A Framework for Behavior-Consistent Specialization of Artifact-Centric Business Processes

Sira Yongchareon[1], Chengfei Liu[1], and Xiaohui Zhao[2]

Faculty of Information and Communication Technologies
Swinburne University of Technology, Victoria, Australia
{syongchareon, cliu}@swin.edu.au

[2]Department of Computing, Faculty of Creative Industries and Business
Unitec Institute of Technology, Auckland, New Zealand
xzhao@unitec.ac.nz

Abstract. Driven by complex and dynamic business process requirements, there has been an increasing demand for business process reuse to improve modeling efficiency. Process specialization is an effective reuse method that can be used to customize and extend base process models to specialized models. In the recent years, artifact-centric business process modeling has emerged as it supports a more flexible process structure compared with traditional activity-centric process models. Although, process specialization has been studied for the traditional models by treating a process as a single object, the specialization of artifact-centric processes that consist of multiple interacting artifacts has not been studied. Inheriting interactions among artifacts for specialized processes and ensuring the consistency of the processes are challenging. To address these issues, we propose a novel framework for process specialization comprising artifact-centric process models, methods to define a specialized process model based on an existing process model, and the behavior consistency between the specialized model and its base model.

1. Introduction

Complex business process requirements from different customer needs, government regulations, outsourcing partners, etc., result in frequent changes and revision to business processes. Therefore, reusability of business processes is highly sought after to improve process modeling efficiency. In this background, organizations strive for a more efficient and systematic approach to flexibly define and extend their business processes. Business process reuse aims to support on-demand customization and extension of existing business processes by establishing a modular and a repository of process components [18]. Business process specialization is deemed as one of main mechanisms that can be used to construct a specific business process by extending a generic reference process model. With specializations, processes can be reported at different levels of generality, and can be compared across the specializations [19].

Current activity-centric modeling approaches focus on the conformation of tasks and the control-flows among tasks according to specific logics. Intuitively, constructing processes with sequenced activities leads to highly-cohesive and tightly-coupled process structures; therefore, process componentization and extension are difficult to be achieved in a natural way [17]. In recent years, *artifact-centric*

approaches to business process modeling have emerged and been widely studied [1, 2, 3, 7, 11, 13, 16, 17, 19]. These approaches naturally lend themselves well to both object-orientation and service-orientation design principles, as they focus on the design of both business artifacts involved in a process and services performing operations on such artifacts. Owning to the object-oriented nature, the artifact-centric models support higher level of flexibility, extensibility, and reusability.

The existing approaches for the specialization of business processes treat a process as a single object [8, 9, 10]; hence, traditional object specialization techniques in object-oriented analysis and design can be applied (e.g., from [4]). For artifact-centric processes, specializations should not only apply on each individual artifact but also on their interactions. Some works have initiated the study of object lifecycles and their interactions within (or between) business processes in various areas, e.g., process adaptation and dynamic changes [20], design compliance [6, 13], conformance checking [16], and contract for inter-org processes [12]. However, a specialization mechanism that takes into account the interactions of objects and the guarantee of behavior consistency between a specialized process and its base process brings in technical challenges and requires further study. To address these challenges, we propose a novel framework for behavior-consistent process specialization that consists of artifact-centric process model, methods to define a specialized process model based on an existing model, and the behavior consistency between the specialized model and its base model.

The remainder of this paper is organized as follows. Section 2 presents an artifact-centric process model and an approach to process specialization. Section 3 discusses the behavioral properties and the consistency between a specialized model and its base model. Section 4 reviews and discusses related works. Finally, the conclusion and future work are given in Section 5.

2. Specialization of Artifact-Centric Business Process Model

To begin with, we briefly introduce an *artifact-centric business process model* (*ACP model*) (e.g., in [2, 3]). The ACP model constitutes of three sets: *artifact classes*, *services*, and *business rules*. An *artifact class* (or *artifact* if the context is clear), containing its relevant attributes and states, is a key business entity involved in business processes. A *service* is a task that performs read/write operations on some artifact(s). A *business rule* is defined in a *Condition-Action* style to associate service(s) with artifact(s). It describes on what *pre-condition* a particular service is invoked, and what *post-condition* that the effect (after performing such service) must satisfy. We can say that a (complete) set of business rules defined in a process model specifies the control logic of the whole process from its beginning to its end. Now, we use two simplified product ordering processes to illustrate and motivate the artifact-centric process specialization, as shown in Fig. 1.

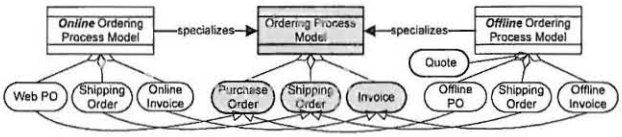

Fig. 1. Generic ordering process with its two specializations

The example depicts a generic ordering process model with its two possible specialized process models: *Online* and *Offline* ordering processes. The former offers a service to only retail customers on the web, while the latter accepts both retail and wholesale customers. The *Online* ordering process has each artifact specializes its base artifact in the ordering process, e.g., *Web PO* specializes *Purchase Order*. Not only the internal behavior of *Web PO* is specialized, but it is possible that some synchronization between *Web PO* and the other artifact(s) may also need to be modified due to the specialization. Now, consider the *Quote* artifact that is added into the *Offline* ordering process. Extending this artifact, of course, requires some synchronization with the other artifact(s), e.g., *Offline PO*. Next, Section 2.1 explains more details about how to define an ACP model and Section 2.2 introduces our approach to define a specialized ACP model based on an existing ACP model.

2.1 Syntax of ACP model

First, we begin with the definitions of an artifact. An *artifact schema* $Z = \{C_1, C_2 ..., C_x\}$ is a finite set of *artifact classes*. Each *artifact* $C_i \in Z$ $(1 \leq i \leq x)$ can be defined as a tuple (A, s^{init}, S, S^f) where set $A = \{a_1, a_2, \ldots, a_y\}$, and each $a_j \in A (1 \leq j \leq y)$ is a name-value pair attribute; set $S = \{s_1, s_2, \ldots, s_z\}$ contains the possible states of the instances of class C_i; s^{init} is the *initial* state, and $S^f \subseteq S$ is a set of *final* states. For example in Fig. 1, the *Purchase Order* (*PO*) artifact can be defined as ({*OrderID*, *SupplierID*, *GrandTotal*, *SubmitDate*, *CompleteDate*}, *init*, {*created*, *confirmed*, *canceled*, *ready to ship*, *dispatched*, *billed*, *closed*}, {*closed*, *canceled*}), and the *Shipping Order* (*SO*) artifact can be defined as ({*ShippingID*, *OrderID*, *SubmitDate*, *ShipDate*, *CompleteDate*}, *init*, {*scheduled*, *in transit*, *arrived*, *completed*}, {*completed*}). Next, we define a *business rule* to capture the processing control logic (in a *Precondition-Action-Postcondition* style). A *business rule*, denoted as r, is a tuple (λ, β, v) where λ and β are a *pre-condition* and *post-condition*, respectively; v is a service that performs read/update operations on the attributes and the processing states of some artifacts in schema Z. In this paper, we restrict both pre- and post-conditions to be expressed by a conjunctive normal form (CNF). This form contains two types of proposition over schema Z: (1) *state proposition* (the *instate* predicate) and (2) *attribute proposition* (the *defined* and scalar comparison operators). We write *defined*(C, a) if attribute $a \in C.A$ of artifact of class C has a value; and *instate*(C, s) if state $s \in C.S$ of artifact of class C is active. Table 1 shows an example (incomplete) set of business rules that are used in our generic ordering process.

Table 1. Example of business rules

r_1 : Buyer confirms *Purchase Order* po to the selected Supplier	
Pre-condition	*instate*(po, *sent_to_supplier*) ∧ *defined*(po, *OrderID*) ∧ *defined*(po.*SupplierID*)
Task	confirmPO(po)
Post-condition	*instate*(po, *confirmed*) ∧ *defined*(po.*SubmitDate*)
r_2 : Supplier creates *Shipping Order so* for *Purchase Order* so	
Pre-condition	*instate*(po, *confirmed*) ∧ *defined*(po.*SupplierID*) ∧ *instate*(so, *init*)
Task	createSO(po, so)
Post-condition	*instate*(po, *ready_to_ship*) ∧ *instate*(so, *scheduled*) ∧ *defined*(so.*ShippingID*) ∧ *defined*(so.*OrderID*)

Definition 1: (Artifact-Centric Process Model or ACP model). An *ACP model*, denoted as Π, is a tuple (Z, V, R), where Z is an artifact schema, V is a set of services, and R is a set of business rules over Z.

We also define two auxiliary functions over business rules R and artifact schema Z of Π in which they are used in later sections. Function $pre\_s(r, C)$ returns a set of states $\{s_1, s_2, \ldots, s_x\}$ where business rule $r \in Z$ and state $s_i \in C.S(1 \leq i \leq x)$ is defined in the $instate$ predicate of the pre-condition of r. Correspondingly, function $pre\_s(r, C)$ returns a set of states of artifact C appearing in the post-condition of r.

2.2 Approach to artifact-centric business process specialization

Intuitively, we can adopt a single object specialization in the traditional object-oriented (OO) approaches (e.g., [4, 8-10]) for an individual artifact class in our model. Apart from the specialization of artifacts in a process, we investigate the specialization of their interactions. In the design and modeling phase, we propose that the specialization of ACP models can be achieved by three construction methods: *artifact refinement*, *artifact extension*, and *artifact reduction*.

- *Artifact refinement.* Process modelers decide to inherit an artifact from a base model by refining (adding/modifying) some corresponding business rules and states to the specialized model. The pre-condition and post-condition of a modified rule may have a state of the supertype refined into new state(s) in the subtype. Note that the refinement can be performed on a single business rule that is used to synchronize two or more artifacts.
- *Artifact extension.* Process modelers decide whether there is a need of any additional artifact for the specialized. Adding new artifacts to a process implies that the process requires not only new business rules (of such artifact) but also synchronization rules (called *sync rules*) between the new artifact and existing artifact(s).
- *Artifact reduction.* Process modelers delete an entire artifact in the specialized process. The removal of an artifact should have a propagated effect on the behavior of other artifact(s) that it synchronizes with.

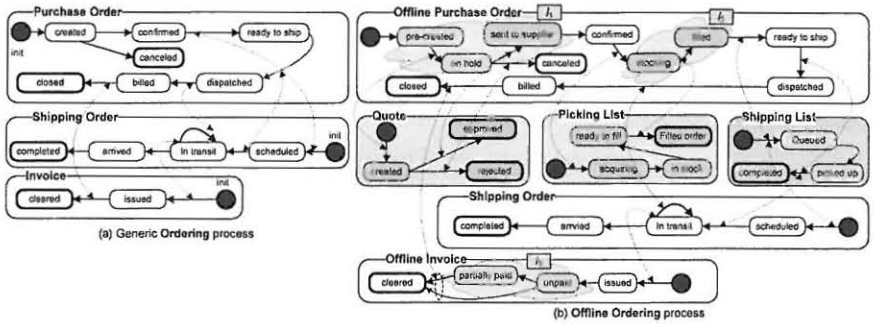

Fig. 2. Example of ACP specialization

Fig. 2 shows an example of specialized *Offline* ordering process and its base ordering process. The dash line linked between transitions of different artifacts indicates the synchronization (by means of using *sync rules*) between such artifacts.

The shaded artifacts represent *extended* artifacts. Similarly, for an existing artifact, a set of gray-shaded states and their corresponding transitions represents the refinement of its base artifact. In Fig. 2 (b), *Offline PO* is specialized by applying *artifact extension* (*Quote*, *Picking List*, and *Shipping List* are added with additional sync rules) and *artifact refinement* (the *created* state and the transition from the *confirmed* state to *ready to ship* state are refined with more details). Similarly, *Offline invoice* is specialized by applying *artifact refinement* on the transition from the *issued* state to the *cleared* state. Next, we define a process specialization function that maps a specialized ACP model to its supertype, called *ACP specialization.*

Definition 2: (ACP specialization). Given two ACP models $\Pi = (Z, V, R)$ and $\Pi' = (Z', V', R')$, we define a specialization relation between Π and Π' by *ACP specialization function* $ps_{\Pi' \to \Pi} : Z' \cup V' \cup R' \to Z \cup V \cup R \cup \{\varepsilon\}$ such that *ps* is a total function mapping from each element in specialized model Π' onto the element in Π or *empty element* ε. Note that we slightly abuse the use of *ps* to map from $C'.S$ onto $C.S$ where $C' \in Z'$ and $C \in Z$.

3. Behavior-Consistent Process Specialization

3.1 Behavioral properties of ACP

We first classify behavioral properties of ACP models into *intra-behavior* and *inter-behavior*. The *intra-behavior* of an artifact describes how an artifact changes its state throughout its lifecycle. Here, we use a deterministic finite state machine to capture the *lifecycle* of an individual artifact. The *inter-behavior* describes how a lifecycle of one artifact depends on the counterpart of another artifact, and it can be represented as state dependency (via a *sync rule*) between artifacts. Then, we discuss about the *soundness* property of individual artifact and the entire process which is constructed by composing all of its artifact lifecycles.

Definition 3: (Lifecycle). Let artifact class $C_i = (A_i, {s_i}^{init}, S_i, {S_i}^f)$ be in ACP model Π. The *lifecycle* of C_i, denoted as $\mathcal{L}_{C_i}$, can be defined as a tuple $(S, s^{init}, \Rightarrow)$ where set $S = S_i \cup {S_i}^f$, $s^{init} = {s_i}^{init}$, and transition relation $\Rightarrow \subseteq S \times R_i \times G_i \times S$ where $R_i \subseteq \Pi.R$ is a set business rules that are used to induce a state transition of artifact C_i, and G_i (guards) is a union set of state preconditions of each business rule in R_i such that each precondition references to a state of other artifact in Π.

Definition 4: (Sync rule). Given ACP model Π, a set of *sync rules* between lifecycles of artifacts $C_x \in \Pi.Z$ and $C_y \in \Pi.Z$ is denoted as $\varphi(\mathcal{L}_{C_x}, \mathcal{L}_{C_y}) = \{r \in \Pi.R \mid \exists(s_i, r, g, s_j) \in \mathcal{L}_{C_x}.\Rightarrow \wedge\ \exists(s_m, r, g, s_n) \in \mathcal{L}_{C_y}.\Rightarrow\}$.

Next, we define *ACP lifecycle* for describing the behavioral aspect of an ACP model consisting of synchronized lifecycles of artifacts. We adapt a state machine composition technique presented in [5] for generating the lifecycle of ACP.

Definition 5: (Lifecycle composition and composed lifecycle). Given ACP model Π, two lifecycles $\mathcal{L}_i = (S_i, s_i^{init}, \Rightarrow_i)$, and $\mathcal{L}_j = (S_j, s_j^{init}, \Rightarrow_j)$ can be composed, denoted as $\mathcal{L}_i \oplus \mathcal{L}_j$, into *composed lifecycle* $\mathcal{L}_c = (S_c, s_c^{init}, \Rightarrow_c)$, where $S_c \subseteq \mathcal{L}_i.S \times \mathcal{L}_j.S$ is a set of composed states, $s_c^{init} = (\mathcal{L}_i.s^{init}, \mathcal{L}_j.s^{init})$ is the initial state, and $\Rightarrow_c \subseteq S_c \times \Pi.R \times G_c \times S_c$ is transition relation where G_c is a set of guards.

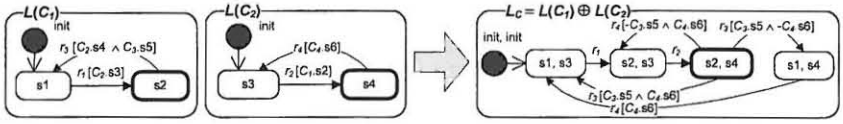

Fig. 3: An example of a lifecycle composition

Fig. 3 shows the composition between the lifecycle of artifact C_1 and the lifecycle of artifact C_2. We attach label $r_i[g]$ to a transition to mean that the transition is fired when both the attribute proposition in the pre-condition of business rule r_i holds and all state propositions (of external lifecycles) in g hold. We denote the counter state condition of $C.s_x$ by symbol $-C.s_x$ in the guard. Next, we define the lifecycle of ACP by using *lifecycle composition*. Given ACP model Π, an *ACP lifecycle* of Π, denoted as $\mathcal{L}_\Pi$, can be generated by iteratively performing *lifecycle composition* of every artifact in Π. For both artifact lifecycle and ACP lifecycle, we define *lifecycle occurrence* to refer to a particular sequence of states occurring from the *init* state to one of the *final* states of the lifecycle. Based on this, we define a *soundness* property to describe the desired and correct behavior of an artifact lifecycle and a process. Given ACP model Π, and *lifecycle* $\mathcal{L}$ (of either an artifact class or Π), a *lifecycle occurrence* is denoted as $\sigma = (s^{init}, ..., s_f)$ such that for every state $s \in \sigma$, there exists final state $s_f \in \mathcal{L}.S$ and s_f can be reached from s^{init} through s by a particular firing sequence of some business rules in R.

Definition 6: (Safe, Goal-reachable, and Sound lifecycle). Given ACP model Π, lifecycle $\mathcal{L} = (S, s^{init}, \Rightarrow)$ (of either an artifact class in $\Pi.Z$ or Π), we define sets of lifecycle states $S = \mathcal{L}.S \cup \{\mathcal{L}.s^{init}\}$ and final states $S^f \subseteq S$. Lifecycle $\mathcal{L}$ is said to be:

- *safe* iff there exists business rule $r \in \Pi.R$ such that r induce one and only one transition in $\mathcal{L}$;
- *goal-reachable* iff, for every non-final state s, there exists s in some lifecycle occurrence; and, for every final state $s_f \in S^f$, there exists occurrence σ such that s_f is the last state of σ;
- *sound* iff $\mathcal{L}$ is *safe* and *goal-reachable*

In the rest of paper, we restrict our discussion only to the *sound* behavior of artifacts and ACP based on their lifecycles (not the changes of artifact's data). However, discussions and formal approaches to data verification can be found in [2].

3.2 Behavior consistent specialization

In this section, we discuss the behavior consistency between a specialized ACP model and its base model when applying three different methods of ACP specialization introduced in Section 2.2: *refinement*, *extension*, and *reduction* of artifact. In object-oriented design approaches, the consistency of (dynamic) object behaviors between subtype and its supertype can be divided into observation consistency and invocation consistency. Observation consistency ensures that if features added at a subtype are ignored and features refined at a subtype are considered unrefined, any processing of an artifact of the subtype can be observed as correct processing from the view of the supertype. The invocation consistency refers to the idea that instances of a subtype can be used in the same way as instances of the supertype. More detailed discussion

about object's behavior consistencies can be found in [4, 10]. In this article, we restrict our discussion of business process specialization to observation consistency (for both artifact and process). On one hand, in the viewpoint of structure, it is ensured that the current processing states of artifact and process are always visible at the higher (abstracted) organizational role. On the other hand, to preserve the behavior consistency it is guaranteed that business rules added at a subtype do not interfere with the business rules inherited from its supertype. Particularly, dealing with changes of synchronization dependencies between artifacts is a major technical issue of *ACP specialization*. Here, we consider *ACP specialization* for an entire process as the product of (1) the specialization of individual lifecycle (*lifecycle specialization*) and (2) the specialization of synchronizations (*sync specialization*).

Definition 7: (Lifecycle specialization). Let ACP model Π' be a specialization of ACP model Π with *ACP specialization* $ps_{\Pi' \to \Pi}$. Given lifecycle $\mathcal{L} = (S, s^{init}, \Rightarrow)$ (of artifact in Π or Π) and lifecycle $\mathcal{L}' = (S', s^{init'}, \Rightarrow')$ (of artifact in Π' or Π'), we define lifecycle specialization relation between $\mathcal{L}$ and $\mathcal{L}'$ based on $ps_{\Pi' \to \Pi}$ by *lifecycle specialization (total) function* $ls_{\mathcal{L}' \to \mathcal{L}} : S' \cup s^{init'} \cup \Rightarrow' \to S \cup s^{init} \cup \Rightarrow \cup \{\varepsilon\}$.

For ACP specialization method by the *refinement* of artifact class, a single state (or a transition) is refined into a set of sub states and sub transitions. Here, we define a fragment of lifecycle, called *L-fragment*, which contains a set of sub states and sub transitions for capturing the refinement, e.g., in Fig. 2, L-fragment l_3 refines a transition of *Invoice* and L-fragment l_1 refines the *created* state of *Purchase Order*. Then, we use *lifecycle specialization function ls* to project every state and transition of a fragment in a specialized lifecycle onto a state or a transition of its base lifecycle.

Definition 8: (L-fragment). *L-fragment* of lifecycle $\mathcal{L}_x = (S, s^{init}, \Rightarrow)$ is a nonempty connected sub-lifecycle of $\mathcal{L}_x$, defined as $\ell^{\mathcal{L}_x} = (S, \Rightarrow, \Rightarrow^{in}, \Rightarrow^{out})$ where,

- $S \subseteq \mathcal{L}_x.S \setminus \{s^{init}\}$ and $\Rightarrow \subseteq S \times \Pi.R \times G \times S \subseteq \mathcal{L}_x.\Rightarrow$,
- $\Rightarrow^{in} = \mathcal{L}_x.\Rightarrow \cap ((\mathcal{L}_x.S \setminus S) \times \mathcal{L}_x.\Rightarrow \times S))$ and $\Rightarrow^{out} = \mathcal{L}_x.\Rightarrow \cap (S \times \mathcal{L}_x.\Rightarrow \times (\mathcal{L}_x.S \setminus S)$ are sets of entry transitions and exit transitions of $\ell^{\mathcal{L}_x}$, respectively,
- such that, for every state s in $\ell^{\mathcal{L}_x}.S$, there exists a sequence of transitions from some entry transition in $\Rightarrow^{in}$ to s and from s to some exit transition in $\Rightarrow^{out}$.

Now, we apply *L-fragment* to the *lifecycle specialization*. Let lifecycle $\mathcal{L}' = (S', s^{init'}, \Rightarrow')$ be a specialization of lifecycle $\mathcal{L} = (S, s^{init}, \Rightarrow)$ by *lifecycle specialization* $ls_{\mathcal{L}' \to \mathcal{L}}$. We denote a set of *refined L-fragments* that are used to refine $\mathcal{L}$ as $lf(\mathcal{L}' \to \mathcal{L}) = \{\ell_1, \ell_2, \ldots, \ell_y\}$ where $\ell_i (1 \le i \le y)$ is an L-fragment in $\mathcal{L}'$ such that ℓ_i does not exist in $\mathcal{L}$.

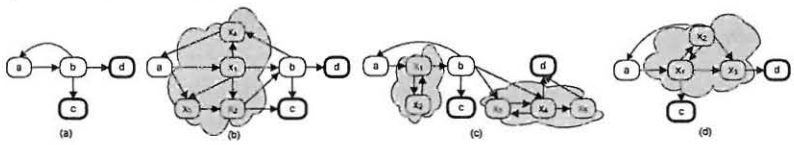

Fig. 4. Examples of L-fragments

For example in Fig. 4, lifecycles (b), (c), and (d) are different specializations of lifecycle (a). Lifecycles (b) and (c) refine some transitions of lifecycle (a), while lifecycle (d) refines only state *b* of lifecycle (a). Next, we want to check whether the

behavior of specialized lifecycle $\mathcal{L}_y$is consistent to the behavior of its base lifecycle $\mathcal{L}_x$. It is understandable that if every lifecycle occurrence of $\mathcal{L}_y$, disregarding the states and transitions added by applying L-fragment, is observable as the same sequence as of $\mathcal{L}_x$, then $\mathcal{L}_y$ is *behavior-consistent* to $\mathcal{L}_x$.

Definition 9: (Behavior-consistent or *B-consistent*). Let lifecycle $\mathcal{L}_y = (S_y, {s_y}^{init}, \Rightarrow_y)$ and lifecycle $\mathcal{L}_x = (S_x, {s_x}^{init}, \Rightarrow_x)$ such that $\mathcal{L}_y$ specializes $\mathcal{L}_x$ with *lifecycle specialization* $ls_{\mathcal{L}_y \to \mathcal{L}_x}$, and $S_{x \cap y} = S_x \cap S_y$ be a set of states that exist in both $\mathcal{L}_x$ and $\mathcal{L}_y$. We have $\mathcal{L}_y$ *B-consistent* to $\mathcal{L}_x$ iff $\forall s_i, s_j \in S_{x \cap y}, \exists (s_i, r, g, s_j) \in \Rightarrow_x, \forall s_k \in S_y \backslash S_{x \cap y}, s_i \overset{*}{\Rightarrow}_y s_k \wedge s_k \overset{*}{\Rightarrow}_y s_j$ where $\overset{*}{\Rightarrow}$ is denoted for a reflexive transitive closure of $\Rightarrow$. We also say that $ls_{\mathcal{L}_y \to \mathcal{L}_x}$ is *B-consistent*.

For instance, the lifecycle in Fig. 4 (b) is not *B-consistent* to the lifecycle in Fig. 4 (a). This is because, in some lifecycle occurrences of lifecycle (b), state a can reach state c (through state x_2) without passing state b; and, state a can reach itself via state x_4 without passing state b. In contrast, we can see that $ls_{\mathcal{L}_c \to \mathcal{L}_a}$ in Fig. 4 (c) and $ls_{\mathcal{L}_d \to \mathcal{L}_a}$ in Fig. 4 (d) are *B-consistent*. Note that we can also apply *B-consistent* for the case of L-fragments. Now, we define L-fragment with a single entry and a single exit state as *atomic L-fragment* (*AL-fragment*) and show how it is considered for the behavioral consistency between two lifecycles.

Definition 10: (AL-fragment). Given ACP model Π and *L-fragment* $\ell = (S, \Rightarrow, \Rightarrow^{in}, \Rightarrow^{out})$ of lifecycle $\mathcal{L}_x = (S, s^{init}, \Rightarrow)$, ℓ_i is called *AL-fragment* iff for every entry transition $\Rightarrow_m \in \ell.\Rightarrow^{in}$, $\Rightarrow_m$ is fired from same source state $s_x \in \mathcal{L}.S \backslash \ell.S$; and, for every exit transition $\Rightarrow_n \in \ell.\Rightarrow^{out}$, $\Rightarrow_n$ is fired to same target state $s_y \in \mathcal{L}.S \backslash \ell.S$.

Theorem 1: Let lifecycle $\mathcal{L}'$ be a specialization of lifecycle $\mathcal{L}$ with a set of *refined L-fragments* $lf(\mathcal{L}' \to \mathcal{L})$, $ls_{\mathcal{L}' \to \mathcal{L}}$ is *B-consistent* if, for every $\ell_i \in lf(\mathcal{L}' \to \mathcal{L})$,

- if ℓ_i refines transition $s_x \Rightarrow_t s_y \in \mathcal{L}.\Rightarrow$ then ℓ_i is an *AL-fragment*; or,
- if ℓ_i refines state $s \in \mathcal{L}.S$ then, for every instate $s_x \in \mathcal{L}.S$ fired to s and for outstate $s_y \in \mathcal{L}.S$ fired from s, s_x can reach s_y in some *L-occurrences* of ℓ_i.

Revisiting Fig. 2, *Offline PO* (with L-fragments ℓ_1 and ℓ_2) and *Offline Invoice* (with L-fragment ℓ_3) are *B-consistent* to *Purchase Order* and *Invoice*, respectively. Next, we define *B-consistent specialization* for both an artifact and a process.

Definition 11: (B-consistent specialization). Given ACP model Π' be a specialization of ACP model Π with ACP specialization $ps_{\Pi' \to \Pi}$, Π' is a *B-consistent specialization* of Π iff $ls_{\mathcal{L}'_\Pi \to \mathcal{L}_\Pi}$ is *B-consistent*. Given artifact $C' \in \Pi'.Z$ be a specialization of artifact $C \in \Pi.Z$ based on $ps_{\Pi' \to \Pi}$, C' is a *B-consistent specialization* of C iff $ls_{\mathcal{L}'_C \to \mathcal{L}_C}$ is *B-consistent*.

3.3 Specialization of synchronization dependencies

This section discusses how changes of artifact interactions (through their synchronization dependencies) affect the behavior of the process in their specialization at both the artifact level and the process level. We classify specialization of synchronizations into three methods: *synchronization* (*sync*) *extension*, *refinement*, and *reduction*. First, *sync extension* is a method of

synchronizing new artifact with an existing artifact without refining any existing sync rule. However, it is achieved by adding a new defined set of sync rules, called *extended sync rules*. Second, *sync refinement* is a method to decompose an individual existing sync rule in the base process to a new set of refined sync rules in the specialized process. A specialized sync rule can be used to synchronize between existing artifacts or between existing artifact(s) and new (extended) artifact(s) added to the specialized process. Last, *sync reduction* is a method of removing synchronization between existing artifacts. Fig. 5 shows an abstracted example of results after applying different sync specialization methods to the base process (a). More discussions on this example will appear through the rest of the paper.

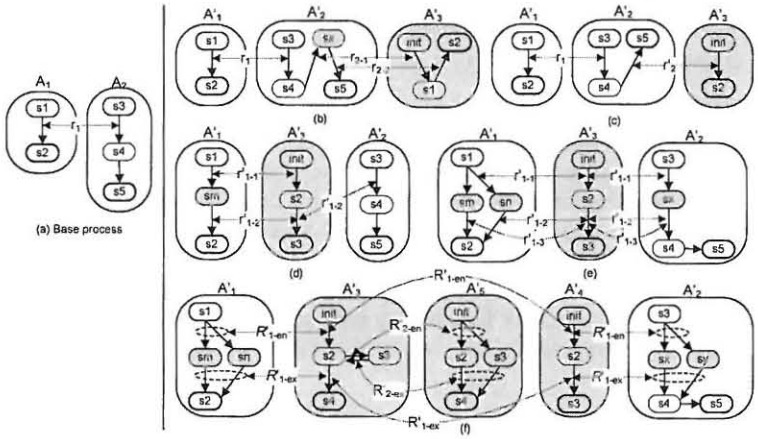

Fig. 5. Examples of sync specializations between artifacts

The consistency of synchronization dependencies in process specialization means whatever changes made to the synchronization of artifacts the behavior of the *composed lifecycle* of such specialized artifacts must be consistent to their composition in the base process. Particularly, adding a new artifact into a specialized process results unobservable behavior of itself in the base process. Similarly, removing existing artifact from the base process in the specialized process can also have an impact on the behavior consistency. However, it is desirable that the overall behaviors of such base and specialized processes (with added or removed artifact) remain consistently observable. For instance, based on our ordering process in Fig. 2, the *Quote* artifact added to the *Offline* ordering process should not interfere with the behavior of the *Purchase Order, Shipping Order* and *Invoice* artifacts and their interactions in its base ordering process. On the other hand, the removed artifact should also not impact the overall behavior consistency of the base process. Next, we define the specialization of the synchronization between two lifecycles followed by detailed discussion on how synchronization is consistently handled when applying each of the three sync operations.

Definition 12: (Sync specialization). Let artifact lifecycles $\mathcal{L}'_{C_x}$ and $\mathcal{L}'_{C_y}$ in specialized ACP model Π' be a *B-consistent specialization* of artifact lifecycles $\mathcal{L}_{C_x}$ and $\mathcal{L}_{C_y}$ in base ACP model Π, respectively. We define *sync specialization*

$ss^{\Pi' \to \Pi}{}_{(\mathcal{L'}_{C_x}, \mathcal{L'}_{C_y}) \to (\mathcal{L}_{C_x}, \mathcal{L}_{C_y})} : \varphi(\mathcal{L'}_{C_x}, \mathcal{L'}_{C_y}) \to \varphi(\mathcal{L}_{C_x}, \mathcal{L}_{C_y}) \cup \{\varepsilon\}$ as a total function that projects a specialized sync rule between $\mathcal{L'}_{C_x}$ and $\mathcal{L'}_{C_y}$ onto its base sync rule between $\mathcal{L}_{C_x}$ and $\mathcal{L}_{C_y}$ or *empty element* ε.

Now, in order to capture and analyze synchronizations between two lifecycles we extend the definition of AL-fragment of isolated lifecycle to *atomic synchronized L-fragment*, called *ASL-fragment*, between two lifecycles.

Definition 13: (*ASL-fragment*). Given ACP model Π, let L-fragment ℓ^{C_x} of artifact $C_x \in \Pi.Z$ synchronize with L-fragment ℓ^{C_y} of artifact $C_y \in \Pi.Z$ via business rules $R^{sync} \in \Pi.R$. ℓ^{C_x} and ℓ^{C_y} are *ASL-fragments* iff both ℓ^{C_x} and ℓ^{C_y}satisfy the property of L-fragment and the following conditions:

- $\forall r \in R^{sync}, \forall s \in pre\_s(r, C_x) \cup post\_s(r, C_x), s \in \ell^{C_x}.S$; and,
- $\forall r \in R^{sync}, \forall s \in pre\_s(r, C_y) \cup post\_s(r, C_y), s \in \ell^{C_y}.S$.

Based on the two conditions of ASL-fragment's definition, two ASL-fragments are restricted to include all possible reachable composite states when they are composed into a composite fragment (by applying *lifecycle composition*). It can be understood that an excluded reachable state (which is supposed to be included in the composite fragment) violates the atomicity of the composition. This implies unobservable (and incomplete) behavior of the composite fragment, i.e., a transition in the composite fragment may lead to any external state and that transition (and its state) cannot be observed within the view of the composite fragment.

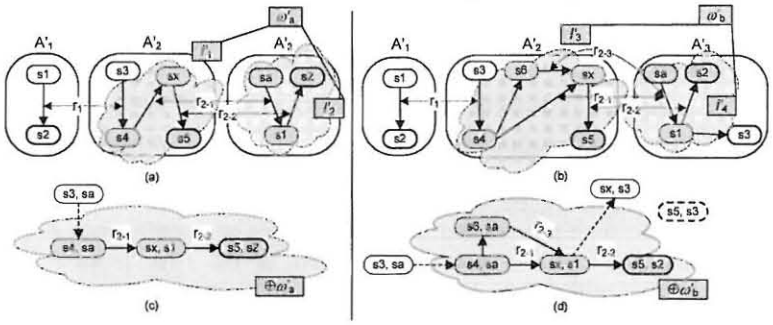

Fig. 6. Examples of the composition of synchronized L-fragments

For example, Fig. 6 (d) shows the composite fragment of two synchronized L-fragments ℓ'_3 and ℓ'_4 in Fig. 6 (b). We can see that composite (and reachable) state (s_x, s_3) is not included in the composite fragment, while state s_x appears in ℓ'_3. Furthermore, composite transition $(s_x, s_1) \Rightarrow (s_x, s_3)$ in the composite fragment leads to state (s_x, s_3) which is excluded from the composite fragment. As such, we cannot observe the complete behavior of both ℓ'_3 and ℓ'_4; therefore, ℓ'_3 and ℓ'_4 cannot be *ASL-fragments*. On the contrary, the composite fragment in Fig. 6 (c) between L-fragments ℓ'_1 and ℓ'_2 in Fig. 6 (a) has no unobservable transition; thus, ℓ'_1 and ℓ'_2 are *ASL-fragments*. Based on the above discussion, we have that the composition of two synchronized L-fragments preserves the *B-consistency* if such two fragments are

ASL-fragments, as shown in Lemma 1. This concept will be mainly used to check the *B-consistency* when applying different sync specializations.

Lemma 1: Let two ASL-fragments ℓ^{C_x} and ℓ^{C_y} be a specialization of ℓ^{C_x} with specialization function $ls_{(\ell^{C_x} \oplus \ell^{C_y}) \rightarrow \ell^{C_x}}$, ℓ^{C_x} is *B-consistent* to the composition of ℓ^{C_x} and ℓ^{C_y}. Analogously, the *B-consistency* is also preserved between the specialization of ℓ^{C_y} and the composition.

3.3.1 Sync extension

With an extension of any new synchronized artifact to the specialized process, we need to guarantee that the consistency is not interfered by the behavior of such artifact. This can be achieved by checking whether a lifecycle of an extended artifact can be completely composed within an embedded lifecycle of an artifact it synchronizes with, as shown in Definition 14 and Lemma 2.

Definition 14: (***ex-lifecycle and*** $\sqsupset$). Let lifecycle $\mathcal{L}'_{C_x}$ in specialized ACP model Π' be *B-consistent* to lifecycle $\mathcal{L}_{C_x}$ in base ACP model Π, and lifecycle $\mathcal{L}'_{C_y}$ of extended artifact C'_y in Π' synchronize $\mathcal{L}'_{C_x}$. We say $\mathcal{L}'_{C_y}$ as an *ex-lifecycle* of $\mathcal{L}'_{C_x}$ if there exists refined L-fragment $\ell_i \in lf(\mathcal{L}'_{C_x} \rightarrow \mathcal{L}_{C_x})$ such that $\mathcal{L}'_{C_y}$ has its whole lifecycle synchronized within ℓ_i, denoted $\ell_i \sqsupset \mathcal{L}'_{C_y}$.

Lemma 2: Based on Definition 12, given refined L-fragment $\ell \in lf(\mathcal{L}'_{C_x} \rightarrow \mathcal{L}_{C_x})$ synchronize with extended lifecycle $\mathcal{L}'_{C_y}$, the composed lifecycle between $\mathcal{L}'_{C_y}$ and ℓ is *B-consistent* to ℓ iff $\ell \sqsupset \mathcal{L}'_{C_y}$ and ℓ is an *ASL-fragment*.

For example, extended artifact A'_3 in Fig. 5 (b) has its whole lifecycle synchronized within artifact A'_2. This case can be explained in more detail by using Fig. 6 (a). We can see that L-fragment ℓ'_1 syncrhonizes with ℓ'_2 which represents the whole lifecycle of A'_3 ($\ell'_1 \sqsupset A'_3$); therefore, A'_3 is an *ex-lifecycle* of artifacts A'_2 and we have the composed lifecycle between ℓ'_1 and ℓ'_2 *B-consistent* to ℓ'_1. One can question that what would be the result if an extended artifact is synchronized with more than one existing artifact. For instance, in Fig. 5 (d), where two existing artifacts A'_1 and A'_2 synchronize with extended artifact A'_3. It is possible to see that the lifecycle of A'_3 is an *ex-lifecycle* of the lifecycle of A'_1 while it does not hold for A'_2. Based on Definition 13, although the condition of ex-lifecycle is satisfied for the synchronization between A'_3 and A'_1, however, it is not held for the synchronization between A'_3 and A'_2. Therefore, the result of iterative composition of such three lifecycles should not satisfy Lemma 2.

3.3.2 Sync refinement

– **Refinement of synchronization for existing artifacts**

We classify specialization patterns of the synchronization between two existing artifacts into two cases. First, one of two artifacts is refined while the other one remain unrefined. Second, both artifacts are refined. With the first case, the effected sync rule(s) of the refinement may have its state condition redefined on either the entry transition or the exit transition of an L-fragment. For the second case, both artifacts have their L-fragment refined. For example, in Fig. 7 (b), sync rules r'_{1-1} and r'_{1-2} are redefined for the exit transition of states s_m and s_n. For the second case, both artifacts have their L-fragment refined, e.g., Fig. 7 (c) and (d). Here, we define

atomic sync L-fragment to ensure a consistent synchronization behavior of synchronized embedded lifecycles between artifacts.

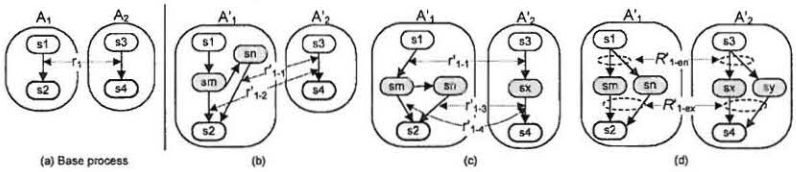

Fig. 7. Sync specializations of existing artifacts

For the refinement of two existing artifacts, we can apply the idea of *ASL-fragments* to check whether the refinement of these artifacts preserves the *B-consistency* of the base process. However, for single artifact refinement, we consider it as a special case since the refinement is applied on a single transition of one artifact not L-fragment. In order to make the transition to qualify L-fragment, so we expand its boundary to cover the source and target states of the transition. Then we can validly apply the ASL-fragments to check *B-consistency*. For instance, in Fig. 7 (b), we have an expanded L-fragment in artifact A'_2, which consists of states s_3 and s_4, synchronizes with L-fragment of A'_2.

Lemma 3: Let artifact lifecycles $\mathcal{L}'_{C_x}$ and $\mathcal{L}'_{C_y}$ in specialized ACP model Π' be *B-consistent* to artifact lifecycles $\mathcal{L}_{C_x}$ and $\mathcal{L}_{C_y}$ in base ACP model Π. Given L-fragment ℓ_m refines transition $\Rightarrow_i$ in $\mathcal{L}_{C_x}$ and L-fragment ℓ_n refines transition $\Rightarrow_j$ in $\mathcal{L}_{C_y}$, if ℓ_m and ℓ_n are *ASL-fragments*, then the composed lifecycle of ℓ_m and ℓ_n is *B-consistent* to the composed lifecycle of $\Rightarrow_i$ and $\Rightarrow_j$ (both $\Rightarrow_i$ and $\Rightarrow_j$ are considered as L-fragments with one transition).

– **Refinement of synchronization for extended artifacts**

Now, we extend the sync refinement between existing artifacts to be able to consider synchronizations between existing and extended artifacts. Recall sync extension, an extended artifact can be considered as an *ex-lifecycle* of an existing artifact if the lifecycle of the extended artifact is entirely synchronized within such existing artifact. We can say that if extended artifact C is used to refine sync rule r, then each artifact that is synchronized by r must have C as its ex-lifecycle. For example in Fig. 5 (d), artifact A'_3 is used to refine sync rule r_1 (between A'_1 and A'_2), and it can be considered as *ex-lifecycle* of both artifacts. A similar case is shown in Fig. 5 (e).

We now consider the scenario that has to deal with synchronizations for multiple extended artifacts, e.g., extended artifact A'_5 in Fig. 5 (f). Similar to the refinement between an existing artifact and an extended artifact, here we extend the sync extension method and *B-consistency* checking to the synchronization for multiple extended artifacts by introducing *transitivity* of ex-lifecycles. We say $\mathcal{L}'_{C_z}$ as a *transitive ex-lifecycle* of $\mathcal{L}'_{C_x}$ if $\mathcal{L}'_{C_z}$ is an ex-lifecycle of $\mathcal{L}'_{C_y}$ and $\mathcal{L}'_{C_z}$ is an ex-lifecycle of $\mathcal{L}'_{C_x}$. Here, we write $\ell_i \sqsupset^+ \mathcal{L}'_{C_z}$ if there exists refined L-fragment $\ell_i \in lf(\mathcal{L}'_{C_x} \rightarrow \mathcal{L}_{C_x})$ such that $\ell_i \sqsupset \mathcal{L}'_{C_y}$ and $\mathcal{L}'_{C_z}$ is an ex-lifecycle of $\mathcal{L}'_{C_y}$. For instance, artifact A'_5 in Fig. 5 (f) has its whole lifecycle synchronized within the lifecycle of artifacts A'_3, and A'_3 is an ex-lifecycle of A'_1; so, we have that A'_5 is a

transitive ex-lifecycle of A'_1. Now, we show how the *B-consistency* of the refinement for extended artifacts can be preserved in Lemma 4.

Lemma 4: Let artifact lifecycles $\mathcal{L}'_{C_x}$ and $\mathcal{L}'_{C_y}$ in specialized ACP model Π' be *B-consistent* to artifact lifecycles $\mathcal{L}_{C_x}$ and $\mathcal{L}_{C_y}$ in base ACP model Π, and let L-fragment ℓ_m refines transition $\Rightarrow_i$ in $\mathcal{L}_{C_x}$ and L-fragment ℓ_n refines transition $\Rightarrow_j$ in $\mathcal{L}_{C_y}$ such that ℓ_m and ℓ_n are *ASL-fragments*. Let extended lifecycle $\mathcal{L}'_{C_x}$ synchronize with ℓ_m or ℓ_n and a set of extended lifecycles Z^{ex} synchronize with $\mathcal{L}'_{C_x}$. The composed lifecycle of all artifacts in Z^{ex}, $\mathcal{L}'_{C_x}$, ℓ_m, and ℓ_n is *B-consistent* to the composed lifecycle of $\Rightarrow_i$ and $\Rightarrow_j$ iff $\forall \mathcal{L}'_{C_i}\{C'_i \in Z^{ex}\}, \ell_m \sqsupset^+ \mathcal{L}'_{C_i} \wedge \ell_n \sqsupset^+ \mathcal{L}'_{C_i}$.

For example in Fig. 5 (e), the composed lifecycle of artifacts A'_1, A'_2, and , A'_3 is *B-consistent* to the composed lifecycle of A'_1 and A'_2 since A'_3 is an *ex-lifecycle* of both A'_1 and A'_2. More complicated case is shown in Fig. 5 (f) having artifact A'_5 extended to artifact A'_3 which is used for the sync refinement of artifacts A'_1 and A'_2. We can see that A'_5 is considered as a *transitive ex-lifecycle* of A'_1 and A'_2; therefore, this refinement preserves the *B-consistency* of the base process.

3.3.3 Sync reduction

Based on the *artifact reduction* method we can remove an artifact in the specialized process model from its base process model. By doing this, all synchronizations between such removed artifact and all other related artifacts that synchronize with it are forced to be removed. We can see that the deleted artifact must be guaranteed not violating the *B-consistency* of the overall process. The deletion of one artifact should propagate to the removal of a part of lifecycle of other artifact(s) that it synchronizes with, i.e., if an artifact is removed, then the sync reduction is applied to all related lifecycles that such artifact synchronizes with including a synchronization that does not directly occur from such removed artifact.

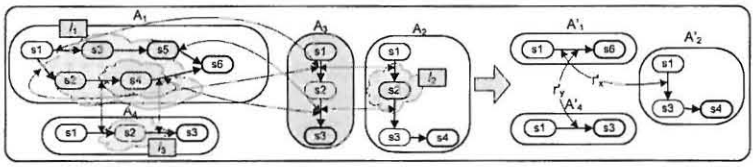

Fig. 8. Reduced lifecycles after applying sync reduction

In Fig. 8, artifact A_3 is to be removed in the specialized process. We can see that both L-fragments ℓ_1 of A_1 and ℓ_2 of A_2 are *reducible L-fragments*, and ℓ_1 has its sub L-fragment (containing state s_4) synchronized with L-fragment ℓ_3 of A_4. Such sub L-fragment in A_1 and ℓ_3 can be considered as *ASL-fragments*. The removal of artifact A_3 does not only cause the reduction of ℓ_2 but also propagates to the reduction of ℓ_3 as well. Therefore, we have ℓ_3 reduced into *reduced transition* $s_1 \Rightarrow_{\ell_3} s_3$ in A'_4 and all sync rules used within ℓ_3 reduced to *reduced sync rule* r'_y for the synchronization between A'_1 and A'_4. Here, we recall the definition of *ASL-fragment* in order to define *reducible L-fragment*, *reduced transition*, and *reduced sync rule* of an artifact.

Definition 15: (*Reducible L-fragment*). Let artifact lifecycle $\mathcal{L}'_{C_x}$ in specialized ACP model Π' be *B-consistent* to artifact lifecycle $\mathcal{L}_{C_x}$ in base ACP model Π, and let

artifact lifecycle $\mathcal{L}_{C_y}$ of $C_y \in \Pi.Z$ that synchronizes $\mathcal{L}_{C_x}$ in Π be removed from Π'. If there exists L-fragment ℓ in $\mathcal{L}_{C_x}$ such that both lifecycles ℓ and $\mathcal{L}_{C_y}$ are *ASL-fragments*, then ℓ is a *reducible L-fragment* of $\mathcal{L}_{C_x}$ for $\mathcal{L}_{C_y}$.

Definition 16: (*Reduced transition and reduced sync rule*).

- Given *reducible L-fragment* ℓ^{C_i} of lifecycle $\mathcal{L}_{C_i}$ for artifact lifecycle $\mathcal{L}_{C_y}$, ℓ^{C_i} can be reduced to *reduced transition* $s_i \Rightarrow_{\ell^{C_i}} s_j$, where s_i is the state entering to ℓ^{C_i} and s_j is the state exiting from ℓ^{C_i}.
- Given a set of *reducible L-fragments* L^{re} for removed artifact $\mathcal{L}_{C_y}$, for every reducible L-fragment ℓ_m in L^{re} and for every sync rule $r \in \varphi(\ell_m, \mathcal{L}_{C_y})$, r is to be (1) *removed* if ℓ_m synchronizes with only $\mathcal{L}_{C_y}$ and ℓ_m does not synchronize with other L-fragment in L^{re}; or (2) *reduced* into *reduced sync rule* r_ℓ where r_ℓ is used to synchronize between transition $\Rightarrow_{\ell_m}$ and transition $\Rightarrow_{\ell_n}$ if there exists ℓ_n in L^{re} such that ℓ_m synchronizes with ℓ_n.

Lemma 5: Let artifact lifecycle $\mathcal{L}_{C_y}$ that synchronizes artifact lifecycle $\mathcal{L}_{C_x}$ in base ACP model Π be removed from specialized ACP model Π'. Given L^{re} be a set of all possible *reducible L-fragments* for $\mathcal{L}_{C_y}$ and T^{re} be a set of *reduced transitions*, the composed lifecycle of all transitions in T^{re} is *B-consistent* to the composed lifecycle of all L-fragments in L^{re} and $\mathcal{L}_{C_y}$.

3.4 Sync specialization and B-consistency

Based on our comprehensive discussion on the three operations of sync specialization and their individual consistency and the *B-consistency* of ACP models, we now are able to define a complete consistency property of sync specialization.

Definition 17: (Synchronization consistent or *S-consistent*). Given ACP model Π' be a specialization of ACP model Π with ACP specialization $ps_{\Pi' \to \Pi}$ and sync specialization $ss_{(\mathcal{L}'_{C_x}, \mathcal{L}'_{C_y}) \to (\mathcal{L}_{C_x}, \mathcal{L}_{C_y})}$, $ss_{(\mathcal{L}'_{C_x}, \mathcal{L}'_{C_y}) \to (\mathcal{L}_{C_x}, \mathcal{L}_{C_y})}$ is said to be *S-consistent* iff, Lemma 2 is held for *sync extension*, Lemmas 3 and 4 are held for *sync refinement*, and Lemma 5 is held for *sync reduction*.

Theorem 2: Let ACP model Π' specialize ACP model Π with *ACP specialization* $ps_{\Pi' \to \Pi}$, Π' is a *B-consistent specialization* of Π based on ps iff,

- for every artifact $C'_i \in \Pi'.Z$ such that C'_i specializes $C_i \in \Pi.Z$, $ls_{\mathcal{L}'_{C_i} \to \mathcal{L}_{C_i}}$ is *B-consistent*; and,
- for every artifact C'_x and C'_y in Π', $ss_{(\mathcal{L}'_{C_x}, \mathcal{L}'_{C_y}) \to (\mathcal{L}_{C_x}, \mathcal{L}_{C_y})}$ is *S-consistent*.

Due to space limitation, proof is omitted. Theorem 2 has an importance of being able to assert the overall behavioral consistency between a specialized ACP model and its base model while only perform fragmental consistency checking based on a specialization, i.e., for an individual artifact and for only a synchronization between artifacts that is added, modified, or deleted in the specialized process. Notably, the model verification can suffer from the state exposition of compositional lifecycle if there are a number of artifacts having many states. Technically, we avoid the state space exposition problem by not composing all artifacts in the model.

4. Related Work and Discussion

The concept of business artifacts was introduced in [1] with the modeling concept of artifact lifecycles. Bhattacharya et al. [2] presented an artifact-centric process model with the study of necessary properties such as reachability of goal states, absence of deadlocks, and redundancy of data. Kuster et al. [6] presented a notion of compliance of a business process model with object lifecycles and a technique for generating the model from such set of lifecycles. Yongchareon and Liu [3, 11] proposed a process view framework to allow role-based customization and inter-org process modeling for artifact-centric business processes. In chorography settings, Van Der Aalst et al. [12] proposed an inter-org process-oriented contract with a criterion for accordance between private view and its public view modelled by open nets (oWFNs). Lohmann and Wolf [7] studied the generation of the interaction model from artifact-centric process models and used artifact composition to validate the model; and later, Lohamnn [13] proposed an approach to generate complaint and operational process model using policies and compliance rules. Fahland et al [16] presented conformance checking technique for interacting artifacts by decomposition into smaller problems so that conventional techniques can apply. Compared to our work, we also use similar composition technique to validate the overall behavior of the model; however, we focus on the fragmental behavior analysis for different methods of sync specializations (extension, refinement, and reduction).

Schrefl and Stumptner [4] studied the consistency criteria of the inheritance of object life cycles. They proposed necessary and sufficient rules for checking behavior consistency between object lifecycles. Some works have attempted to tackle the specializations of processes using state diagrams [8], the inheritance of (Petri-net based) workflow [10], and the behavior compatibility (consistency) between process models [14, 15]. However, these works only focused on the inheritance of single object lifecycle or workflow model. We extend their study to the synchronization between lifecycles. aAlthough [9] claimed that a specialization of processes cannot be viewed and treated analogously as a specialization of a single object, their work mainly treated the behavior of a process as the behavior of a single (dataflow) diagram. This approach still lacks detailed discussion and analysis of how objects and their interactions are considered in a specialized process, while our work takes into account the specialization of synchronizations between objects.

5. Conclusion and Future Work

This paper formally proposes the notion of process specialization for artifact-centric business processes with a comprehensive analysis of the behavioral consistency between a specialized process and its base process. For artifact-centric models, not only a local behavior of artifact but also the interaction behavior, which is described by sync business rules, can be specialized. One main outcome of this paper is the formal studies on the conditions for preserving the behavior consistencies of both intra-behavior and inter-behavior of artifacts in a specialized process based on our three proposed specialization methods (extension, refinement, and reduction). In the future, we will develop an efficient mechanism and a prototype for the consistency checking based on our proposed theorems.

References

1. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification, 2003, *IBM Systems Journal* 42(3), pp. 428–445.
2. Bhattacharya, K., Gerede, C, Hull, R: Towards Formal Analysis of Artifact-Centric Business Process Models, in *BPM* 2007. p. 288-304.
3. Yongchareon, S., Liu, C.: Process View Framework for Artifact-Centric Business Processes, In: OTM 2010, LNCS 6426, pp. 26–43.
4. Schrefl, M., Stumptner, M.: Behavior-Consistent Specialization of Object Life Cycles, 2002, *ACM Transactions on Software Engineering and Methodology*, vol. 11, pp. 92-148.
5. Lind-Nielsen, J., Andersen, H, Hulgaard, H, Behrmann, G, Kristoffersen, K, and Larsen, K.: Verification of Large State/Event Systems Using Compositionality and Dependency Analysis, 2001, *Formal Methods in System Design*, vol. 18(1), pp. 5-23.
6. Küster, J., Ryndina, K., Gall, H.: Generation of Business Process Models for Object LifeCycle Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 165–181. Springer, Heidelberg (2007)
7. Lohmann, N, Wolf, K.: Artifact-centric Choreographies, In: *ICSOC* 2010, LNCS 6470, pp. 32-46.
8. Wyner, G. M., Lee, J.: Process Specialization: Defining Specialization for State Diagrams, 2002, *Computational & Mathematical Organization Theory*, Vol. 8, pp. 133-155.
9. Wyner, G. M., Lee, J.: Defining specialization for dataflow diagrams, 2003, *Information Systems*, vol. 28, pp. 651-671.
10. Van Der Aalst, W.M.P., Basten, T.: Inheritance of workflows: an approach to tackling problems related to change, 2002, *Theoretical Computer Science*, vol. 270, pp. 125-203.
11. Yongchareon, S., Liu, C., Zhao, X.: An Artifact-centric View-based Approach to Modeling Inter-organizational Business Processes, In: *WISE* 2011, LNCS 6997, pp. 273-281.
12. Van Der Aalst, W.M.P, Lohmann, N, Masuthe, P, Stahl C, and Wolf, K.: Multipart Contrats: Agreeing and Implementing Interorganizational Processes, *The Computer Journal*, Vol. 53, No. 1, pp.90-106.
13. Lohmann, N.: Compliance by Design for Artifact-Centric Business Processes, In: *BPM 2011*, LNCS 696, pp. 99-115.
14. Weidlich, M., Mendling, J., Weske, M.: Efficient Consistency Measurement Based on Behavioral Profiles of Process Models, 2011, *IEEE Transactions on Software Engineering*, vol. 37 (3), pp.410-429.
15. Weidlich, M., Dijkman, R., Weske, M.: Deciding Behaviour Compatibility of Complex Correspondences between Process Models, In: BPM 2010, LNCS 6336, pp. 78-94.
16. Fahland, D., Leoni, M.D., van Dongen, B.F., van der Aalst, W.M.P.: Conformance Checking of Interacting Processes with Overlapping Instances, In: *BPM 2011*, LNCS 6896, pp. 345-361.
17. Kumaran, S., Liu, R., Wu, F.Y.: On the Duality of Information-Centric and Activity-Centric Models of Business Processes, In: *CAiSE 2008*, LNCS 5074, pp. 32-47.
18. Rosa, M.L., Reijers, H.A., Van Der Aalst, W.M.P., Dijkman, R.M., Mendling, J.: APROMORE: An advanced process model repository, 2011, *Expert Systems with Applications*, vol. 38, pp. 7029-7040.
19. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: *OTM 2008*, LNCS, vol. 5332, pp. 1152–1163.
20. Muller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures, In: *CAiSE 2008*, LNCS 5074, pp. 48-63.