process model derived from \( \Gamma M \) as \( M_t \); \( \Gamma \) is sound if \( M_t \) is sound.

\[ s_y \text{ is a state of } M \text{ according to sub condition } \Gamma r_y \lambda M \text{, and } s_f \text{ is a state of } M' \text{ according to sub condition } \Gamma r_f \beta M'. \]

For example, \( s_x = (s_3, s_6, s_9, s_{13}) \) and \( s_Y = (s_{17}, s_{20}, s_{25}, s_{27}) \) in Example 2 (Figure 3).

Theorem 1. Given a global adaptation model \( \Gamma \), \( \Gamma \) is sound if \( \Gamma M \) and \( \Gamma \) are sound.

Proof. By definition of the soundness of a global adaptation model, proof of Theorem 1 is equivalent to prove that \( M_t \) is sound if \( \Gamma M \) and \( \Gamma \) are sound. By Definition 5 and definition of the soundness of a lifecycle, \( M_t \) is sound if and only if Formulas (1) and (2) hold for the lifecycle of \( \Gamma M_t \), i.e.,

\[
\forall r \text{ rule}(L_{M_t}), (s_x, r, s_y) L_{M_t}, E_t \land (s_m, r, s_n) \notin L_{M_t}, E_t, \quad (3)
\]

\[
\forall s \text{ L}_{M_t}, V_s \land \land \exists f, s_f L_{M_t}, V_s \land \land \Rightarrow \Rightarrow s_f. \quad (4)
\]

Therefore, Theorem 1 holds if we can confirm the following hypothesis: Formulas (3) and (4) hold if \( \Gamma M \), \( \Gamma M' \) and \( \Gamma \) are sound. This hypothesis can be divided as two parts: (I) Formula (3) holds if \( \Gamma M \), \( \Gamma M' \) and \( \Gamma \) are sound; and (II) Formula (4) holds if \( \Gamma M \), \( \Gamma M' \) and \( \Gamma \) are sound. Next, we prove these two parts separately. Suppose the lifecycle of \( \Gamma M \) is \( L_M \); the lifecycle of \( \Gamma M' \) is \( L_{M'} \); and \( \Gamma \) is sound if \( M_t \) is sound. For convenience of description.

Proof of Hypothesis Part (I). To prove Hypothesis Part (I), we divide transitions of \( L_{M_t} \) as three types. Rules appear in \( L_{M_t} \), i.e., rule(L_{M_t}) can be divided as three sub sets, rules derive from \( \Gamma M \), marked as \( R_M \); rules derive from \( \Gamma M' \), marked as \( R_{M'} \). According to construction process of the lifecycle of an artifact-centric business process model, there are three types of transitions in \( L_{M_t} \) that correspond to these three sub sets of rules. These transitions have the form of \( (s_m, g\circ s_m', \alpha, r, s_m' \circ s_m'^{\alpha}, \Delta) \) and \( (s_m, g\circ s_m', \alpha, r, s_m' \circ s_m'^{\alpha}, \Delta) \), respectively. \( s_m, g\circ s_m' \) and \( s_m' \) are two states of \( L_M \) and \( s_m'^{\alpha} \) is the initial state \( L_M, s_0 \). Notation \( \circ \) represents concatenation.
operation. \( s_{m, \text{init}, \text{gf}} \) is either the initial state or a state \( s_{m, \text{gf}} \) of \( L_{M_{r}} \), where \( s_{m, \text{gf}}[i] = s_{gf} \ (1 \leq i \leq |M.Z|) \). \( s_{m, \text{a}} \) and \( s_{m, \text{b}} \) are two states of \( L_{M_{r}} \). According to Definition 4 and the construction process of the lifecycle \( L_{M_{r}} \) (omitted in this paper), we have the following two facts:

\[
(s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}}, r, s_{m, \text{b}} \circ s_{m, \text{init}, \text{b}}) \ L_{M_{r}}.E_{1} \equiv (s_{m, \text{a}}, r[s_{m, \text{a}}], s_{m, \text{b}}) \ L_{M_{r}}.E_{1}, \tag{5}
\]

\[
(s_{m, \text{init}, \text{gf}}, r', s_{m, \text{init}, \text{a}} \circ s_{m, \text{init}, \text{b}}) \ L_{M_{r}}.E_{1} \equiv (s_{m, \text{a}}, r'[s_{m, \text{a}}], s_{m, \text{b}}) \ L_{M_{r}}.E_{1}. \tag{6}
\]

Since \( \Gamma.M \) and \( \Gamma.M' \) are sound, the following two formulas

\[
\forall r \ R_{M_{r}} \ (s_{x}, r, s_{y}) \ L_{M_{r}}.E_{I} \land (s_{m}, r, s_{n}) \notin L_{M_{r}}.E_{I} \tag{7}
\]

\[
\forall r \ R_{M'_{r}} \ (s_{x}, r, s_{y}) \ L_{M'_{r}}.E_{I} \land (s_{m}, r, s_{n}) \notin L_{M'_{r}}.E_{I} \tag{8}
\]

hold while considering Formulas (1), (5) and (6). Since \( \Gamma.r_{a} \) is sound and transition \((s_{m,x} \circ s_{m', \text{init}}, \Gamma.r_{a}, s_{m, \text{gf}} \circ s_{m', y})\) is unique in \( L_{M_{r}} \), we have that

\[
\forall r \ R_{M_{r}} \ (s_{x}, r, s_{y}) \ L_{M_{r}}.E_{I} \land (s_{m}, \Gamma.r_{a}, s_{n}) \notin L_{M_{r}}.E_{I}. \tag{9}
\]

\( s_{m,x} \) is an state of \( L_{M_{r}} \) according to \( \Gamma.r_{a} \Delta M_{r} \), denoted as \( \text{prestate}(\Gamma.r_{a}) \); \( s_{m', y} \) is a state of \( L_{M'_{r}} \) according to \( \Gamma.r_{a} \Delta M'_{r} \), denoted as \( \text{poststate}(\Gamma.r_{a}) \). Here, we confirm Hypothesis Part (I) when transitions confine to the same type by Formulas (7)–(9). Next, we release this restriction. Since \( \Gamma.M.Z \cap \Gamma.M'.Z = \emptyset \), we have \( r_{x} \neq r_{y}, r_{x} \in \Gamma.M.R \) and \( r_{y} \in \Gamma.M'.R \). Therefore, we can have \( r \neq r' \), where \( r \) and \( r' \) belong to different rule sets \( R_{M_{r}}, R_{M'_{r}} \) and \( [\Gamma.r_{a}] \). Here, it is sufficient to show that Hypothesis Part (I) holds while considering Formulas (7)–(9). Proof of Hypothesis Part (I) is complete.

Proof of Hypothesis Part (II). To prove Hypothesis Part (II), we divide lifecycle \( L_{M_{r}} \) as two parts: lifecycle fragments that process instances are not adapted; and lifecycle fragments that process instances are adapted. Because \( M \) and \( M' \) are sound, by Formulas (2), (5) and (6), we can have that

\[
\forall s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}} \ L_{M_{r}}.V_{s} \land \neg \hat{f}(s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}}), \exists s_{m, \text{gf}} \circ s_{m, \text{init}, \text{a}}, s_{m, \text{gf}} \circ s_{m, \text{init}, \text{a}} \ L_{M_{r}}.V_{s} \land \tag{10}
\]

\[
\hat{f}(s_{m, \text{gf}} \circ s_{m, \text{init}, \text{a}}) \land L_{M_{r}}.s_{0} \Rightarrow * s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}} \land s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}} \Rightarrow * s_{m, \text{gf}} \circ s_{m, \text{init}, \text{a}}
\]

and

\[
\forall s_{m, \text{init}} \circ s_{m, \text{a}} \ L_{M_{r}}.V_{s} \land \neg \hat{f}(s_{m, \text{init}} \circ s_{m, \text{a}}), \exists s_{m, \text{init}} \circ s_{m, \text{gf}}, s_{m, \text{init}} \circ s_{m, \text{gf}} \ L_{M_{r}}.V_{s} \land \tag{11}
\]

\[
\hat{f}(s_{m, \text{init}} \circ s_{m, \text{gf}}) \land L_{M_{r}}.s_{0} \Rightarrow * s_{m, \text{init}} \circ s_{m, \text{a}} \land s_{m, \text{init}} \circ s_{m, \text{a}} \Rightarrow * s_{m, \text{init}} \circ s_{m, \text{gf}}.
\]

\( s_{m, \text{init}} = L_{M_{r}}.s_{0} \). \( s_{m, \text{gf}} \) and \( s_{m, \text{gf}} \) are final states in \( L_{M'_{r}} \) and \( L_{M_{r}} \), respectively. Formulas (10) and (11) depict the lifecycle fragments that process instances are not adapted. Note that \( s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}} \) and \( s_{m, \text{init}} \circ s_{m, \text{gf}} \) are seen as final states in \( L_{M_{r}} \) for the fact that \( s_{m, \text{gf}} \circ s_{m, \text{init}, \text{a}} \) and \( s_{m, \text{init}} \circ s_{m, \text{gf}} \) are business goals of business process instances that are not adapted. Therefore, both \( \hat{f}(s_{m, \text{a}} \circ s_{m, \text{a}}) \) and \( \hat{f}(s_{m, \text{init}} \circ s_{m, \text{gf}}) \) return true in Formulas (10) and (11). Next, we discuss the lifecycle fragment that business process instances are adapted. Because \( M' \) and \( \Gamma.r_{a} \) are sound, by Formulas (2) and (6), we can have that:

\[
\forall s_{m, \text{gf}} \circ s_{m, \text{c}} \ L_{M_{r}}.V_{s} \land \neg \hat{f}(s_{m, \text{gf}} \circ s_{m, \text{c}}), \exists s_{m, \text{gf}} \circ s_{m, \text{gf}}, s_{m, \text{gf}} \circ s_{m, \text{gf}} \ L_{M_{r}}.V_{s} \land \tag{12}
\]

\[
\hat{f}(s_{m, \text{gf}} \circ s_{m, \text{gf}}) \land s_{m, \text{gf}} \circ s_{m, \text{y}} \Rightarrow * s_{m, \text{gf}} \circ s_{m, \text{c}} \land s_{m, \text{gf}} \circ s_{m, \text{c}} \Rightarrow * s_{m, \text{gf}} \circ s_{m, \text{c}}.
\]

Because \( M \) is sound, by Formulas (2) and (5), we can have that \( L_{M_{r}}.s_{0} \Rightarrow * s_{m, \text{a}} \circ s_{m, \text{a}} \). As \((s_{m, \text{a}} \circ s_{m, \text{init}, \text{a}}, \Gamma.r_{a}, s_{m, \text{gf}} \circ s_{m, \text{y}}) \) and \( s_{m, \text{gf}} \circ s_{m, \text{y}} \Rightarrow * s_{m, \text{gf}} \circ s_{m, \text{c}}, \) we can have that \( L_{M_{r}}.s_{0} \Rightarrow * s_{m, \text{gf}} \circ s_{m, \text{c}} \). Since \( L_{M_{r}}.s_{0} \Rightarrow * s_{m, \text{gf}} \circ s_{m, \text{c}} \) and Formula (12) hold, it is sufficient to show that following formula
\forall s, m, f, g, L, M, \Gamma, V, s \land \neg \exists s, m, f, g, L, M, \Gamma, V, s \land \exists s, m, f, g, L, M, \Gamma, V, s \land

\text{V}_s \land \neg \exists s, m, f, g, L, M, \Gamma, V, s \land \exists s, m, f, g, L, M, \Gamma, V, s \land

\text{holds. By Formulas (10), (11) and (13), it is sufficient to show that Hypothesis Part (II) holds. The proof of Theorem 1 is complete. □}

It is known that the evolution of a business process instance is nondeterministic due to the flexibility and declarative properties of artifact-centric business processes [25]. A business process instance will reach a completely different final state even with the same business data. Furthermore, there exists unexpected business goals of $M'$. Therefore, we must guarantee that a business process instance that is adapted by an adaptation rule can reach an acceptable business goal, i.e., business goal that are acceptable for a customer. We provide the correctness of an adaptation rule to guarantee this constraint.

Before presenting the correctness of an adaptation rule, we provide a definition of an adaptation goal, which is used to define the acceptable business goals. An adaptation goal is defined by domain experts.

**Definition 7.** Given a global adaptation model $\Gamma$, an adaptation goal of business process instances that are adapted by $\Gamma.r_a$ is a set of final states $S_{bg} = \{s_1, \ldots, s_n\}$, where $s_i (1 \leq i \leq n)$ is a final state of the lifecycle of $\Gamma.M'$.

**Definition 8.** Given a global adaptation model $\Gamma$ and an adaptation goal $S_{bg}$ of business process instances that are adapted by $\Gamma.r_a$, $\Gamma.r_a$ is correct if $\text{reach}(\Gamma) \subseteq S_{bg}$.

In Definition 8, $\text{reach}(\Gamma)$ returns a set of reachable final states from state $s_{tp}$ in the lifecycle of $\Gamma.M'$, where $s_{tp} = \text{poststate}(\Gamma.r_a)$. For example, $\text{reach}(\Gamma) = \{(s_{19}, s_{23}, s_{26}, s_{30})\}$, $s_{tp} = \text{poststate}(r_{21}) = (s_{17}, s_{20}, s_{25}, s_{27})$ in Example 2 (Figure 3).

We provide an algorithm $\text{iscorrect()}$ to check the correctness of an adaptation rule. In the algorithm, we suppose $\Gamma.M$, $\Gamma.M'$ and $\Gamma.r_a$ are sound. The algorithm is described as Algorithm 1.

---

**Algorithm 1.** Correctness checking algorithm

**Input:** a global adaptation model $\Gamma$ and an adaptation goal $S_{bg}$.

**Output:** true or false.

0. Begin

1. $s_{tp} \leftarrow \text{poststate}(\Gamma.r_a)$

2. $q.add(s_{tp})$, $S_f \leftarrow \text{null}$, $S \leftarrow S \cup \{s_{tp}\}$

3. While $q$ is not null

4. $s_x \leftarrow q.remove()$

5. $M' \leftarrow \Gamma.M'$

6. For each $(s_a, r, s_b)$ in $L_{M'.E_t}$

7. If $s_a = s_x$ and $s_b$ not in $S$

8. $q.add(s_b)$, $S \leftarrow S \cup \{s_b\}$

9. If $f(s_b)$

10. $S_f \leftarrow S_f \cup \{s_b\}$

11. End

12. End

13. End

14. End

15. If $S_f \subseteq S_f$

16. Return true

17. Return false

18. End
In the algorithm, $q$ is a queue. Lines 3–14 compute the set of reachable final states $S_f$ from state $s_{rp}$ in $L_{M'}$ transitively, i.e.,
\[
\forall s_{m'_sf} \in S_f \ s_{rp} \Rightarrow^* s_{m'_sf} \wedge f(s_{m'_sf}).
\]

Since $\Gamma.M, \Gamma.M'$ and $\Gamma.r_a$ are sound, by Theorem 1, we have $\Gamma$ is sound. From semantics of function $\text{reach}()$, we can see that $S_f = \text{reach}()$. By Definition 8 and Lines 15–17, Algorithm 1 is correct. Let $N = \max(|S|)$ and $K = |L_{M'}, E_{f}|$, the time complexity of Algorithm 1 is $O(NK)$.

In Theorem 2, there is a function $\text{adaptgoal}()$. $\text{adaptgoal}(\Gamma.r_a)$ returns the adaptation goal of business process instances that are adapted by $\Gamma.r_a$.

**Theorem 2.** Given a global adaptation model $\Gamma$, if $\Gamma$ is sound and $\Gamma.r_a$ is correct, a business process instance that is adapted by $\Gamma.r_a$ can reach one business goal $s$ of $L_{M'}$, where $s \in \text{adaptgoal}(\Gamma.r_a)$.

**Proof.** Since $\Gamma$ is sound, we have a set of reachable final states by $\text{reach}(\Gamma)$. By Formulas (6) and (13), we have that state $s_{m'_sf}(s \in \text{reach}(\Gamma))$ can be reached from $s_{m'_sf} \in \text{poststate}(\Gamma.r_a)$ in $L_{M'}$. Reversely, by Formulas (13) and (6), we have that a process instance of $\Gamma.M$ that is adapted to $\Gamma.M'$ by $\Gamma.r_a$ can reach a final state $s'$ of $L_{M'}$, where $s'$ must be in $\text{reach}(\Gamma)$. Since $\Gamma.r_a$ is correct, it is obvious that $\text{reach}(\Gamma) \subseteq \text{adaptgoal}(\Gamma.r_a)$ by Definition 8. Therefore, $s' \in \text{adaptgoal}(\Gamma.r_a)$. The proof of Theorem 2 is complete. \hfill $\Box$

**Example 4.**

In this example, we use aforementioned purchase business process to illustrate our proposed global adaptation model. Suppose the purchase business process is improved by providing options for a customer to pick up an order by him/herself or deliver the order by an express. The old model of the purchase business process is marked as $M_P$ (Figure 1) and the new model of the purchase business process is marked as $M_{P'}$ (Figure 5).

![Figure 5. The purchase business process model $M_{P'}$.](image)

Compare with $M_P$, a new attribute $\text{shipmentType}$ is added in artifact $\text{Shipment'}$; two business rules $r_9$ and $r_{10}$ are added in $M_{P'}$; and two states are added to artifacts $\text{Order'}$ and $\text{Shipment'}$. After the new model is deployed, a business process instance can be adapted from $M_P$ to $M_{P'}$ by our proposed adaptation rule. Here, we only discuss one circumstance. That is, a customer selects to change the type of shipment to self-pick-up and the order is at state $\text{shipment\_created}$. Note that, after the new
model has been deployed, there is a user interface for a customer to change the type of shipment, i.e., self-pick-up or express, if an order has not reached state ready_for_shipping. To implement the adaptation in this circumstance, we define an adaptation rule in Table 3.

Table 3. The adaptation rule $r_a$ that is used for the adaptation.

<table>
<thead>
<tr>
<th>$r_a$</th>
<th>Adapt a purchase process instance of $M_P$ to $M_P'$, where the state of the order is shipment_created and customer changed the type of the shipment to self-pick-up.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_M$</td>
<td>$\text{instate}(\text{Order}, \text{shipment_created}) \land \text{instate}(\text{Payment}, \text{prepaid}) \land \text{instate}(\text{Shipment}, \text{shipment_created}) \land \text{shipmentType} = 1$</td>
</tr>
<tr>
<td>$\lambda_{M'}$</td>
<td>$\text{instate}(\text{Order}', \text{init}) \land \text{instate}(\text{Payment}', \text{init}) \land \text{instate}(\text{Shipment}', \text{init})$</td>
</tr>
<tr>
<td>$v_a$</td>
<td>adapt(Order $o$, Payment $p$, Shipment $s$, Order $o'$, Payment $p'$, Shipment $s'$, Integer shipmentType)</td>
</tr>
<tr>
<td>$\beta_M$</td>
<td>$\text{instate}(\text{Order}, s_{rg}) \land \text{instate}(\text{Payment}, s_{rg}) \land \text{instate}(\text{Shipment}, s_{rg})$</td>
</tr>
<tr>
<td>$\beta_{M'}$</td>
<td>$\text{instate}(\text{Order}', \text{wait_for_customer}) \land \text{instate}(\text{Payment}', \text{prepaid}) \land \text{instate}(\text{Shipment}', \text{wait_for_customer})$</td>
</tr>
</tbody>
</table>

In Table 3, the type of a shipment is self-pick-up if shipmentType equals to 1. According to the definition of the adaptation rule, we can have that prestate($r_a$) = (shipment_created, prepaid, shipment_created) and poststate($r_a$) = (wait_for_customer, prepaid, wait_for_customer). According to $L_{M_P}$ (Figure 4), state (shipment_created, prepaid, shipment_created) is reachable in $L_{M_P}$. Similarly, state (wait_for_customer, prepaid, wait_for_customer) is reachable in $L_{M_P'}$. As such, $r_a$ is sound by Definition 6. Therefore, all purchase process instances that are adapted by the adaptation rule can reach some business goals of $L_{M_P'}$.

In the proposed circumstance, a customer selects to pick up the order by him/herself, such that the order should be closed after the order is picked up and paid by the customer, i.e., the adaptation goal for a business process instance that is adapted by $r_a$ is adaptgoal($r_a$) = \{(closed, paid, customer_picked_up)\}. The state of a purchase process instance is poststate($r_a$) after $r_a$ is evaluated. According to Lines 3–14 of Algorithm 1 and $L_{M_P'}$, we can have that the set of reachable states from state (wait_for_customer, prepaid, wait_for_customer) in $L_{M_P'}$ is $reach(\Gamma) = \{(\text{closed}, \text{paid}, \text{customer_picked_up})\}$, where $\Gamma = (M_P, M_P', r_a, v_a, r_a)$. Because $reach(\Gamma) \subseteq adaptgoal(r_a)$, $r_a$ is correct by Definition 8. By Theorem 2, a purchase process instance of $M_P$ that is adapted by $r_a$ can finally reach an acceptable business goal that belongs to adaptgoal($r_a$). In this example, there is only one business goal (closed, paid, customer_picked_up), which is expected by customers.

5. Tool Prototype

From Example 4, we can see that it is relatively easy for a process designer to define an adaptation rule and an adaptation goal when implementing the adaptation of a model of an artifact-centric business process instance. However, it is a hard work for process designer to judge whether an adaptation rule is sound and correct. To solve this, we developed a tool prototype for helping a process designer to define a sound and correct adaptation rule.

In the tool, we implemented a function to compute the lifecycle of a process model according to lifecycles of artifacts of the process model. We also implemented a function to judge whether the state that corresponds to a condition of an adaptation rule is reachable. Based on these two functions, we implemented Algorithm 1 as one of the main features of the tool. The tool was developed based on Java 1.8, Java Universal Network/Graph Framework (JUNG) 2.1.1 and BeautyEye 3.7 [26,27]. Figure 6 demonstrates the screenshot of verification result of Example 4. Note that names of artifacts of $M_P$ end with “A” and names of artifacts of $M_P'$ end with “B”.

A process designer can import two or more artifact-centric business process models through File menu of the tool, where these models are stored as APMT (Artifact-Centric Process Model based on Label Transition System) document [5]. An APMT document, which is based on XML format, captures the behavior of an artifact-centric business process based on the definitions of lifecycles of artifacts and their synchronization. A process designer can click on the name of an artifact on the left of the main panel to check the lifecycle of the artifact, where the lifecycle is presented in the tabbed panel that is named as “Lifecycle”.

A process designer can define an adaptation rule through Tools menu or click on the title of the tabbed panel that is named as “Rule”. The defined adaptation rule and these imported business process models are used as input of the tool. Verification results for the defined adaptation rule are presented at the bottom of the tabbed panel that is named as “Rule”.

6. Discussions

In this paper, we propose an adaptation rule for adapting a model of an artifact-centric business process instance dynamically and use a global adaptation model to describe the adaptation. We prove that a global adaptation model is sound if the adaptation rule and the two artifact-centric business process models in the global adaptation model are sound. Business process instances that are adapted by the adaptation rule can reach some acceptable business goals if the adaptation rule is correct and the global adaptation model is sound. During the adaptation, both changes of data (artifact information model) and process (lifecycles of artifacts) are considered.

Although we discuss the adaptation of a model of an artifact-centric business process instance theoretically, our theoretical results can be used directly to support the adaptation of business process instances in the execution level if business processes are running on the framework proposed in [25]. Because they claim that the framework supports execution of artifact-centric business process model (ACP model) directly without model transformation. Furthermore, our proposed global adaptation model can be seen as an artifact-centric business process model. Comparison of the framework with other works can be found in [25].
In this paper, our proposed adaptation rule is used to adapt a model of a business process instance. From another perspective, our adaptation rule can also be used to combine lifecycle fragments of different business process models to construct complex business processes, because a global adaptation model can be seen as an equivalent artifact-centric business process model according to its definition. From this perspective, artifact-centric business process model is improved with the ability of modeling sub business processes, which, to our best knowledge, has not been addressed in the artifact-centric business process community.

However, in a global adaptation model, there is only one adaptation rule. In real-world, multiple adaptation rules should be defined to adapt business process instances of different circumstances. Moreover, transitive adaptation, i.e., a process instance is adapted more than one time, is not discussed. Nevertheless, the same results can also be derived when multiple adaptation rules are defined in one global adaptation model. Because a global adaptation model with one adaptation rule can be seen as an ordinary artifact-centric business process model, such that the soundness of the global adaptation model can be proved recursively. Furthermore, the adaptation goal of business process instances that are adapted by an adaptation rule is restricted locally to the adaptation rule, such that the correctness of an adaptation rule is independent of other adaptation rules.

There are also some limitations of our proposed method. First, an adaptation rule and the adaptation goal of business process instances that are adapted by the adaptation rule are defined by human. Second, our proposed method is closed related to ACP model [25]. There are some other similar artifact-centric business process modeling approaches, such as GSM and CMMN [3,9,10], where a milestone (state in ACP model) is reached if a guard (a condition for opening a stage) of a stage (action in ACP model) is satisfied and the stage is executed and condition of the sentry that corresponds to the milestone is satisfied. Comparing ACP model with GSM and CMMN models, they all depict artifact behaviors by lifecycles of artifacts. Differently, a lifecycle of an artifact in ACP model is driven by Condition–Action business rules, and a lifecycle of an artifact in GSM and CMMN models is driven by variants of Event–Condition–Action rules. We try to adapt our proposed method to apply to GSM and CMMN models.

7. Conclusions

In this paper, we propose an adaptation rule for adapting a model of an artifact-centric business process instance and describe the adaptation by a global adaptation model. We prove that the business process instance can reach an acceptable business goal without deadlock if the adaptation rule is sound and correct. We developed a tool prototype for helping a process designer to define a sound and correct adaptation rule. Possible applications and limitations of our proposed method are discussed in the Discussion Section. Our future work will try to adapt our proposed approach to other models, like GSM and CMMN.

Author Contributions: Conceptualization, J.Z.; Methodology, J.Z.; Software, J.Z.; Writing-Original Draft Preparation, J.Z.; Writing-Review & Editing, J.Z. and G.L.; Supervision, G.L.

Funding: This research received no external funding.

Acknowledgments: The authors gratefully acknowledge the editors and the anonymous reviewers for their detailed and constructive criticisms, which have helped us to improve the quality and presentation of our manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References


