Computing

Reusing Artifact-centric Business Process Models: A Behavioral Consistent Specialization Approach

--Manuscript Draft--

Manuscript Number:	COMP-D-18-00419R1
Full Title:	Reusing Artifact-centric Business Process Models: A Behavioral Consistent Specialization Approach
Article Type:	Original Research Article
Corresponding Author:	Sira Yongchareon, Ph.D Auckland University of Technology NEW ZEALAND
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Auckland University of Technology
Corresponding Author's Secondary Institution:	
First Author:	Sira Yongchareon, Ph.D
First Author Secondary Information:	
Order of Authors:	Sira Yongchareon, Ph.D
	Chengfei Liu
	Xiaohui Zhao
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	Process reuse is one of the important research areas that address efficiency issues in business process modeling. Similar to software reuse, business processes should be able to be componentized and specialized in order to enable flexible process expansion and customization. Current activity/control-flow centric workflow modeling approaches face difficulty in supporting highly flexible process reuse, limited by their procedural nature. In comparison, the emerging artifact-centric workflow modeling approach well fits into these reuse requirements. Beyond the classic class level reuse in existing object-oriented approaches, process reuse faces the challenge of handling synchronization dependencies among artifact lifecycles as parts of a business process. In this article, we propose a theoretical framework for business process specialization that comprises an artifact-centric business process model, a set of methods to design and construct a specialized business process model from a base model, and a set of behavioral consistency criteria to help check the consistency between the two process models.
Response to Reviewers:	Revision of Manuscript COM-D-18-00419 Reusing Artifact-centric Business Process Models: A Behavioral Consistent Specialization Approach First of all, we'd like to express our deepest gratitude to the reviewers for all of their highly valuable comments on our submission, which have been fully taken into account in this revision of this article. As detailed below, every attempt has been made to address the comments raised by the reviewers. In the remainder of this letter, we explain point-by-point on how the issues raised by the reviewers were addressed in this revised version of the manuscript, including details of the changes that were made in the paper. We hope that these changes will satisfy the requirements of the reviewers and make the article acceptable for publication in Computing.

Note that in the following discussions, the reviewers' comments are formatted in italics, while our responses appear in normal fonts.

Reviewer #1

Comment 1:One major weakness of this paper is the missing related work on data-driven, object-aware, and artifact-centric processes. [Künzle 2011], [Ksenia 2006], [Michael 2002], [Jochen 2007], [Lohmann 2013], [Fahland 2011]
Response: We thank the reviewer for indicating some missing related work. We added discussions of all the missing related work in the revised version as follows. [Künzle 2011], [Ksenia 2007 (MODELS 2006)], [Jochen 2007], [Lohmann 2013], and [Fahland 2011] are all added and discussed in the second to last paragraph of Section 6.1, while [Michael 2002] is added and discussed in the second paragraph of Section 4.2 and the third paragraph of Section 6.1, respectively.

Comment 2:As claimed in this paper, the major contribution of this paper is a theoretical study on the necessary and sufficient behavioral-preserving consistency conditions for specializing artifact lifecycles and their synchronization dependencies. However, this paper hadn't remarkable improvement compared with the work in [Yongchareon 2012]. Many concepts (such as ACP model, artifact refinement, artifact extension, and artifact reduction) are coincident both in these papers.
Response: We thank the reviewer for raising this issue. Actually, many of the notions used in our previous work [2012] are reused as a basis for our extension done in this work. In this extended paper, apart from some of newly added lemmas and theorems with proofs as the outcomes of our further study, we also extended our published paper in several aspects including (1) improvement to the specialization methods discussed in Sections 4.1 and 4.2, (2) improvement to the existing B-consistency notion, i.e., Definition 4.2 vs. Definition 9 in [2012], (3) improvement to the fragmental analysis method via the introduction of SL-fragments and S-region notions to help verify B-consistency, (4) a new propagation method for artifact reduction, and (5) a new tool prototype (Artifact-M). To make these extension points clear, we added this discussion in the last paragraph of Section 6.1 of the revised version.

Comment 3:in Definition 3.4, the description " pre_s(r, Ci) returns a set of states S…" is incorrect in some degree. Since Ci represents single artifact, thus pre_s(r, Ci) indicates a state rather than a set of states.
Response: We thank the reviewer for indicating this error. We corrected both the pre_s and post_s functions to return a single state and all the formulas that refer to these functions in the revised version.

Comment 4:Would you please give a clear definition of the term "composite task" appeared in Example 3.1? How to guarantee the state change of the relevant artifacts for the composite task (dispatchGoods(po,so), issueInvoice(po,iv))?
Response: We thank the reviewer for this advice. We changed the wording from "composite task" to "a concurrent invocation of two tasks" in the revised version. We use a post-condition in a business rule to imply that if the rule is successfully fire, then the state change of the affected artifacts must satisfy the post-condition defined. We added this explanation under Definition 3.4.

Comment 5:How to explain the term "artifact rules" in Figure 4.1?
Response: We thank the reviewer for indicating this. Actually, it is a typo and we corrected it to "business rules" in the revised version.

Comment 6:Some grammatical errors in this paper should be avoidable.
Response: We thank the reviewer for this feedback. In the revised version, we tried to fix all the English problems. We also consider hiring a professional proofreader prior to publishing.

Comment 7:All third level titles in Section 6 should be changed as second level titles. For example, "6.1.1.
Response: We thank the reviewer for pointing out this error. We fixed them in the revised version. For example 6.1.1 is changed to 6.1 now.

Reviewer #3

Comment 1:First, the paper lacks a clear discussion about the scope of the approach and assumptions made. In terms of object-oriented computing, what types of relations between the artifacts are considered in the proposed approach? For example, in OO, there are (exclusive and inclusive) containment relations, association relations between objects. In terms of business process models, what control dependencies between tasks are considered? In the paper, task dependencies are captured by business rules for individual artifacts and synchronisation across different artifacts, in an abstract way. It is not clear how certain control dependencies, such as event-based choice, milestone, multiinstance initiation and synchronisation etc (refer to the set of 20 classic workflow control-flow patterns for these control dependencies), are captured. It is worth noting that the reason of raising these questions here is not to ask the authors to consider all the OO relations and control-flow patterns, but to make it very clear upfront, what are considered (and can be supported) within the scope of the approach.
Response: We thank the reviewer for this comment. In contrast to the process-based/control-flow perspective, control-flows are not explicitly captured in the artifact model which is based on the use of business rules (ECA). The scope of the control dependencies can be derived from how the rules are defined. In summary, when compared to the activity-centric approach, our model supports the following control-flow patterns defined in [van der Aalst et al., 2003]: Sequence, Parallel Split, Synchronization, Exclusive Choice, Simple Merge, Multi-choice, Synchronizing Merge, Multi-Merge, and Discriminator. For example, our rules support choices through the use of the instate predicate of the event part (pre-condition in Definition 3.4) and a parallel split through the definition of a rule in the action part, e.g., rule r3 in Table 3.1. The relation between artifacts is scoped via the use of sync rule definition (Definition 3.5) to define synchronization dependency between them. We have added this discussion in the last paragraph of Section 3.1 in our revised version. In addition, our specialization relation is based a super and sub-type relationship defined in Definition 4.1, that maps the subtype to its supertype based on our three specialization methods: artifact refinement, artifact extension and artifact reduction, which are discussed in Sections 4.1 and 4.2.

Comment 2:Second, the notions of 'business rules' and 'synchronisation' require much precise and elaborated definitions. They are essential constructs to link / integrate individual artifacts into the context of a business process. IMHO, process-oriented or artifact-oriented approaches model business processes from different angle, while the behaviour/semantics (or the richness of control dependencies) of a business process should remain the same regardless of how it being modelled. Hence, in arifactoriented modelling, the control dependencies between tasks are captured by the business rules and the 'synchronisation' between the rules. According to Definition 3.5, the synchronisation is captured by 'a sync rule' which is simply a common business rule shared between different artifacts. Later when behavioural properties are proposed, e.g. in Sections 3.2 and 4.2, it is not clearly how sync rules play a role.
Response: We thank and agree with the reviewer for this comment. The Sync rules play the role in lifecycle composition which composes all the artifact lifecycles (taking into account their synchronizations) into a single composite lifecycle, aka. a synchronized product, which is semantically analogous to a process. Under the Definition 3.9 of our revised version, we added three inference rules used for lifecycle composition and an example of how sync rules play their role in a composition. The composition mechanism later is used for our fragmental consistency checking for each of our three methods of sync specialization discussed in Section 4.2.3.

Comment 3:Third, the approach is defined in a very object-oriented manner, and is heavy with abstract/formal definitions. It would be much better if most of the concepts are given an example of how they manifest in the context of business processes. The example presented in Section 2 and Section 5 can be used to illustrate the notions defined in section 4. Also, in the current form of the paper, the definitions are hard to read. I understand that the authors use set notations in their definition, and IMHO, it is strange to see something like r.\lamda, Cx.S, which are usually used in object-oriented programming.
Response: We thank the reviewer for this feedback. Following the advice, we added/revised some explanation and illustrated examples for complex definitions

	presented in Section 4 of the revised version. We hope our explanation along with examples help readers better understand the definitions. Also, we use a dot notation between a tuple and its element to refer to the element of that tuple. For example, we can write C.S for a set of states S of an artifact class C (i.e., C.S = S of C. Note that C is a tuple. We added an explanation about using it under Definition 3.1 and we hope this is acceptable.

Reusing Artifact-centric Business Process Models: A Behavioral Consistent Specialization Approach

SIRA YONGCHAREON, Auckland University of Technology
CHENGFEI LIU, Swinburne University of Technology
XIAOHUI ZHAO, University of Canberra

Process reuse is one of the important research areas that address efficiency issues in business process modeling. Similar to software reuse, business processes should be able to be componentized and specialized in order to enable flexible process expansion and customization. Current activity/control-flow centric workflow modeling approaches face difficulty in supporting highly flexible process reuse, limited by their procedural nature. In comparison, the emerging artifact-centric workflow modeling approach well fits into these reuse requirements. Beyond the classic class level reuse in existing object-oriented approaches, process reuse faces the challenge of handling synchronization dependencies among artifact lifecycles as parts of a business process. In this article, we propose a theoretical framework for business process specialization that comprises an artifact-centric business process model, a set of methods to design and construct a specialized business process model from a base model, and a set of behavioral consistency criteria to help check the consistency between the two process models.

Categories and Subject Descriptors: **H.1.0 [Models and Principles]**: General; **D.2.1 [Software Engineering]**: Requirements/Specifications—*Methodologies (e.g. object-oriented, structured)*

General Terms: Design

Additional Key Words and Phrases: Artifact-centric process modeling; Business process modeling; Business process reuse; process specialization; Behavioral consistency

1. INTRODUCTION

With business processes becoming more complex and dynamic, process model reuse is highly required to improve the efficiency of business process modeling. In this respect, organizations demand a more effective and systematic approach to reusing their already designed process models. As found in object-oriented software reuse, several benefits have been shown in practice to incrementally improve and support highly complex but loosely coupled software components. A reuse mechanism is seen to be very attractive and important as it provides several economic benefits, such as cost/time-savings, qualitative improvements, and economies of scale. Similarly, business process reuse should aim to support a flexible extension and adaptation of existing process models.

A notion of artifact-centric (operational) business process modeling has been proposed to shift the way we design and model a process from the activity-oriented perspective of a process to understanding key business-relevant entities and how they evolve through the operations of the process [Nigam and Caswell, 2003]. The approach is beneficial in enabling a natural modularity and componentization of business operations and varying levels of abstraction [Cohn and Hull, 2009] as it provides an intuitively natural, robust, and flexible structure for understanding and specifying business processes and operations in four explicit, inter-related but separable dimensions in the specification of business processes – which are *business artifacts*, *artifact lifecycles*, *services* (a.k.a. tasks), and *associations* (between artifacts and services) [Hull, 2008]. Associations are generally implemented by means of business rules. The approach has been adopted and proved practically successful in industries including *insurance* (e.g., [Chao et al., 2009]), *insurance* (e.g., [Kumaran et al., 2008]), and *healthcare* (e.g., [Künzle and Reichert, 2011; Chiao et al., 2013]).

The artifact-centric approach naturally lends itself well to both the object-oriented and service-oriented design principles, as it focuses on the design of both business artifacts involved in a process and business tasks/services performing operations on such artifacts. Built based on an object-oriented structure, artifact-centric process models can achieve a higher level of adaptability, reusability, and extensibility [Yongchareon et al., 2012]. We observed that the existing object specialization approach to software (reuse) design can provide a fundamental basis for our study. Although the object specialization/generalization has been well studied in the object-oriented analysis and design in the last decade (e.g., in [van Der Aalst and Basten, 2001, 2002; Wyner and Lee, 2002; Schrefl and Stumptner, 2002]), the specialization/generalization of process models especially where the objects (artifacts) of the process are primarily concerned has never been studied. Dependencies between artifacts become a major concern as they may cause the behavior of a specialized process to be inconsistent with (and unobservable from) the behavior of its base process. It raises a non-trivial issue when an artifact and its dependency can be added, modified, or removed in the specialized process. In other words, it is possible that one can define a specialized process by modifying or adding synchronizations between existing objects or inserting/removing an object into/from the specialized process. These specialization operations must be guaranteed not violating the behavioral consistency between the base process and its specialization. Especially, the interaction (or synchronization) dependencies between artifacts in a process to be reused need to be considered to ensure that the process can be consistently observed from its base process, therefore, allowing processes to be represented and monitored from different levels of granularity and therefore comparing them across the specializations can be achieved more effectively and naturally [Hull, 2008].

In this article, we propose a systematic framework for specializing *Artifact-Centric Process* (ACP) models, which comprises several dependent artifact classes. The framework includes three construction methods for process specialization: *artifact refinement*, *artifact extension*, and *artifact reduction*. We study the dynamic behavior of artifacts and their synchronization dependency in a process and formulate a set of behavioral consistency rules and a fragmental behavior analysis to help validate the consistency between two processes. With our specialization approach, process modelers can reuse an already defined ACP model at both artifact level and process level. A specialized process model can be consistently observed at the different level of specialization hierarchy, thus, allowing both artifact and process instances of the specialized process to be aggregated and reported at a more generic level. More importantly, with our approach, we can ensure compliance in specialized processes to company-wide rules and legal requirements and at the same time improved reusability as well as ensuring the correct adaptations of processes. Note that the work reported in this paper has extended the work in [Yongchareon et al., 2012] in several aspects, e.g. an extended framework with new and improved specialization methods and notions, comprehensive discussion and formal analysis, a number of new necessary and sufficient lemmas and theorems along with detailed proofs as theoretical outcomes of our study.

We organize the rest of this paper as follows. Section 2 introduces our research motivation along with an example and problem statements. Section 3 discusses ACP models. Section 4 discusses our approach to process specialization and the behavioral consistency between two process models. Section 5 illustrates a case study on the specializations of a purchasing process. Section 6 discusses related work, and the summary of this article is presented in Section 7.

2. MOTIVATION AND PROBLEM STATEMENTS

Here, we introduce a purchasing business process (between a buyer and a supplier) and explain how it is modelled using the artifact-centric approach, as shown in Figure 2.1. We first start our discussion by identifying business artifacts that are involved in the business process and describing how they can be modeled. We can see that the purchasing process consists of three core business artifacts, which are *Invoice*, *Shipping Order*, and *Purchase Order*. The interrelations between artifacts can be drawn using dashed lines as *synchronization dependencies* of such artifacts.

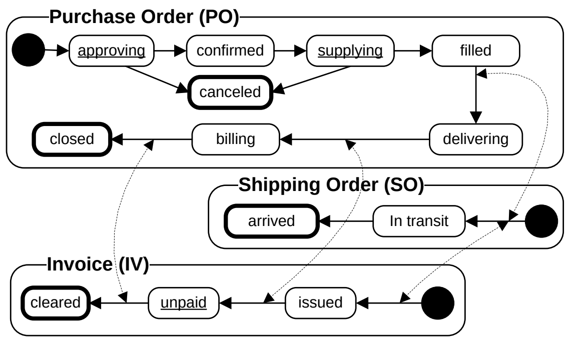

Figure 2.1: An overall view of artifact-centric purchasing processes

Now, let us briefly describe this process in more detail. At the start, a purchaser creates a *purchasing order* consisting of a list of goods. Immediately after created, the purchase order is waiting for the approval. Once approved, the seller confirms and sends the order to a selected supplier. When After the purchase order is received, the supplier starts supplying the goods required for the order. The supplier cancels the order if the goods are out of stock. Otherwise, the order is filled with goods and a *shipping order* is created for delivery. When the goods are in transit, the supplier issues an *invoice*, and then sends it to the buyer and waits for the payment. The supplier closes the order once the payment has been cleared by the buyer.

For example, we consider the case of a retailer and a supplier that agree to have online purchasing process separate with offline purchasing process based on the reuse of existing generic purchasing process. The supplier and the retailer require some systematic approach that can help them design and model business processes to satisfy such requirements. With the artifact-centric business process modeling, we borrow existing object-oriented specialization methods and apply them to the specialization of a process. We can consider a process model as a class container that consists of interrelated artifact classes. Figure 2.2 illustrates our case of having a *Generic purchasing* process model and two specialized models, which are *Online purchasing process* and *Offline purchasing process*. The former only provides services to retail Internet customers, while the latter is defined for retailers and wholesalers. In Figure 2.2, a round-shape rectangle represents an artifact class used in a process model (drawn by an aggregation relationship). A specialization relationship between ACP models indicates that one ACP model is a specialized model of one another (base model). The analogous meaning of the specialization relationship is also applied for artifact classes.

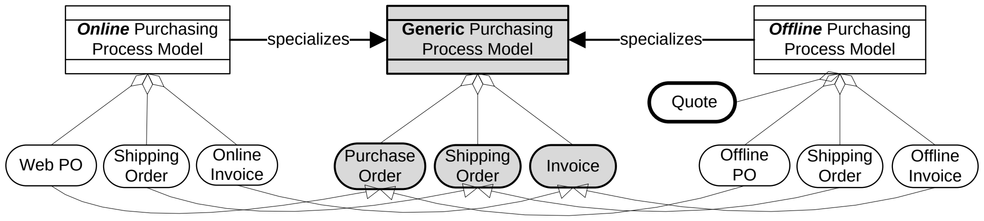

Figure 2.2: Generic purchasing process with two specializations.

The generalization and specialization of ACP models allow different abstraction levels of both process-related and artifact-related management and reports. For instance in Figure 2.2, regional sale managers may want to monitor high-level aggregated information of how many orders now are shipped but their invoices have not yet been cleared, while online sales executives may want to only manage their web orders sold through the website. In order to achieve higher-level abstraction, instances of *Web PO* and instances of *Offline PO* must be aggregable (or comparable) and representable as instances of *Purchase Order,* i.e., the behaviors of *Web PO* and *Offline PO* are consistently observable at the *Purchase Order*'s view. Analogously, from the process-oriented perspective, the instances of both *Offline purchasing* process and *Online purchasing* process must be consistently observable in the *Generic purchasing* process. We can see some significant advantages of using an artifact-centric modeling approach to model business process from the above benefits as also perceived in the object-oriented design. The reuse can be achieved at both business artifact level and process level; thus, increase the level of modeling efficiency and adaptability. Apart from that, in traditional activity-centric process modeling approach, ways to manage and monitor business data yet remain difficult and inefficient. Technically, this is because the data in the activity-centric processes is not concerned as the first-class citizen of the model; thus, additional mechanisms to capture such data or model transformations are required.

Dependencies between artifacts become a major concern since they can lead the behavior of a specialized process inconsistent with the behavior of its base process. Especially, it raises a non-trivial issue when an artifact and its dependency can be added, removed, or modified in the specialized process. Consider artifact classes defined in a specialized ACP model. One can think that for every artifact class, each artifact should inherit its base class of the base ACP model, e.g., the *Online* purchasing process has each of its artifact models derives from its base artifact. For example, *Web PO* is considered a specialization of *Purchase Order*. We can also notice that the synchronization between *Web PO* and the other artifact(s) is modified as part of the specialization. In addition, it is possible that a newly defined artifact class can be added to the specialized process, e.g. in Figure 2.2, the *Quote* artifact is added to the *Offline purchasing* process. Obviously, extending the *Quote* artifact to the process needs extra synchronization with some other artifact(s), e.g., *Offline PO*.

Technically, the specialization of an ACP model brings in the following key questions.

—What is an approach to allowing process modelers to systematically define and specialize the process model?

—How to preserve the behavioral consistency between a base process and its specialized process?

To address the above questions, we propose a theoretical framework for specializing an ACP model, as overviewed in Figure 2.3. The framework consists of three core components: (1) an ACP model, (2) a set of methods to define and construct a specialization of process models, and (3) a set of behavior consistency rules and a validation mechanism.

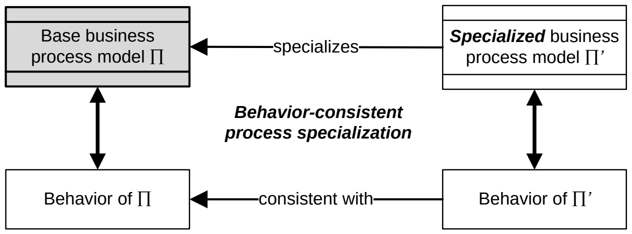
Base business process model Π
specializes
Specialized business process model Π'
Behavior-consistent process specialization
Behavior of Π
consistent with
Behavior of Π'

Figure 2.3: Overview of our framework

3. ARTIFACT-CENTRIC PROCESS (ACP) MODEL

In this section, we introduce definitions and syntax for modeling an artifact-centric process and its related properties.

3.1 ACP Model

We formally introduce an *ACP model* for capturing specification of a business process based on the model presented in our previous work [Yongchareon et al., 2012, 2015]. The *ACP model* has the following core parts including *artifact classes*, *artifact schema*, *tasks*, and *business rules*.

Definition 3.1: *(Artifact class).* We denote C for an *artifact class* and it is a tuple (S, s^{init}, S^f) where $S = \{s_1, s_2, \ldots, s_y\}, s_i \in S (1 \leq i \leq y)$ is a state, s^{init} denotes an initial state, and $S^f \subset S$ is a finite set of final states.

For the rest of the paper, we use a dot notation (.) between a tuple and its element to refer to the element of that tuple. For example, from *Definition 3.1*, we can write $C.S$ for a set of states S of an artifact class C.

Definition 3.2: *(Artifact schema).* We denote $Z = \{C_1, C_2, \ldots, C_x\}$ as an *artifact schema,* where $C_i \in Z (1 \leq i \leq x)$ is an artifact class.

Definition 3.3: (Task). We denote $V = \{v_1, v_2, \ldots, v_x\}$ as a set of business tasks (or services).

Definition 3.4: *(Business Rule).* We define business rules to regulate the execution of a task(s). A task(s) is executed under a business rule if the pre-conditions of the rule is satisfied. In addition, the post-conditions of a business rule are also defined to restrict the conditional effect after the rule is invoked. A *business rule* r is triple (λ, β, v) where,

—λ and β are finite sets of pre-conditions and post-conditions. Both condition sets are defined in a conjunctive normal form (CNF) using an *instate* predicate, which can be written as $instate(C, s)$ if state $s \in C.S$ of an artifact class C is active.

—$v \in V$ is a set of tasks that are fired upon the satisfaction of the rule.

We use a post-condition in a business rule to imply that if the rule is successfully fire, then the state change of the affected artifacts must satisfy the post-condition defined and to guarantee the existence of artifact's state changes, the following statement must hold. For some states $s_i, s_j \in C.S$, if there exists $instate(C, s_i)$ in a pre-condition of a business rule, then there exists $instate(C, s_j)$ in the post-conditions of the rule.

Table 3.1 shows key business rules with necessary conditions for the purchasing process introduced in our motivating example.

Table 3.1: Example of business rules

$\boldsymbol{r_1}$ **:** A buyer confirms a *PO*		
λ	$instate(po, approving)$	
v	$confirm(po)$	
β	$instate(po, confirmed)$	
$\boldsymbol{r_2}$: A supplier accepts a *PO* and starts supplying the goods ordered in the *PO*		
λ	$instate(po, confirmed)$	
v	$supply(po)$	
β	$instate(po, supplying)$	
$\boldsymbol{r_3}$: A supplier dispatches goods and simultaneously creates a SO and *IV* for the PO		
λ	$instate(po, filled) \wedge instate(iv, init) \wedge instate(so, init)$	
v	$dispatchGoods(po, so)$	$issueInvoie(po, iv)$
β	$instate(po, delivering) \wedge instate(iv, issued) \wedge instate(so, in\_transit)$	
$\boldsymbol{r_4}$ **:** A supplier receives a payment for the IV and closes the PO		
λ	$instate(po, billing) \wedge instate(iv.unpaid)$	
v	$closeOrder(po, iv)$	
β	$instate(iv.cleared)\ instate(po, closed)$	

To help define the behavior of an artifact, we define the following functions.

—$pre\_s(r, C_i)$ returns a state s if there exists $instate(C_i, s)$ in a precondition λ of r.

—$post\_s(r, C_i)$ returns a state s if there exists $instate(C_i, s)$ in a post-condition β of r.

We also define a *sync rule* to capture simultaneous state changes of artifacts via a single rule.

Definition 3.5: *(Sync rule)*. We call *sync rule* r if for some artifacts C_x and C_y, if there exists $instate(C_x, s_i)$ and $instate(C_y, s_m)$ in a precondition λ of r and $instate(C_x, s_j)$ and $instate(C_y, s_n)$ in a postcondition β of r, where $s_i, s_j \in C_x.S$ and $s_m, s_n \in C_y.S$.

Example 3.1. In Table 3.1, a business rule r_4 is a sync rule that is used to change the state of the *PO* artifact (*billing* → *closed*) and the state of *IV* artifact (*unpaid* → *cleared*) simutanously. Similarly, a sync rule r_3 is used for the synchronization of the three artifacts *PO*, *SO,* and *IV* (notice a concurrent invocation of two tasks upon two different artifacts in r_3).

Definition 3.6: *(ACP model)*. We can define $\Pi = (Z, V, R)$ for an *ACP model* that consists of an artifact schema Z, tasks V and business rules R.

It is worthwhile to reiterate that, in contrast to the process-based/control-flow perspective, control-flows are not explicitly captured in the artifact model which is based on the use of business rules (ECA). The scope of the control dependencies can be derived from how the rules are defined. In summary, when compared to the activity-centric approach, our model supports the following control-flow patterns defined in [van der Aalst et al., 2003]: *Sequence, Parallel Split, Synchronization, Exclusive Choice, Simple Merge, Multi-choice, Synchronizing Merge, Multi-Merge, and Discriminator*. For example, our rules support choices through the use of an *instate* predicate of the event part (pre-condition) and a parallel split through the definition of a rule in the action part, e.g., in *Example 3.2.* The relation between artifacts is defined via the use of sync rules to define synchronization dependency between them.

3.2 Behavioral properties

This section discusses the artifact's behavior, i.e. its lifecycle evolution from the initial to the final states. Formally, we use the notion of Labeled Transition System to model an artifact's lifecycle as well as a synchronized product of all the artifact lifecycles defined in an ACP model to represent the behavior of the entire process.

Definition 3.7: *(Lifecycle of artifact, $\overset{*}{\Rightarrow}$).* Given $C_i = (S_i, s_i{}^{init}, S_i{}^{f})$ be an artifact class in Π, we denote $\mathcal{L}_{C_i}$ for a (artifact) *lifecycle* of C_i, which can be defined as a tuple $(S, s^{init}, \Rightarrow)$ where,

—set of states $S = S_i$, an initial state $s^{init} = s_i{}^{init}$,

—$\Rightarrow \subseteq S \times R_i \times G_i \times S$ is a set of state transitions, where,

—$R_i \subseteq \Pi.R$ are business rules that trigger state changes of an artifact C_i such that

$$\forall r \in \Pi.R, \exists s_x, s_y \in, C_i.S, s_x = pre_{s(r,C_i)} \wedge s_y = post\_s(r, C_i) \rightarrow r \in R_i\,,$$

—G_i are guards that contain state pre-conditions from every business rule in R_i, of which refers to some state of the other artifact, i.e.,

$$G_i = \bigcup_{j=1}^{|\Pi.Z|} \{instate(C_j, s) \big| \exists r \in R_i, \exists C_j \in \Pi.Z, s = pre\_s(r, C_j) \wedge C_j \neq C_i\}\,,$$

We also denote $\overset{*}{\Rightarrow}$ for a reflexive transitive closure of $\Rightarrow$ and we write $s_i \overset{*}{\Rightarrow} s_j$ if a state s_j is reachable from a state s_i. Next, we define a *lifecycle occurrence* as a sequence of states and corresponding transitions.

Definition 3.8: *(Lifecycle occurrence or L-occurrence).* Let Π and $\mathcal{L} = (S, s^{init}, \Rightarrow)$ be an *ACP model* and a *lifecycle* (of either an artifact class or Π), respective, we can define a particular sequence of states, regarding a sequence of firing transitions in $\mathcal{L}$, as an *L-occurrence* of $\mathcal{L}$. An *L-occurrence* in $\mathcal{L}$ from a state s_x to a state s_y is denoted as $\sigma^{\mathcal{L}:\, s_x \rightarrow s_y} = (s_x, \ldots, s_y)$ where $s_x, s_y \in S$ such that,

$$\forall s \in \sigma^{\mathcal{L}:\, s_x \rightarrow s_y}, s_x \overset{*}{\Rightarrow} s \wedge s \overset{*}{\Rightarrow} s_y.$$

We write a transition $s_s \overset{r[g]}{\Longrightarrow} s_t$ to refer to the change of state from a source state s_s to a target state s_t if r is fired and g holds true. We may write $s_s \Rightarrow s_t$ without its superscription in a clear context.

Next, we define an *ACP lifecycle* to describe the behavior of an ACP that comprises a set of synchronized lifecycles of artifacts defined in the ACP model. Technically, the lifecycle of a whole ACP process can be obtained by composing every artifact lifecycle defined for the process. The lifecycle of an ACP model can be constructed using a lifecycle composition technique proposed in [Yongchareon et al., 2012]. The composed lifecycle of ACP model is used for the verification of the model.

Definition 3.9: *(Lifecycle composition and ⊗).* Given $\mathcal{L}_i = (S_i,\, s_i^{init},\, \Rightarrow_i)$ and $\mathcal{L}_j = (S_j,\, s_j^{init},\, \Rightarrow_j)$ be two lifecycles, we denote $\mathcal{L}_c = \mathcal{L}_i \otimes \mathcal{L}_j = (S_c,\, s_c^{init},\, \Rightarrow_c)$ for the *lifecycle composition* of $\mathcal{L}_i$ and $\mathcal{L}_j$ where,

—$S_c \subseteq \mathcal{L}_i.\mathrm{S} \cup \mathcal{L}_j.\mathrm{S}$ are states,

—$s_c^{init} = (\mathcal{L}_i.s^{init}, \mathcal{L}_j.s^{init})$ is an initial state,

— $\Rightarrow_c \subseteq S_c \times \Pi.R \times G_c \times S_c$ are transitions in the composed lifecycle.

We can formulate $\Rightarrow_c$ of the *composed* lifecycle $\mathcal{L}_c$ by applying the three inference rules defined below.

Let $g[s^x_{\mathcal{L}_i}/state(\mathcal{L}_i,\ s_x)]$ denote that a state $s^x_{\mathcal{L}_i}$ in a guard g is *substituted* (denoted by symbol /) by $true$ or $false$ (of state predicate) depending on whether the local state of $\mathcal{L}_i$ is s_x. We can formulate a transition relation $\Rightarrow_c$ of a *composed lifecycle* $\mathcal{L}_c$, by using the following three inference rules.

$$\frac{(s^x_{\mathcal{L}_i}, r, g_1, s^y_{\mathcal{L}_i}) \in \Rightarrow_i}{\left(\left(s^x_{\mathcal{L}_i}, s^x_{\mathcal{L}_j}\right), r, g_c, \left(s^y_{\mathcal{L}_i}, s^x_{\mathcal{L}_j}\right)\right) \in \Rightarrow_c,\ g_c = g_1\left[s^x_{\mathcal{L}_j}/state(\mathcal{L}_j, s_x)\right]} \tag{3.1}$$

$$\frac{(s^x_{\mathcal{L}_j}, r, g_2, s^y_{\mathcal{L}_j}) \in \Rightarrow_j}{\left(\left(s^x_{\mathcal{L}_i}, s^x_{\mathcal{L}_j}\right), r, g_c, \left(s^x_{\mathcal{L}_i}, s^y_{\mathcal{L}_j}\right)\right) \in \Rightarrow_c,\ g_c = g_2\left[s^x_{\mathcal{L}_i}/state(\mathcal{L}_i, s_x)\right]} \tag{3.2}$$

$$\frac{(s^x_{\mathcal{L}_i}, r, g_1, s^y_{\mathcal{L}_i}) \in \Rightarrow_i \ \wedge\ (s^x_{\mathcal{L}_j}, r, g_2, s^y_{\mathcal{L}_j}) \in \Rightarrow_j}{\left(\left(s^x_{\mathcal{L}_i}, s^x_{\mathcal{L}_j}\right), r, g_c, \left(s^y_{\mathcal{L}_i}, s^y_{\mathcal{L}_j}\right)\right) \in \Rightarrow_c,\ g_c = g_1\left[s^x_{\mathcal{L}_j}/state(\mathcal{L}_j, s_x)\right] \wedge g_2\left[s^x_{\mathcal{L}_i}/state(\mathcal{L}_i, s_x)\right]} \tag{3.3}$$

Rule (3.1) and Rule (3.2) are applied when a business rule r is fired on only individual lifecycles $\mathcal{L}_i$ and $\mathcal{L}_j$, respectively. Rule (3.3) is applied when a *sync rule* r is fired on both lifecycles $\mathcal{L}_i$ and $\mathcal{L}_j$. As the three inference rules apply the substitution of state conditions of two lifecycles in the composition, a reference to an external lifecycle is not to be replaced.

Figure 3.1 illustrates an example of lifecycle composition of two lifecycles. In this figure, we can see that r_3 is a *sync rule* used to synchronize an artifact C_1 with artifacts C_2 and C_3. Similarly, a sync rule r_4 is used to synchronize artifacts C_2 and C_4.

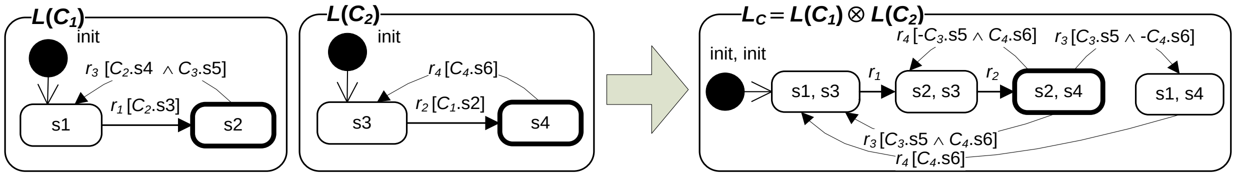

Figure 3.1 An example of lifecycle composition (taken from [Yongchareon et al, 2012])

Definition 3.10: *(ACP Lifecycle).* We denote $\mathcal{L}_\Pi$ for an *ACP lifecycle* of an *ACP model* Π and $\mathcal{L}_\Pi$ can be constructed based on applying *lifecycle composition* on every artifact of Π.

We also define a *soundness* property for a desirable behavior of a lifecycle of an artifact or ACP.

Definition 3.11: *(Sound lifecycle).* A *lifecycle* $\mathcal{L} = (S, s^{init}, \Rightarrow)$ of an *ACP Model* $\Pi = (Z, V, R)$ is *sound* if:

—every $r \in \Pi.R$ fires one and only one transition,

—every non-final state s is reachable from s^{init} and s eventually reaches any final state, i.e.,

$$\forall s \in S \backslash S^f, \exists s_f \in S^f, s^{init} \overset{*}{\Rightarrow} s\ \wedge\ s \overset{*}{\Rightarrow} s_f,$$

—every final state s_f is reachable from the initial state, i.e.,

$$\forall s_f \in S^f, s^{init} \overset{*}{\Rightarrow} s_f\ ,$$

4. PROCESS SPECIALIZATION

In this section, we present our process specialization approach. The discussion about specialization methods for artifact-centric processes is presented in Section 4.1 and then behavioral-consistent specialization is discussed in Section 4.2.

Figure 4.1 illustrates an example of hierarchies of *ACP specializations* built up by applying different ways of process specialization methods. We define two layers for the specialization. The first layer represents a set of artifact classes serving as base artifacts disregarding any specific process-related context. The second layer represents the hierarchy of specializations where each specialization is for a specific process context. In the second layer, the relationships of all inheritances of a generic process model form a hierarchical tree structure having the generic model as the root of the hierarchy. The root level represents a *generic ACP model*. Each generic model can be created by considering a set of *generic* artifact classes and a set of *generic* business rules used for coordinating the interaction between those artifacts for achieving the goals of a specific process. The lower levels present *specialized ACP models* where each of which inherits its corresponding base model. In the hierarchy, we call *subtype* for a specialized process model (or a specialized artifact class) of a process model (or an artifact class) it inherits, which is called *super-type*. Note that we exclude multiple inheritances (i.e., one model specializes two or more models) from our study as we believe that the concept is unnecessarily complex for modeling business processes.

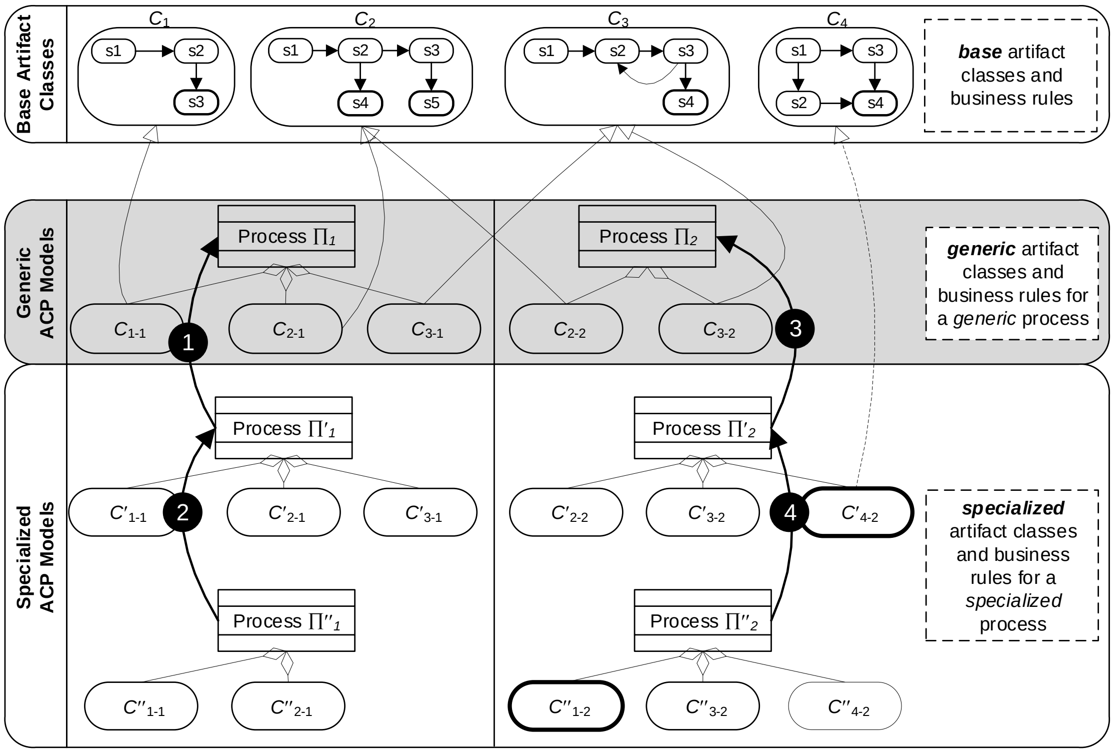

Figure 4.1: ACP specializations with four different specialization methods

4.1 Process specialization methods

Here, we classify *ACP specializations* into two categories based on the existence of an artifact class in the inherited process model and in its base model: *full specialization* and *partial specialization*.

—*Full specialization* describes the situation where for every artifact class in a specialized model, each class is a *refined* subtype of its super-type in the base model.

—*Partial specialization* means that there exists an artifact class in a specialized model but not in the base model, or vice versa.

Example 4.1. In Figure 4.1, specialization (1) is a *full specialization* since each artifact in a specialized process Π'_1 inherits its supertype in a generic process Π_1. Specializations (2), (3), and (4) are *partial specializations*. Specialization (2) constructs a specialized process Π'_1 from its super process Π'_1 by inheriting artifacts C''_{1-1} and C''_{2-1} from artifacts C'_{1-1} and C'_{2-1}, respectively, but not an artifact C'_{3-1}.

Specialization (3) shows an extension of an artifact by adding a new artifact C'_{4-2} to a specialized process Π'_2, Lastly, specialization (4) shows a substitution (i.e., reduction then extension) of an artifact C'_{2-2} by an artifact C''_{1-2} in a specialized process Π''_2. Revisiting our example in Figure 2.2, the *Online purchasing* process is a *full specialization* of the *Generic purchasing* process, while the *Offline purchasing* process is a *partial specialization*.

Based on the above categories, we propose the following three construction methods for ACP model specialization.

—*Artifact refinement*. Inheriting an artifact class from a base model by refining a base business rule(s) and its base state(s). The pre-conditions and the post-conditions of the modified rule may contain a newly defined state(s) in the subtype derived from the state of its super-type. The refinement can also be performed on a sync rule.

—*Artifact extension*. Including an additional artifact in the specialized model (which it does not previously exist in the base model). Introducing a new artifact to a model requires not only a new rule(s) but also a sync rule for synchronizing the new artifact with the other artifact(s) in the model.

—*Artifact reduction*. In contrast to artifact extension, process modelers can remove an artifact found in the base model if it is no longer required in the specialized model.

One can observe that whenever an artifact is added into or removed from a specialized process, the behavior of the process should be changed. Even for a fully specialized process that has each artifact inheriting its super-type in the base process, the behavior of the former may be inconsistent with the behavior of the latter as some of the specialized artifacts may have their behavior changed (or refined). The behavior of one artifact and the change of the state dependency between artifacts can affect the overall behavioral consistency of the process. Specifically, we need to observe the behavior of the specialized process and its base process and study conditions that make the behavior of the specialized process constructed by any of the above three methods consistent with its base process.

4.2 Behavioral-consistent specialization

In this section, we discuss how to specialize an ACP model while preserving the *behavioral consistency* of its base ACP model. Then, we introduce an *ACP specialization function* that maps a specialized model to its base model. Lastly, we discuss how each of the specialization methods (*refinement*, *extension*, and *reduction*) can be achieved with the guarantee of behavioral consistency.

In traditional object-oriented design approaches, the notion of behavior consistency can be divided into an observation consistency and invocation consistency. The observation consistency is used to guarantee that if a feature added in a subtype is ignored or a feature refined at the subtype are considered unrefined, any processing state of the subtype can be observed from the view of the super-type. On the other hand, the invocation consistency means that an instance of the subtype can be used in the same way as an instance of the super-type [Schrefl and Stumptner, 2002]. In our research, we restrict our study on observation consistency as our research focuses on the monitoring and reporting aspects of business processes. The observation consistency, on the one hand, can be used to guarantee that a current processing state of an artifact(s) or process always appear at a higher abstraction level. On the other hand, it can be used to ensure that an artifact or rule extended or modified in the specialized model does not interfere with the other artifact(s) or rule(s) derived from its super-type. Technically, handling synchronization dependencies between artifacts to ensure observation consistency is not trivial in process specialization.

Definition 4.1: *(ACP Specialization)*. Let $\Pi = (Z, V, R)$ and $\Pi' = (Z', V', R')$ be an ACP model and the specialization of an ACP model, respectively, we define *ACP specialization function* $ps_{\Pi' \to \Pi} : Z' \cup V' \cup R' \to Z \cup V \cup R \cup \{\varepsilon\}$ as a total function used to map each of the elements in Π' onto either each of the elements in Π or a *null element* ε. It is noted that we allow the use of ps for mapping $C'.S$ onto $C.S$ where $C' \in Z'$ and $C \in Z$.

The *refinement*, *extension*, and *reduction* of artifact can be expressed by the *ACP specialization function* ps as follows.

—*Artifact refinement*. Let an artifact $C' \in Z'$ be a refinement of an artifact $C \in Z$, a set of business rules $R_x{}' \subseteq R'$ be a refinement of a business rule $r \in R$, a set of tasks $V_y{}' \subseteq V'$ be a refinement of a tasks $v \in V$, and a set of states $S_z{}' \subseteq C'.S$ be a refinement of a state $s \in C.S$. The statements below must be satisfied for ps:

(a) $ps(C') = C$,

(b) $\forall r_i{}' \in R_x{}', ps(r_i{}') = r$,

(c) $\forall v_j{}' \in V_y{}', \exists r_i{}' \in R_x{}',\ ps(v_j{}') = v\ \wedge v_j{}' \in r_i{}'.V$,

(d) $\forall s_k{}' \in S_z{}', \exists r_i{}' \in R_x{}', ps(s_k{}') = s \wedge\ s_k{}' \in pre\_s(r_i{}', C') \cup\ post\_s(r_i{}', C')$

—*Artifact extension*. Let an artifact $C' \in Z'$, a set of business rules $R_x{}' \subseteq R' \setminus R$, and a set of tasks $V_y{}' \subseteq V' \setminus V$ be extended into Π'. The following statements must be satisfied for ps:

(a) $ps(C')\ =\ \varepsilon$,

(b) $\forall r_i{}' \in R_x{}', ps(r_i{}') = \varepsilon$,

(c) $\forall v_j{}' \in V_y{}', \exists r_i{}' \in R_x{}',\ ps(v_j{}') = \varepsilon$.

—*Artifact reduction*. Let an artifact C be removed from Z in Z'. The following statements must be satisfied for ps:

(a) $\nexists C' \in Z'\ \rightarrow ps(C')\ =\ C$,

(b) $\forall r_i \in R, \exists s \in pre\_s(r_i, C) \cup\ post\_s(r_i, C)\},\ \nexists r_i{}' \in R'\ \ \rightarrow ps(r_i{}') = r_i$,

(c) $\forall v_J \in \{r.v \mid \exists r \in R,\ s \in pre\_s(r, C) \cup\ post\_s(r, C)\}, \nexists v_j{}' \in V'\ \rightarrow ps(v_j{}') = v_j$.

It is noted that if an artifact, a task, or a business rule remains unmodified in the specialized model, then $ps(x)\ =\ x$.

We can consider the specialization of an ACP model as the product of individual artifact lifecycle specialization and synchronization specialization. Next, in Section 4.2.1, we discuss the specialization of isolated (non-synchronized) lifecycles. Then, Section 4.2.3 discusses how the inter-behavior of interacting artifacts (via sync rules) is taken into account for the ACP specialization.

4.2.1 Behavioral consistency

We use the notion of behavioral consistency, called *B-consistency* between two artifact-centric process models, that has been proposed in [Yongchareon et al., 2012] to check whether an ACP model (or an artifact) that derives from its base ACP model (or a base artifact) is consistently observable. The *B-consistency* is based on a weak bi-simulation equivalence relation [Bloom, 1995] for creating a behavior-consistent specialized process. By replacing a lifecycle fragment to be abstracted with a *silent* (τ) *action*, a weak bi-simulation can be applied to the abstract lifecycle and its original lifecycle.

Definition 4.2: *(B-consistent, ≃)*. Let $\mathcal{L}_y = (S_y, s_y{}^{init}, \Rightarrow_y)$ and $\mathcal{L}_x = (S_x, s_x{}^{init}, \Rightarrow_x)$ be two lifecycles such that $\mathcal{L}_y$ derives from $\mathcal{L}_x$ where $S_{x\cap y} = S_x \cap S_y$ is a set of states found from both $\mathcal{L}_x$ and $\mathcal{L}_y$. We say that $\mathcal{L}_y$ and $\mathcal{L}_x$ are *B-consistent* (denoted as $\mathcal{L}_y \simeq \mathcal{L}_x$) if the following conditions hold:

$$—\forall s_i, s_j \in S_x, \exists s_i, s_j \in S_y, \exists (s_i, r, g, s_j) \in \Rightarrow_x \\ \rightarrow \forall s_k \in S_y \setminus S_{x\cap y}, s_i \overset{*}{\Rightarrow}_y s_k\ \wedge\ s_k \overset{*}{\Rightarrow}_y s_j \tag{4.1}$$

$$—\forall s_i, s_j \in S_x, \exists s_i, s_j \in S_y, \nexists (s_i, r, g, s_j) \in \Rightarrow_x \\ \rightarrow \forall s_k \in S_y \setminus S_{x\cap y}, \neg(s_i \overset{*}{\Rightarrow}_y s_k\ \wedge\ s_k \overset{*}{\Rightarrow}_y s_j) \tag{4.2}$$

$$—\forall s_i, s_j, s_m \in S_x, \exists s_j \notin S_y, \exists (s_i, r_i, g_i, s_j) \in \Rightarrow_x \\ \rightarrow \exists (s_j, r_j, g_j, s_{m,}) \in \Rightarrow_x, \exists s_k \in S_y \setminus S_{x\cap y}, s_i \overset{*}{\Rightarrow}_y s_k\ \wedge\ s_k \overset{*}{\Rightarrow}_y s_m \tag{4.3}$$

Conditions 4.1 and 4.2 are used to restrict each transition in $\mathcal{L}_x$ to be derived by a sub-lifecycle in $\mathcal{L}_y$. Condition 4.3 is used to constrain the substitution of a state of $\mathcal{L}_x$ with a sub-lifecycle in $\mathcal{L}_y$. Note that we generalize the definition of *B-consistency* for checking any type of lifecycle constructs. In a clear context, we may write $\Pi' \simeq \Pi$ to mean that the lifecycle of an ACP model Π' and that of an ACP model Π are *B-consistent*.

4.2.2 Lifecycle specialization

Based on *ACP specialization* with our focus only on the behavioral perspective, we define *lifecycle specialization* to map a lifecycle of a subtype to its super-type. Note that we generalize its definition for both an artifact class and a process model.

Definition 4.3: *(Lifecycle specialization).* Given a specialized ACP model Π' constructed based on an ACP model Π via a *ACP specialization function* $ps_{\Pi' \to \Pi}$, a lifecycle $\mathcal{L} = (S, s^{init}, \Rightarrow)$, and a lifecycle $\mathcal{L}' = (S', s^{init'}, \Rightarrow')$, a lifecycle specialization relationship between $\mathcal{L}$ and $\mathcal{L}'$ can be defined using a *lifecycle specialization mapping* (total) function $ls_{\mathcal{L}' \to \mathcal{L}} : S' \cup s^{init'} \cup \Rightarrow' \to S \cup s^{init} \cup \Rightarrow \cup \{\varepsilon\}$, where ε is an *empty element*.

The following *B-consistent lifecycle specialization* can then be defined based on the *lifecycle specialization* and *B-consistency*.

Definition 4.4: *(B-consistent specialization, $\simeq_{ls}$).* Given ACP models Π' and Π, let an artifact lifecycle $\mathcal{L}' = (S', s^{init}, \Rightarrow')$ in a Π' be a specialized lifecycle that inherits an artifact lifecycle $\mathcal{L} = (S, s^{init}, \Rightarrow)$ in Π via the use of a *lifecycle specialization* $ls_{\mathcal{L}' \to \mathcal{L}}$. $\mathcal{L}'$ is a *B-consistent specialization* of $\mathcal{L}$, denoted as $\mathcal{L}' \simeq_{ls} \mathcal{L}$, if $\mathcal{L}' \simeq \mathcal{L}$. Correspondingly, $ls_{\mathcal{L}' \to \mathcal{L}}$ can be said *B-consistent*.

Next, we define lifecycle specialization based on a fragment of a lifecycle, namely *L-fragment,* and discuss how an *L-fragment* can be used to construct a *B-consistent* specialized lifecycle, as shown in Theorem 4.1.

Definition 4.5: (L-fragment and AL-fragment). An *L-fragment* $\ell = (S, \Rightarrow, \Rightarrow^{in}, \Rightarrow^{out})$ of *lifecycle* $\mathcal{L}$ can be defined as a sub-lifecycle of $\mathcal{L}$ where,

—$S \subseteq \mathcal{L}.S \setminus \{s^{init}\} \cup S^f$ is a finite set of states where S^f are final states in $\mathcal{L}$,

—$\Rightarrow \subseteq S \times R^{\mathcal{L}} \times G^{\mathcal{L}} \times S \subseteq \mathcal{L}.\Rightarrow$ are transitions in ℓ where $R^{\mathcal{L}}$ are business rules and $G^{\mathcal{L}}$ are guards in $\mathcal{L}$,

—$\Rightarrow^{in} = \mathcal{L}.\Rightarrow \cap ((\mathcal{L}.S \setminus S) \times R^{\mathcal{L}} \times G^{\mathcal{L}} \times S))$ is a finite set of transitions that enter to ℓ,

—$\Rightarrow^{out} = \mathcal{L}.\Rightarrow \cap (S \times R^{\mathcal{L}} \times G^{\mathcal{L}} \times (\mathcal{L}.S \setminus S))$ is a finite set of transitions that exit from ℓ.

An *AL-fragment* can be defined as a special case of an L-fragment where it holds the atomicity property namely SESE which is used for the analysis of program control-flow graphs in the compiler theory [Johnson et al., 1994] and the structure analysis of business process models [Vanhatalo et al., 2007].

Definition 4.6: (Refine, Refined L-fragment). Given a lifecycle specialization $ls_{\mathcal{L}' \to \mathcal{L}}$, we can define $lf(\mathcal{L}' \to \mathcal{L}) = \{\ell_1, \ell_2, \dots, \ell_y\}$ as a set of refined *L-fragments* for *refining* $\mathcal{L}$, where $\ell_i (1 \le i \le y)$ is an L-fragment in $\mathcal{L}'$ such that all the states and transitions of ℓ_i do not exist in $\mathcal{L}$.

Next, we formulate a theorem to help guarantee *B-consistency* of a specialized process model when an L-fragment is used to refines a transition or a state of the base process model.

THEOREM 4.1 (B-CONSISTENT REFINED L-FRAGMENT). Given a specialized artifact lifecycle $\mathcal{L}'$ based on an artifact lifecycle $\mathcal{L}$ and *refined L-fragments* $lf(\mathcal{L}' \to \mathcal{L})$, we have $\mathcal{L}' \simeq_{ls} \mathcal{L}$ if, for every refined L-fragment $\ell_i \in lf(\mathcal{L}' \to \mathcal{L})$,

— ℓ_i is considered as an *AL-fragment* if ℓ_i refines a transition $s_x \Rightarrow_t s_y \in \mathcal{L}.\Rightarrow$ or,

— if ℓ_i refines a state $s \in \mathcal{L}.S$, then for every incoming state $s_x \in \mathcal{L}.S$ that is fired to s and every outgoing $s_y \in \mathcal{L}.S$ that is fired from s, s_y is reachable from s_x via some L-occurrences of ℓ_i.

The proof of THEOREM 4.1 can be found in the appendix.

The first statement of THEOREM 4.1 is for a case where an L-fragment refines a transition of a lifecycle in the base ACP model, the L-fragment must be atomic in order to preserve B-consistency. A transition is naturally atomic by its definition thus the refined fragment is confined to be atomic. On the other hand, the second statement is used for a case where an L-fragment refines a state of a lifecycle in the base ACP model. With the state refinement, it is more complicated as a state may have multiple entry and exit states/transitions associated with it. Therefore, the condition is put in place to ensure that after the refinement, all the exit states associated with exit transitions are reachable from the entry states.

Next, we show in THEOREM 4.2 that the *B-consistent refinement* guarantees the soundness of the specialized lifecycle.

THEOREM 4.2: (SOUND AND B-CONSISTENT REFINEMENT). Given an ACP specialization $ps_{\Pi' \to \Pi}$ based on an artifact refinement and a set of *refined L-fragments* $lf(\mathcal{L}' \to \mathcal{L})$ ($\mathcal{L}'$ can be the lifecycle of any specialized artifact in $\Pi'.Z$). Π' is a *sound B-consistent specialization* of Π if every $\ell_i \in lf(\mathcal{L}' \to \mathcal{L})$ holds all the conditions in THEOREM 4.1 and ℓ_i is *sound*.

The proof of THEOREM 4.2 can be found in the appendix.

The major component of the THEOREM 4.2 is based on the *soundness* (of each of the fragments) that is used to refine a state or a transition of an artifact in the base process model. If all the fragments are *sound,* then the refinement is *sound* and so is the specialized process model.

Next, we define a behavior-consistent specialized artifact and a behavior-consistent specialized process based on the notion of *B-consistent* (lifecycle) specialization.

Definition 4.7: *(Behavior-consistent specialized artifact and process).* Given an ACP specialization $ps_{\Pi' \to \Pi}$, Π' is a *behavior-consistent specialization* of Π if $ls_{\mathcal{L}'_\Pi \to \mathcal{L}_\Pi}$ is *B-consistent*. Similarly, given a specialized artifact $C' \in \Pi'.Z$ based on $ps_{\Pi' \to \Pi}$, C' is a *behavior-consistent specialization* of C if $ls_{\mathcal{L}'_C \to \mathcal{L}_C}$ is *B-consistent*.

4.2.3 Sync specialization

Next, we discuss how changes of artifact synchronization impact the behavior of a specialized process. We can group the methods of specialization of synchronizations (so-called *sync specialization*) into the following three primitive operations as follows.

—*Sync refinement* refers to a method for decomposing a sync rule in a base model into a set of refined sync rules in a specialized model.

—*Sync extension* refers to a method to add the synchronization between a new artifact and an existing artifact with the introduction of a set of exnteded sync rules.

—*Sync reduction* refers to a method for removing synchronization between existing artifacts from a specialized process model.

Table 4.1 shows the overall relations between the three construction methods of ACP specialization introduced in Section 4.1 and the three primitive sync specialization operations. Note that the artifact extension generally requires both sync extension and sync refinement operations. The details of each operation are discussed later in this section.

Table 4.1: Relations between ACP specialization methods and Sync specialization

	Sync extension	Sync refinement	Sync reduction
Artifact refinement		√	
Artifact extension	√	√	
Artifact reduction			√

Next, we discuss synchronization dependencies and the consistency criteria used for behavioral-consistent process specialization. In an essence, the behavior consistency should be preserved between the composed lifecycle of specialized artifacts in a specialized model and their composition in the process model it derives from. We formally define synchronization specialization between lifecycles and then we discuss how the synchronization introduced can be consistently handled w.r.t each of the three sync specialization operations introduced—*refinement, extension, and reduction*.

Definition 4.8: (Sync specialization). Given $\mathcal{L}'_X$ and $\mathcal{L}'_Y$ be artifact lifecycles in an ACP model Π' that are behavior-consistent specializations of artifact lifecycles $\mathcal{L}_X$ and $\mathcal{L}_Y$ in an ACP model Π. We denote $ss_{(\mathcal{L}'_X,\, \mathcal{L}'_\mathrm{Y}) \rightarrow (\mathcal{L}_\mathrm{X}, \mathcal{L}_\mathrm{Y})} : \varphi(\mathcal{L}'_X, \mathcal{L}'_\mathrm{Y}) \rightarrow \varphi(\mathcal{L}_\mathrm{X}, \mathcal{L}_\mathrm{Y}) \cup \{\varepsilon\}$ for *a sync specialization (total) function* that projects a sync rule of both $\mathcal{L}'_X$ and $\mathcal{L}'_\mathrm{Y}$ onto its corresponding sync rule of $\mathcal{L}_X$ and $\mathcal{L}_\mathrm{Y}$ or an empty element ε, where

$\varphi(\ell_x, \ell_y) = \{r \in \Pi.R \mid \exists(s_i, r, g, s_j) \in \ell_x.\Rightarrow \wedge\ \exists(s_m, r, g, s_n) \in \ell_y.\Rightarrow\}$ are *sync rules* for ℓ_x and ℓ_y.

A sync rule is considered unchanged if, for every sync rule $r \in \varphi(\mathcal{L}'_X, \mathcal{L}'_\mathrm{Y}), r \in \varphi(\mathcal{L}_\mathrm{X}, \mathcal{L}_\mathrm{Y}), ss_{(\mathcal{L}'_\mathrm{X},\, \mathcal{L}'_\mathrm{Y}) \rightarrow (\mathcal{L}_\mathrm{X}, \mathcal{L}_\mathrm{Y})}$.

4.2.4 Synchronized Fragments

In this section, we define a *synchronized region* to model synchronization dependencies between lifecycles using the *S-region* and *SL-fragment* notions proposed in [Yongchareon et al., 2015]. An *S-region* is used to represent a set of synchronized L-fragments and the atomicity property of an S-region can be verified via checking the composability and boundedness of the SL-fragments contained in the S-region. This can be done adapting the proposed fragmental composition technique initially proposed in [Yongchareon et al., 2015]. Technically, we need to determine if SL-fragments of an S-region can form an atomic S-region, called *AS-region*.

Definition 4.9: *(SL-fragment and S-region).* Let Π be an ACP model, we can define a S-region $\omega = (\Gamma, R^{sync})$ in Π where,

—$\Gamma = \{\ell^{C_1}, \ell^{C_2}, \ldots, \ell^{C_x}\}$ are L-fragments in synchronization, where $\ell^{C_i} \in \Gamma (1 \leq i \leq x)$ is an *SL-fragment* of $\mathcal{L}_{C_i}$ $(C_i \in \Pi.Z)$,

—$R^{sync} \subseteq \Pi.R$ are sync rules exclusively defined for the synchronization of transitions of some L-fragments in Γ.

For example, In Figure 4.2 (a), An S-region ω_a has an SL-fragment l_1 synchronized with an SL-fragment l_2 via two sync rules r_1 and r_2. In Figure 4.2 (b), An S-region ω_b consists of SL-fragments l_3 and l_4, and sync rules $R^{sync} = \{r_1, r_2, r_3\}$. Note that a sync rule r_4 is not in ω_b since it is not part of l_3 and l_4.

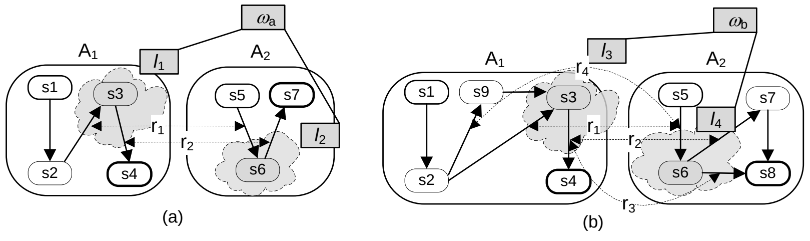

Figure 4.2. An example of S-regions and SL-fragments.

LEMMA 4.1 (ATOMIC COMPOSITION OF SL-FRAGMENTS). Given an ACP model Π, let an S-region $\omega = (\Gamma, R^{sync})$ and $Z^\Gamma \subseteq \Pi.Z$ be artifacts such that there exists an L-fragment for each of Z^Γ in Γ. The composed lifecycle of all the SL-fragments in Γ is *AL-fragment* and *sound* if, for each $\ell^{C_i} \in \Gamma$, the following statements hold:

(1) ℓ^{C_i} is an AL-fragment,

(2) $\forall \mathcal{L}_{C_j} (C_j \in \Pi.\mathrm{Z} \setminus Z^\Gamma), \varphi(\ell^{C_i}, \mathcal{L}_{C_j}) = \emptyset$,

(3) $\forall \ell^{C_\mathrm{x}}, \ell^{C_\mathrm{y}} \in \Gamma, \forall r \in \varphi\left(\mathcal{L}_{C_x}, \mathcal{L}_{C_y}\right), \exists s \in \mathcal{L}_{C_i}.S, \exists s_s \in \ell^{C_i}.S, s_s \overset{r}{\Rightarrow} s \in \mathcal{L}_{C_i}.\Rightarrow \rightarrow s_s \overset{r}{\Rightarrow} s \in \ell^{C_i}.\Rightarrow \wedge\ r \in R^{sync}$,

(4)$\forall \ell^{C_x}, \ell^{C_y} \in \Gamma, \forall r \in \varphi\left(\mathcal{L}_{C_x}, \mathcal{L}_{C_y}\right), \exists s \in \mathcal{L}_{C_i}.S, \exists s_t \in \ell^{C_i}.S,\ s \overset{r}{\Rightarrow} s_t \in \mathcal{L}_{C_i}.\Rightarrow \rightarrow s \overset{r}{\Rightarrow} s_t \in \ell^{C_i}.\Rightarrow \wedge\ r \in R^{sync}$.

The proof of LEMMA 4.1 can be found in the appendix.

Note that Conditions (2), (3), and (4) in LEMMA 4.1 limit any two to-be-composed SL-fragments in Γ to contain all the transitions and the necessary sync rules required for synchronization.

Definition 4.10: (ASL-fragment and AS-region). Given an *S-region* $\omega = (\Gamma, R^{sync})$, if ω holds Lemma 4.1, then ω is called an *atomic S-region* (*AS-region*) and each L-fragment in Γ is called an *ASL-fragment*.

4.2.5 Sync extension

For a sync extension operation, if a new (synchronized) artifact is added (or extended) into a specialized process model, the consistency must be ensured not undermined by the behavior of the added artifact. We discuss how we can check the composability of the extended artifact and an L-fragment the artifact synchronizes with. We show that LEMMA 4.2 can be used to preserve *B-consistency* of a specialized model with the added artifact and its base process model.

Next, we define an *ex-lifecycle* for an artifact that can be entirely synchronized within an L-fragment of an existing artifact. A lifecycle $\mathcal{L}_{C_j}$ is fully-embedded in a lifecycle $\mathcal{L}_{C_i}$ if all the scope of the synchronization between some L-fragment $\ell^{C_i} \prec \mathcal{L}_{C_i}$ and $\mathcal{L}_{C_j}$ are within the boundary of ℓ^{C_i}.

Definition 4.11: *(Ex-lifecycle, $\sqsupset$).* Given $\mathcal{L}'_X$ be a lifecycle in an ACP model Π' that specializes a lifecycle $\mathcal{L}_X$ in an ACP model Π such that $\Pi' \simeq \Pi$, a lifecycle $\mathcal{L}'_Y$ of an extended artifact Y' in Π' synchronizes $\mathcal{L}'_X$ with a set of lifecycle specializations $\{ls_{\mathcal{L}'_X \to \mathcal{L}_X}, ls_{\mathcal{L}'_Y \to \varepsilon}\}$, and a sync specialization $ss_{(\mathcal{L}'_X, \mathcal{L}'_Y) \to (\mathcal{L}_X, \mathcal{L}_Y)}$ such that $ps(Y') = \varepsilon \wedge \varphi(\mathcal{L}'_X, \mathcal{L}'_Y) \neq \emptyset$ and for every sync rule $r' \in \varphi(\mathcal{L}'_X, \mathcal{L}'_Y), ss(r') = \varepsilon$, we have $\mathcal{L}'_Y$ as an *ex-lifecycle* of $\mathcal{L}'_X$ if $\mathcal{L}'_Y$ has its entire lifecycle synchronized with some transitions in a refined L-fragment ℓ_i where $\ell_i \in lf(\mathcal{L}'_X \to \mathcal{L}_X)$, also denoted as $\ell_i \sqsupset \mathcal{L}'_Y$, i.e.,

—for every entry transition and exit transition $\Rightarrow_t$ in $\mathcal{L}'_Y$, $\Rightarrow_t$ synchronizes with some transition in ℓ_i; and,

—there does not exist sync rule $r_z' \in \varphi(\mathcal{L}'_X, \mathcal{L}'_Y)$ such that r_z' synchronizes any transition that exists in $\mathcal{L}'_Y$ with any transition that does not exist in ℓ_i.

The *ex-lifecycle* notion can be used to ensure that every L-occurrence of an extended artifact can reach the final state within some refined (and synchronized) L-fragment. In addition, we observe that: if an extended artifact terminates after all existing artifact(s) have terminated, then the composed lifecycle (of these all artifacts) contains some L-occurrences that lead to open-ended termination inconsistency (when comparing to the composition of existing artifacts). It is undesirable that such a case occurs in a specialized process since the specialized process cannot terminate itself at the same state as its base process does. Therefore, we do not consider this case in our study when checking its behavior consistency and we assume that all extended artifacts must terminate before all existing artifacts terminate. Figure 4.3 shows an example of the case where the open-ended termination inconsistency occurs in the process (b) that specializes its base process (a).

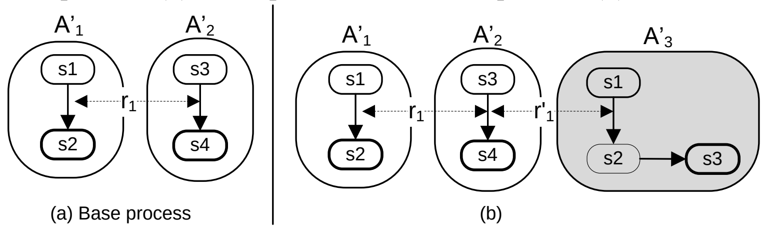

Figure 4.3: Open-ended termination inconsistency.

LEMMA 4.2. Let $\ell \in lf(\mathcal{L}'_X \to \mathcal{L}_Y)$ be a refined L-fragment that synchronizes with an extended lifecycle $\mathcal{L}'_Y$. We have that $\mathcal{L}'_Y \otimes \ell$ is *B-consistent* with ℓ if $\ell \sqsupset \mathcal{L}'_Y$ if ℓ is an *AL-fragment*.

The proof of LEMMA 4.2 can be found in the appendix.

From Lemma 4.2, if $\ell \sqsupset \mathcal{L}'_Y$ and ℓ is an ASL-fragment, then $\mathcal{L}'_Y$ is also considered as an ASL-fragment, and therefore we can build an AS-region for $\mathcal{L}'_Y$ and ℓ. If an AS-region can be built containing a refined L-fragment of an artifact and a lifecycle of a different artifact, then we have the whole lifecycle fully embedded in the refined L-fragment. Therefore, we can ensure that the *B-consistency* between the refined L-fragment and the embedded lifecycle as the AS-region guarantees the atomicity of the composition of such two, i.e. if an AS-region cannot be formed then the *B-consistency* is violated.

4.2.6 Sync refinement

4.2.6.1 Refinement of synchronization between two existing artifacts

There are two specialization patterns of the synchronization between two existing artifacts. First is the case where one artifact is refined while the other one is unchanged, and the second case is when both artifacts are refined.

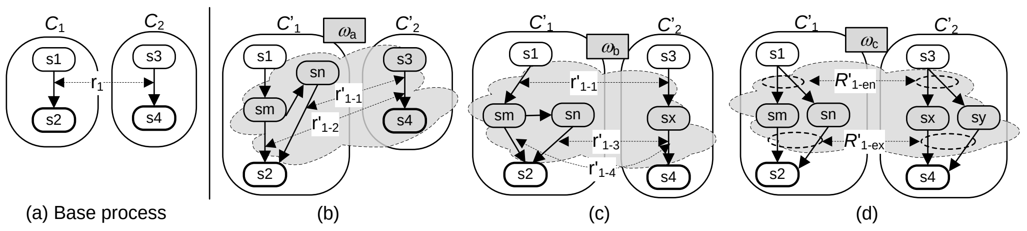

Figure 4.4: Sync refinements of existing artifacts.

Example 4.2. Figure 4.4 (b) illustrates an example of the sync refinement of a single artifact. We can see that a sync rule r_1 in Figure 4.4 (a) is refined to r'_{1-1} and r'_{1-2} with a refined SL-fragment in an artifact C'_1 containing states s_m and s_n. While Figure 4.4 (c) and (d) demonstrate the sync refinement work for artifacts C'_1 and C'_2.

Next show how we apply the AS-region notion to help verify the *B-consistency* condition.

Definition 4.12: *(Refined S-region).* Let $\mathcal{L}'_X$ and $\mathcal{L}'_Y$ be specialized artifact lifecycles in an ACP model Π' and $\mathcal{L}'_X$ and $\mathcal{L}'_Y$ be *B-consistent* with artifact lifecycles $\mathcal{L}_X$ and $\mathcal{L}_Y$ in an ACP model Π, respectively, with lifecycle specializations $\{ls_{\mathcal{L}'_X \rightarrow \mathcal{L}_X}, ls_{\mathcal{L}'_Y \rightarrow \mathcal{L}_Y}\}$ and a sync specialization $ss_{(\mathcal{L}'_X, \mathcal{L}'_Y) \rightarrow (\mathcal{L}_X, \mathcal{L}_Y)}$, such that for every sync rule $r' \in \varphi(\mathcal{L}'_X, \mathcal{L}'_Y), ss(r') \neq \varepsilon$. Let a sync rule $r \in \varphi(\mathcal{L}_X, \mathcal{L}_Y)$ be used to synchronize a transition $\Rightarrow_i$ in $\mathcal{L}_X$ with a transition $\Rightarrow_j$ in $\mathcal{L}_Y$. A *refined S-region* can be defined as follows.

—If an L-fragment $\ell_m \in lf(\mathcal{L}'_Y \rightarrow \mathcal{L}_Y)$ refining $\Rightarrow_j$ is synchronized with a transition $\Rightarrow_t \in \mathcal{L}'_X.\Rightarrow$, then we define a *refined S-region* $\omega = (\{\ell_m, \ell_n\}, \varphi(\ell_m, \ell_n))$ where an (unrefined) L-fragment ℓ_n contains a single transition $\Rightarrow_t$ and its source state and target state, such that for every sync rule $r' \in \varphi(\ell_m, \ell_n), ss(r') = r$.

—If an L-fragment $\ell_m \in lf(\mathcal{L}'_Y \rightarrow \mathcal{L}_Y)$ refining $\Rightarrow_j$ is synchronized with an L-fragment $\ell_n \in lf(\mathcal{L}'_X \rightarrow \mathcal{L}_Y)$ refining $\Rightarrow_i$, then we define a *refined S-region* $\omega = (\{\ell_m, \ell_n\}, \varphi(\ell_m, \ell_n))$ such that for every sync rule $r' \in \varphi(\ell_m, \ell_n), ss(r') = r$.

Example 4.3. In Figure 4.4 (b), an S-region ω_a is defined as a refined S-region of its base process model in Figure 4.4 (a) based on the first statement of *Definition 4.12.* The refined S-region consists of the shaded L-fragment of C'_1, the L-fragment that is expanded from a transition $(s_3 \Rightarrow s_4)$ of C'_2, and a set of refined sync rules $\{r'_{1-1}, r'_{1-2}\}$. In Figure 4.4 (c) and (d), both refined S-region ω_b and ω_c are defined based on the second statement of *Definition 4.12.*

LEMMA 4.3. Given two specialized artifact lifecycles $\mathcal{L}'_X$ and $\mathcal{L}'_Y$ that are *B-consistent* with artifact lifecycles $\mathcal{L}_X$ and $\mathcal{L}_Y$, a *refined S-region* $\omega = (\{\ell_m, \ell_n\}, \varphi(\ell_m, \ell_n))$ where ℓ_m refines a transition $\Rightarrow_i$ in $\mathcal{L}_X$ and ℓ_n refines a transition $\Rightarrow_j$ in $\mathcal{L}_Y$, if ℓ_m and ℓ_n are ASL-fragments, then $\ell_m \otimes \ell_n$ is *B-consistent* with $\Rightarrow_i \otimes \Rightarrow_j$.

The proof of LEMMA 4.3 can be found in the appendix.

4.2.6.2 Refinement of synchronization between one existing artifact and one extended artifact

We can apply the notion of ex-lifecycle from the sync extension method to the refinement of synchronization between an existing artifact and an extended artifact as follows.

LEMMA 4.4. Given $\mathcal{L}'_X$ and $\mathcal{L}'_Y$ be artifact lifecycles that are *B-consistent* with artifact lifecycles $\mathcal{L}_X$ and $\mathcal{L}_Y$, $\omega = (\{\ell_m, \ell_n\}, \varphi(\ell_m, \ell_n))$ be a refined *S-region* where ASL-fragment ℓ_m refines a transition $\Rightarrow_i$ of $\mathcal{L}_X$ and ASL-fragments ℓ_n refines a transition $\Rightarrow_j$ of $\mathcal{L}_Y$, and $\mathcal{L}'_Z$ be an extended lifecycle that synchronizes with ω, we can say that $\ell_m \otimes \ell_n \otimes \mathcal{L}'_Z$ is *B-consistent* with $\Rightarrow_i \otimes \Rightarrow_j$ if,

$$\ell_m \sqsupset \mathcal{L}'_Z \wedge \ell_n \sqsupset \mathcal{L}'_Z.$$

The proof of LEMMA 4.4 can be found in the appendix.

4.2.6.3 Refinement of synchronization between one existing artifact and a set of extended artifacts

We reuse the sync extension method and the notion of *ex-lifecycle* for the synchronization of multiple extended artifacts. Suppose that we have an existing artifact C_3 and extended artifacts C_1 and C_2. If C_1 has it lifecycle completely synchronized and encapsulated within the lifecycle of C_2 and $C_2 \sqsupset C_3$, then we can transitively consider C_1 as an *ex-lifecycle* of C_3. Next, we define a *transitive ex-lifecycle* to capture the transitivity of ex-lifecycles.

Definition 4.13: *(Transitive ex-lifecycle, $\sqsupset^+$)*. Give $\mathcal{L}'_X$ be a specialized artifact lifecycle that is *B-consistent* with an artifact lifecycle $\mathcal{L}_X$, $\mathcal{L}'_Y$ and $\mathcal{L}'_Z$ be lifecycles of two extended artifacts Y' and Z', respectively with a set of lifecycle specializations $\{ls_{\mathcal{L}'_X \to \mathcal{L}_X}, ls_{\mathcal{L}'_Y \to \varepsilon}, ls_{\mathcal{L}'_Z \to \varepsilon}\}$ and a set of sync specializations $\{ss_{(\mathcal{L}'_X, \mathcal{L}'_Y) \to (\mathcal{L}_X, \mathcal{L}_Y)}, ss_{(\mathcal{L}'_Y, \mathcal{L}'_Z) \to (\mathcal{L}_Y, \mathcal{L}_Z)}\}$, $\mathcal{L}'_Z$ is a *transitive ex-lifecycle* of $\mathcal{L}'_X$ if $\mathcal{L}'_Z$ is an ex-lifecycle of $\mathcal{L}'_Y$ and $\mathcal{L}'_Y$ is an ex-lifecycle of $\mathcal{L}'_X$ such that

—$ps\left(Y'\right) = \varepsilon \wedge \varphi(\mathcal{L}'_X, \mathcal{L}'_Y) \neq \emptyset$ and $ps\left(Z'\right) = \varepsilon \wedge \varphi(\mathcal{L}'_Y, \mathcal{L}'_Z) \neq \emptyset$,

—$\forall r' \in \varphi(\mathcal{L}'_X, \mathcal{L}'_Y) \cup \varphi(\mathcal{L}'_Y, \mathcal{L}'_Z), ss(r') = \varepsilon$.

Correspondingly, we have that a refined L-fragment $\ell_i \sqsupset^+ \mathcal{L}'_Z$ where $\ell_i \in lf(\mathcal{L}'_X \to \mathcal{L}_X)$ if $\ell_i \sqsupset \mathcal{L}'_Y$ and $\mathcal{L}'_Z \sqsupset \mathcal{L}'_Y$.

Next, Lemma 4.5 shows how Lemma 4.4 can be extended to consider the transitivity of ex-lifecycles for the *B-consistency* checking.

LEMMA 4.5. Let two artifact lifecycles $\mathcal{L}'_X$ and $\mathcal{L}'_Y$ be *B-consistent* with artifact lifecycles $\mathcal{L}_X$ and $\mathcal{L}_Y$, and let a refined *S-region* $\omega = (\{\ell_m, \ell_n\}, \varphi(\ell_m, \ell_n))$ where ASL-fragments ℓ_m refines a transition $\Rightarrow_i$ in $\mathcal{L}_X$ and ASL-fragments ℓ_n refines a transition $\Rightarrow_j$ in $\mathcal{L}_Y$. Given $\mathcal{L}'_X$ be an extended lifecycle that synchronizes with ℓ_m or ℓ_n in ω and Z^{ex} be a set of extended lifecycles that synchronize with $\mathcal{L}'_X$, the lifecycle composition of ℓ_m, ℓ_n, $\mathcal{L}'_X$ and every artifact in Z^{ex} is *B-consistent* with $\Rightarrow_i \otimes \Rightarrow_j$ if,

$$\forall \mathcal{L}'_{C_i}\{C'_i \in Z^{ex}\}, \ell_m \sqsupset^+ \mathcal{L}'_{C_i} \wedge \ell_n \sqsupset^+ \mathcal{L}'_{C_i}.$$

The proof of LEMMA 4.5 can be found in the appendix.

4.2.7 Sync reduction

Based on the *artifact reduction* method we can remove an artifact in a process that specializes its base process. By doing this, all synchronizations between such removed artifact and all other related artifacts that synchronize with it should also be removed. The effect of the deletion of one artifact should propagate to the removal of a part of a lifecycle of another artifact(s) that the deleted artifact is synchronized with. The deleted artifact must be guaranteed not violating *B-consistency*. Here, we discuss how *ASL-fragments* and *AS-regions* can be reused to define a *reducible lifecycle* of an artifact.

Definition 4.14: *(Reducible L-fragment).* Given $\mathcal{L}'_X$ be an artifact lifecycle in an ACP model Π' that specializes an artifact lifecycle $\mathcal{L}_X$ in a base ACP model Π and $\mathcal{L}'_X$ is *B-consistent* with $\mathcal{L}_X$, assume an artifact lifecycle $\mathcal{L}_Y$ of $Y \in \Pi.Z$ that is synchronized with $\mathcal{L}_X$ in Π be removed from Π' with a set of lifecycle specializations $\{ls_{\mathcal{L}'_X \to \mathcal{L}_X}, ls_{\mathcal{L}'_Y \to \mathcal{L}_Y}\}$ and a sync specialization $ss_{(\mathcal{L}'_X, \mathcal{L}'_Y) \to (\mathcal{L}_X, \mathcal{L}_Y)}$ such that $\nexists C' \in \Pi'.Z \to ps(C') = Y$. If an L-fragment ℓ in $\mathcal{L}_X$ and s $\mathcal{L}_Y$ are ASL-fragments and ℓ is an *AS-region*, then ℓ can be considered a *reducible L-fragment* of $\mathcal{L}_X$ for $\mathcal{L}_Y$.

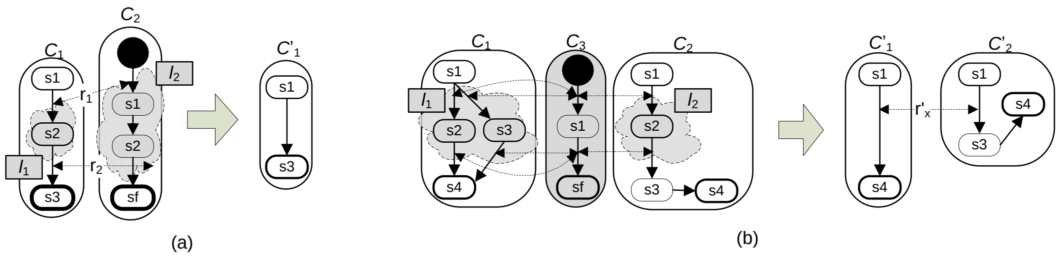

Figure 4.5: Examples of reduced lifecycles based on sync reduction.

Example 4.4. In Figure 4.5 (a), an artifact A_2 is to be removed in a specialized process model. We can see that the L-fragment ℓ_1 of an artifact C_1 synchronizes with an artifact C_2 and both ℓ_1 and C_2 are ASL-fragments from which an AS-region can be constructed. Therefore, we have ℓ_1 as a reducible L-fragment of C_1 for C_2. Similarly, in Figure 4.5 (b), an L-fragment ℓ_2 of C_1 and an L-fragment ℓ_3 of C_2 are synchronized with C_3. So, we can construct one AS-region consisting of ℓ_2 and C_3 as well as another one AS-region consisting of ℓ_3 and C_3. As such, we have ℓ_2 and ℓ_3 as reducible L-fragments of C_1 and C_2, respectively, for C_3.

Definition 4.15: *(Reduced transition).* Given $\mathcal{L}_Y$ be an artifact lifecycle that is removed from an ACP model Π' and L^{ex} be a set of artifact lifecycles that synchronize with $\mathcal{L}_Y$, if, for every artifact lifecycle $\mathcal{L}_{C_i}$ in L^{ex}, there exists a *reducible L-fragment* ℓ^{C_i} of $\mathcal{L}_{C_i}$ for $\mathcal{L}_Y$, then ℓ^{C_i} can be reduced to a *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 exit state of ℓ^{C_i}.

Example 4.5. Based on Example 4.4, in Figure 4.5 (a), an ASL-fragment ℓ_1 of C_1 can be reduced to a reduced transition $s_1 \Rightarrow_{\ell_1} s_3$ in C'_1. Similarly, in Figure 4.5 (b), ASL-fragments ℓ_2 of C_1 and ℓ_3 of C_2 can be reduced to reduced transitions $s_1 \Rightarrow_{\ell_2} s_4$ in C'_1 and $s_1 \Rightarrow_{\ell_3} s_3$ in C'_2, respectively.

Importantly, once an artifact is removed and an effected lifecycle is reduced, all sync rules associated with that lifecycle should also be either reduced or removed.

Definition 4.16: *(Reduced sync rule).* Based on Definition 4.15, given a set of reducible L-fragments L^{re} for a removed artifact $\mathcal{L}_Y$, for every reducible L-fragment ℓ_m in L^{re}, for every sync rule $r \in \varphi(\ell_m, \mathcal{L}_Y)$, r can be:

—*removed* if ℓ_m is synchronized with only $\mathcal{L}_Y$ and ℓ_m is not synchronized with another L-fragment in L^{re}; or,

—*reduced* into a reduced sync rule r_ℓ where r_ℓ is used to synchronize a transition $\Rightarrow_{\ell_m}$ with a transition $\Rightarrow_{\ell_n}$ if there exists ℓ_n in L^{re} such that ℓ_m is synchronized with ℓ_n.

Example 4.6. In Figure 4.5 (a), an artifact C_2 is removed in a specialized process model. The result of removal propagates to the reduction of an L-fragment containing state s_2 with its entering and exiting transitions that are synchronized with C_2. We can see that the L-fragment can be reduced to a single transition ($s_1 \Rightarrow_{\ell_1} s_3$) in C'_1. All the sync rules that are used by C_1 and C_2 can also be removed based on the first condition of Definition 4.16.

Example 4.7. In Figure 4.5 (b), an artifact C_3 is removed in a specialized process model leaving only artifacts C'_1 and C'_2 where the affected ASL-fragments of both artifacts are reduced into reduced transitions. All synchronizations among C_1, C_2, and C_3 are also reduced to a reduced sync rule r'_x based on the second condition of Definition 4.16. The reduced sync rule is then used to synchronize between a reduced transition resulting from the reduction of an L-fragment in C'_1 and a reduced transition resulting from the reduction of an L-fragment in C'_2.

Next, we consider a possible propagation of a result from the removal of an artifact. If an artifact is removed, then the sync reduction should be applied to all related lifecycles that the artifact is synchronized with, including a synchronization that does not directly occur from the removed artifact.

Definition 4.17: *(Propagated sync reduction).* Based on Definition 4.15, given a set of reducible L-fragments L^{re} for a removed artifact $\mathcal{L}_Y$, for every reducible L-fragment ℓ_m in L^{re}, if there exists an L-fragment ℓ_n in $\mathcal{L}_X$ of an artifact X in a base ACP model such that ℓ_n is an ASL-fragment, then ℓ_n is a reducible L-fragment of $\mathcal{L}_X$ for $\mathcal{L}_Y$. We say that the effect of the removal of $\mathcal{L}_Y$ propagates to the reduction of ℓ_n via ℓ_m.

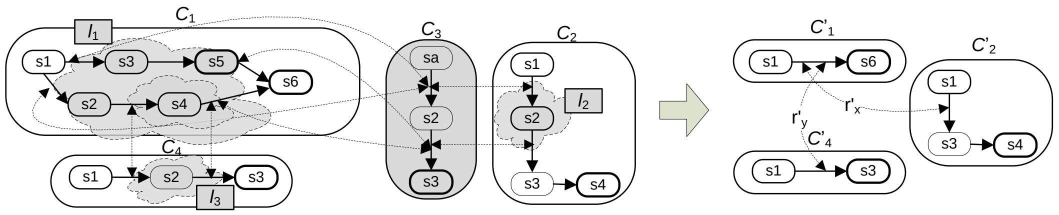

Figure 4.6: Example of propagated sync reduction.

Example 4.8. In Figure 4.6, an artifact C_3 is removed in the specialized process model. We can see that both L-fragments ℓ_1 of an artifact C_1 and ℓ_2 of an artifact C_2 are reducible L-fragments, and ℓ_1 has its sub L-fragment (containing state s_4) synchronized with an L-fragment ℓ_3 of C_4. The sub L-fragment in C_1 and ℓ_3 can be considered as ASL-fragments and can be formed as an AS-region. The removal of an artifact C_3 not only causes the reduction of ℓ_2 but its effect also propagates to cause the reduction of ℓ_3. Therefore, we have ℓ_3 reduced into a reduced transition $s_1 \Rightarrow_{\ell_3} s_3$ in C'_4 and all sync rules defined in ℓ_3 reduced to a reduced sync rule r'_y for the synchronization between C'_1 and C'_4.

LEMMA 4.6. Let an artifact lifecycle $\mathcal{L}_Y$ that is synchronized with artifact lifecycle $\mathcal{L}_X$ be removed. Let L^{re} be the set of all possible reducible L-fragments for $\mathcal{L}_Y$ and T^{re} be a set of reduced transitions, then the lifecycle composition of all transitions in T^{re} is *B-consistent* with the lifecycle composition of all L-fragments in L^{re} and $\mathcal{L}_Y$.

The proof of LEMMA 4.6 can be found in the appendix.

4.2.8 Sync specialization consistency

This section discusses a complete sync consistency (S-consistency) property of ACP model by taking the sync specialization into account.

Definition 4.18: *(S-consistent).* Given Π' be an ACP model that specializes an ACP model Π with $ps_{\Pi' \to \Pi}$ and $ss_{(\mathcal{L}'_X, \mathcal{L}'_Y) \to (\mathcal{L}_X, \mathcal{L}_Y)}$, $ss_{(\mathcal{L}'_X, \mathcal{L}'_Y) \to (\mathcal{L}_X, \mathcal{L}_Y)}$ is said to be *S-consistent* if,

—LEMMA 4.2 holds for the sync extension based on Definition 4.11,

—LEMMA 4.3 and LEMMA 4.5 hold for the sync refinement based on Definition 4.12 and Definition 4.13, respectively,

—LEMMA 4.6 holds for the sync reduction based on Definition 4.14.

Next, we take both the lifecycle specialization and the sync specialization into account to verify whether a specialized ACP model is *B-consistent* with its base ACP model.

THEOREM 4.3. Given Π' be an ACP model that specializes an ACP model Π with $ps_{\Pi' \to \Pi}$, Π' is a *behavior-consistent specialization* of Π if,

—$\forall C'_i \in \Pi'.Z, ps(C'_i) = C_i \in \Pi.Z, ls_{\mathcal{L}'_{C_i} \to \mathcal{L}_{C_i}}$ is *B-consistent*,

—$\forall C'_x \in \Pi'.Z, \forall C'_y \in \Pi'.Z, ss_{(\mathcal{L}'_{C_x}, \mathcal{L}'_{C_y}) \to (\mathcal{L}_{C_x}, \mathcal{L}_{C_y})}$ is *S-consistent*.

The proof of THEOREM 4.3 can be found in the appendix.

5. EXAMPLE REVISITED

In this section, we illustrate how our business process specialization approach can be applied to the reuse of purchasing processes defined by buyers and sellers (or suppliers). We use our motivating business process introduced in Section 2 as a generic process to start with. Now, in the response to the high growth of sales and the organizational changes of both companies, both parties set new requirements for the purchasing processes while keeping their original processes for the purpose of historical analysis and reporting. Based on the new requirements, both the buyers and the supplier must design a new purchasing process based on the extension of their existing process. Here, we describe the requirements in detail. First, the buyers want to have their purchasing quotes approved before submitting a purchase order to the supplier; therefore, an additional *quote* artifact is needed to incorporate into the processes. Second, the supplier needs a *picking list* document to support its internal inventory operations and a *shipping list* for the delivery. In addition, the supplier also allows partial payments for wholesale buyers. Finally, they all agreed with their new process model (namely, *Offline ordering* process) that is a specialization of the generic purchasing process, as illustrated in Figure 5.1. Gray-shaded artifacts and states (with corresponding business rules) are used for the refinement and extension on the generic purchasing process as needed by the requirements.

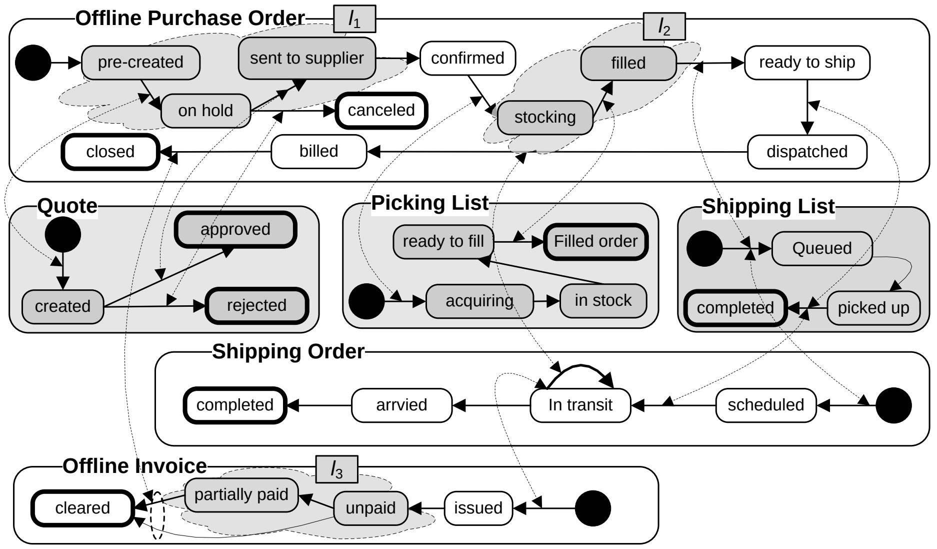

Figure 5.1: Offline ordering process.

In Figure 5.1, we can see that Offline Purchase Order and Offline Invoice specialize Purchase Order and Invoice, respectively, in the generic purchasing process (cf. Figure 2.1). The specialization of these two artifacts is achieved by applying the artifact refinement method (represented using atomic L-fragments ℓ_1 and ℓ_2 on Offline Purchase Order and refined L-fragment ℓ_3 on Offline Invoice). We can see that Shipping List, Picking List, and Quote are introduced and extended to the offline purchasing process in which they can be achieved using the artifact extension method. The Picking List and Quote artifacts synchronize with Offline Purchase Order within AL-fragments ℓ_1 and ℓ_2, respectively. We apply sync extension to Shipping List and Offline Purchase Order, which are ex-lifecycles of Offline Purchase Order. We can see that Shipping Order specializes its base class without any lifecycle refinement; however, it defers its initial synchronization with Offline Purchase Order (from after the confirmed state to after the filled state). This is also like the case of the synchronization on Invoice (with a refined L-fragment). In this example, every lifecycle specialization is B-consistent, and every sync specialization is S-consistent. If we compose all artifacts in the offline purchasing process and compare with the lifecycle composition of every artifact in the generic purchasing process, then we will have the lifecycle of the offline purchasing process covering the lifecycle of the generic purchasing process. Thus, the *B-consistency* is preserved between the lifecycle of the former and the lifecycle of the latter, i.e., the former is a behavior-consistent specialization of the latter.

We have implemented a tool, namely *Artifact-M*, for automatic *B-consistent* checking of specialized process models based on a given base process model. The Artifact-M system requires two inputs files: Artifact-centric process LTS definition file and Process view definition file. The process view definition includes a set of specialized process models to be validated. Artifact-M and the process example used in this manuscript are publicly available for download at https://sites.google.com/site/maxsirayongchareon/artifact-m/acp-i. Process modelers can use this tool to help create specialized ACP process models from a base process model and check if the specialized models are consistent with the base model.

6. RELATED WORK

In the context of business processes, there are various methods in process modeling and design that focus on reuse, e.g., workflow inheritance, reuse of reference models, design by selection and patterns. Although our work in this article is based on the inheritance (or specialization) concept, we acknowledge other related work that aims at supporting the reuse of business processes. Section 6.1, 6.2, and 6.3, discuss existing works based on workflow inheritance, configurable process models, and design by selection and patterns, respectively.

6.1 Inheritance of workflows

Business process design and modeling in today face some problems dealing with changes and process expansions to capture new business requirements in a systematic and rigid manner. Inheritance is considered as one of the key reuse mechanisms in an object-oriented design approach that allows for the definition of a subclass inherits the features of a specific superclass. A precise definition of inheritance (or specialization) promises to be as useful in process modeling likewise in object modeling as it can help organizations to better understand, maintain, and reuse process models [Wyner and Lee, 2002]. Particularly, inheritance concepts are useful to check whether a new workflow process inherits some desirable properties of an existing workflow process [van Der Aalst and Basten, 2002].

Initially, van Der Aalst and Basten [1997] introduced four notions of lifecycle inheritance based on Petri nets [Reisig, 1985] with the use of branching bi-simulation [van Glabbeek and Weijland, 1996] as an equivalence notion. Their inheritances are projection inheritance, protocol inheritance, protocol/projection inheritance, and lifecycle inheritance. *Projection inheritance* is defined based on abstraction. The behavior regarding tasks that exist in both workflow nets is an observable behavior. Added tasks in the inherited workflow can be executed but are not observable, while *protocol inheritance* is defined based on encapsulation. *Life-cycle inheritance* is either a protocol and/or projection inheritance.

Similarly, the behavior-consistent inheritance criteria for object life cycles have been studied by Schrefl and Stumptner [2002] based on the three operations similar to our three proposed specialization methods. They proposed a set of rules for validating behavioral consistency between the lifecycles of two objects. Their work is based on the idea of inheritance proposed by [Basten and van der Aalst, 2001]. Both Harel and Kupferman [2002] and [Van Der Aalst and Basten, 2002] similarly studied the nature of object-oriented system design and a notion of behavioral inheritance for classes should be considered to preserve trace inclusion or simulation.

Furthermore, van Der Aalst and Basten [2001, 2002] proposed to use their inheritance notions to address problems to support not only the reuse (customization) of workflow process but also the management of changes. They proposed *inheritance-preserving transformation rules* (*transfer rules*) to tackle four problems of workflow processes: dynamic changes, management information, inter-organizational interface agreements, and customizing business processes.

Here, we focus on the customization of workflow process and management information.

—*Management information*. Caused by the change of process, multiple variants of the same process can be expected. The number of variants is limited in evolutional changes. However, ad-hoc changes may cause the number of variants up to the order of magnitude of the number of cases. In [van Der Aalst and Basten, 2002], an aggregated view of the work in progress can be achieved based on transfer rules to support the management of a workflow process.

—*Customizing business processes*. Given two business processes, what is the difference between those processes and how much does it cost to customize a process such that it coincides with the other. This requires a delta analysis by deciding where both processes agree on, i.e., to determine the Great Common Devisor (GCD), which can be tackled by the lifecycle inheritance notion. If a set of workflow processes is related under inheritance relationships, it is not difficult to find the GCD given that the inheritance relation defines a partial order (i.e., inheritance relation is reflexive, anti-symmetric, and transitive [Basten and van Der Aalst, 2001]).

Wyner and Lee, [2002, 2003] argued that even though object-oriented analysis and design methodologies take full advantage of object specialization hierarchy, a hierarchy of process specialization is not supported in major process representations, such as state diagrams, data flow diagrams, and UML representations. It is an implicit assumption that a process can be specialized by treating it as just another object. They proposed an approach in the form of a set of transformations to transform a process description into a specialization, which is represented by a state diagram and a data flow diagram. Their approach can be used as a method for categorizing and analyzing processes. Their process specialization definition for a state diagram is compatible with the traditional notion of specialization previously discussed. However, as they considered an entire process as a separate object, therefore dependencies among objects associated with the process are not taken into account when defining specialization.

Weidlich et al. [2010] studied process model compatibility based on behavior inheritance and projection. Later, they proposed a concept of behavioral profile for capturing behavioral constraints of a process model [Weidlich et al. [2011]. Their notion is opposed to the trace equivalence concept and consistency measures but can be effectively used to quantify the differences between two process models.

The existing work discussed above have mainly focused on a single (object) lifecycle inheritance otherwise have treated a process as a single object. Based on those works, it is easy to see that the study of a specialization methodology in a conventional object-oriented design approach can be reused and extended to support the artifact-centric process model. From characteristics of artifact-centric business processes, process specialization should not only be used for specializing a single artifact but synchronization dependencies between artifacts. Object lifecycles and interactions have been studied in various research domains including adaptation and changes of a process [Muller et al., 2008], design compliance [Küster et al., 2007; Ryndina et al., 2007; Künzle and Reichert, 2011; Lohmann, 2011], scenario-based specification [Uchitel et al., 2003], conformance checking [Fahland et al., 2011], and service contract for inter-organizational workflows [Van Der Aalst et al., 2010].

Fahland et al., [2011] proposed conformance checking to compare the behavior described by a process model to process executions in an actual information system. Their method is based on behavioral conformance and interaction conformance via the use of proclet systems to describe artifacts and their interactions and checking how good a given proclet system describes all events recorded in the database. [Künzle and Reichert, 2011] proposed a PHILharmonicFlows framework to support object-aware process management including important aspects in process modeling, execution and monitoring. Apart from considering objects, relations, and attributes, they also studied object behavior as well as object interactions to support modeling of processes a different level of granularity. Ryndina et al., [2007] and Küster et al., [2007] proposed a similar approach to establishing consistency of a business process model and an object life cycle and the approach is based on consistency notions (life cycle compliance and coverage) related to the concepts of equivalence and refinement of formal process specifications. Lohmann, N. [2013] presented an approach to automatically construct business process models that are compliant by design. The approach is based on compliance rules which express constraints on activity execution, data, and location information and by composing the rules to an artifact-centric model, checking for compliant behavior can be reduced to instead that for the reachability of final states. All the above work laid a theoretical ground for the development of our *B-consistency* notion and consistency checking presented in our work. Based on this foundation, our work extends the existing notions with our proposed fragmental consistency checking technique to support artifact inheritance methods which include refinement, extension and reduction and how the consistency notion can be adopted and applied.

In our previous work [Yongchareon et al., 2012], we have preliminarily studied how artifact-centric business processes can be specialized. The previous study includes understanding synchronization dependency between artifacts and how their behavioral consistency can be maintained. In this paper, we extend our previous work in several aspects including: (1) improvement to the specialization methods discussed in Sections 4.1 and 4.2, (2) improvement to the previous *B-consistency* notion, (3) improvement to the fragmental analysis method via the introduction of the *SL-fragments* and *S-region* notions to help verify overall *B-consistency* based on a region without validating the entire model, (4) a new propagation method for artifact reduction, and (5) a new tool prototype (Artifact-M). We also provide all necessary and sufficient lemmas and theorems to support the study along with proofs as outcomes of the new study.

6.2 Configurable process models

It is well perceived that business process modeling is labor-intensive and highly detailed as to support the development of workflow systems. To cope with this issue, consortia and vendors have defined reference process models to avoid having to repeatedly create process models from scratch. Reference models are generalized to capture recurrent business operations in a given domain allowing them to be individualized to fit the specific requirements of different organizations, thus, promoting the reuse of proven practices (e.g. Supply Chain Operations Reference Model (SCOR) [Stephens, 2001], IT Infrastructure Library (ITIL) [Taylor and Probst, 2003], SAP Reference Model [Curran and Keller, 1997]). This type of models is known as a configurable model. Several works have been proposed to provide a configuration of models in different extended modeling languages (e.g., C-EPCs [Rosemann and van der Aalst, 2007], C-iEPCs [La Rosa et al., 2011], C-WF-nets [van der Aalst et al., 2010], C-SAP and C-BPEL [Gottschalk et al., 2008], and C-YAWL [Gottschalk et al., 2008]).

van der Aalst et al., [2008] proposed a framework for configuring reference process models (based on WF-nets) in a correctness-preserving manner. The framework includes a technique to derive propositional logic constraints and guarantee the syntactic correctness of the derived model. The framework permits the correctness checking at any intermediate step of the configuration. A set of constraints is evaluated at the time a value is assigned to a variation point. The configuration step is applied if the constraints are satisfied; otherwise, a reduced propositional logic formula is computed to help identify additional variation points required to be configured simultaneously as to preserve the semantic correctness. A workflow that is derived based on a configuration step from a sound workflow is always sound. Later, they showed in [van der Aalst et al., 2010] that the state-space explosion problem can be avoided from their behavioral correctness checking of process configurations.

Most recently, van der Aalst et al., [2012] proposed a novel approach for verifying configurable process models. The approach is inspired by the Operating Guidelines (OGs) used for partner synthesis [Massuthe and Schmidt, 2005; Lohmann et al., 2007] by viewing the configuration process as an external service and computing a characterization of all such services which meet particular requirements via the notion of configuration guideline. This work was motivated from the problem that the verification of the models can be difficult because the number of possible configurations, which can be achieved by restriction (i.e., hiding and blocking on tasks), grows exponentially from the number of configurable elements. In addition, concurrency and branching structures may cause configuration decisions to interfere with each other, and therefore, introduce deadlocks, livelocks, and other anomalies. The result showed that all configurations posing no behavioral problems can be characterized at design time instead of repeatedly checking every single step of configuration. Their approach is highly generic and imposes no constraints on the configurable process models, and all computations are done at design time and not at configuration time. The approach is implemented in a YAWL Editor [YAWL Foundation, 2012] featured with Wendy tool [Lohmann, N., Weinberg, 2010] to ensure correctness while configuring C-YAWL models.

We observed that with the artifact-centric approach, it is natural to construct a repository of artifact classes and the process models likewise in an object-oriented software design approach. With this nature together with the specialization mechanism (of artifacts and a process), the artifacts and their involved process can be reused in a more flexible, componentized way.

6.3 Design by selection and patterns

Similar idea to the reuse of reference process model, the design by patterns or selection takes a use of existing models in process model repositories with the help from pre-defined patterns or queries to retrieve existing models, and then customize or generate new models based on the existing models.

Zdun et al. [2007] proposed a pattern language for process-oriented integration of services to describe the practices with a modeling concept based on a catalog of 13 pattern primitives defined based on Unified Modeling Language version 2 (UML2) profile [OMG, 2004] for activity diagrams and Object Constraint Language (OCL) [OMG, 2003], each of primitives is precisely specified modeling element that represents a pattern. A model-driven tool has been developed to support the generation and validation of models in other languages, e.g., BPEL.

Awad et al. [2011] observed that the information process model repositories are not fully exploited during process modeling, thus reducing the efficiency and quality of process design. As such, they proposed a design by selection approach for business process design that uses process repositories and facilitates reuse of BPMN process model components, which can be static or flexible. Static components represent the specific aspects of the process model, while flexible components can be customized and reused in an efficient manner. Their approach was built on top of the Oryx [Decker et al., 2008], which is an open process modeling platform and repository, and the BPMN-Q query language [Award, 2007; Sakr and Awad, 2010] to retrieve process components from the repository.

Process model similarity search is used to support the process models retrieval from repositories of business process models. It focuses on finding the similarity between two process models based on their either structural aspect or/and behavioral aspects.

van Dongen et al., [2008] proposed to use causal footprints as an abstract representation of the behavior model derived from an EPC-based process model. Based on the causal footprint of two models, they used a vector space model [Salton et al., 1975] from information retrieval area for the calculation of their similarity, and validated their approach by using SAP Reference Model with an implementation in the ProM framework [Process Mining Group, 2012].

Dijkman et al., [2011] presented three similarity metrics that are used to answer queries on process repositories dealing with the problem of retrieving process models in the repositories that resemble a given process model or a fragment of a model.

(1) Node matching similarity to compare element labels and attributes attached to model elements between two models

(2) Structural similarity to compare element labels and the topology between two models

(3) Behavioral similarity to compare element labels and causal relations captured in the models

All of the above approaches provide means to support the reuse of existing models stored in process model repositories in more semantic manner. As their models are constructed based on the activity and control flows, the retrieval mechanisms of a whole process model or a fragment of model do not take the data aspects (from input/output of tasks) into account, e.g., the data dependencies between two models or fragments are not considered. In the artifact-centric approach, the repositories should store a collection of artifact classes and a process definition, which defines how they are used for particular business processes. A traditional object-oriented reuse mechanism can be applied for artifact classes; however, the reuse of process definition that describes which artifact class is used in a business process and the relationships and dependencies among them needs further investigation.

From a perspective of artifact-centric modeling, Calvanese et al. [2009] studied challenges around the comparison of artifact-centric processes using the notion of dominance. The notion is applied to check whether the executions of one process emulate another comparing process. Nevertheless, this study mainly considered comparing snapshots of the process execution but changes of internal lifecycles (e.g., extension, refinement, and reduction of artifacts) of the behavior of artifact lifecycles and their dependencies are not considered.

7. CONCLUSIONS AND FUTURE WORK

In this article, we presented how our process specialization approach can help process modelers to construct and reuse existing business processes. Stemming from object-oriented design and modeling principle, our approach is deemed to provide a higher degree of adaptability and modularity than traditional activity-centric business process modeling approaches. As discussed, the object specialization considers the reuse of individual object class in software development, while our approach introduces the process-centric reuse to business process management. We attempt to mingle the existing software design principle with the design of business processes in a more coherent way. This should at least expand the area of research on how software engineering concepts can be realized and applied to business process management. Particularly, we studied the synchronization between lifecycles and demonstrated how the behavior consistency of two processes can be guaranteed.

By applying our behavior-consistent restriction to the specialization mechanism, our approach not only improves the modeling and design efficiency of business processes but also allows for efficient runtime management capabilities. We are capable of aggregate lifecycles of artifacts and processes and view them at a more abstract level. This feature facilitates the consistent monitoring and reporting of business data and processes. The major contribution of our work is a theoretical study on the necessary and sufficient behavioral-preserving consistency conditions for specializing artifact lifecycles and their synchronization dependencies. In the future, we plan to extend and customize our framework to address different requirements from real-world industrial case studies and then evaluate how well the fully-fledged framework can be applied in practice.

APPENDIX

In this appendix, we provide detailed proofs of our Lemmas and Theorems.

PROOF OF LEMMA 4.1. We can prove the lemma by construction using the conditions of an AL-fragment (in *Definition 4.5*), the three inference rules for lifecycle composition (in *Definition 3.9*), and the soundness condition (in *Definition 3.11*) to show that if all the conditions of the lemma hold, the composition of all SL-fragments is *atomic* and *sound*. We prove the necessity of each of the four conditions as follows.

—*For the first condition*. We have if $\ell^{C_i} \in \Gamma$ is not an AL-fragment, then the resulted fragment from the composition of ℓ^{C_i} and any other fragment is not an AL-fragment. This is because, based on the three inference rules for the lifecycle composition defined in Definition 3.9, if ℓ^{C_i} has either multiple entry or exit transitions or both, the composition yields multiple transitions for the synchronized product as well. The first condition holds the soundness condition since the AL-fragment is always *sound* and no synchronization is stated in this condition.

The second, third, and fourth conditions of Lemma 4.1 are used to restrict two SL-fragments (to be composed for S-region) to have all sync rules and transitions that are necessary for synchronizing L-fragments in Γ. We can prove that the three conditions are necessary as follows.

—*For the second condition*. Consider a transition with a sync rule. Based on the inference rule 3.3 for the synchronization composition defined in *Definition 3.9*, every sync rule that is used to synchronize between two lifecycles, those transitions, and states related to the sync rule will be included in the synchronized product. So, if a sync rule is used to synchronize ℓ^{C_i} with any L-fragment that is not in Γ, then a transition and its related states of such L-fragment will be included in the composition result. This clearly means that there will exist a transition from/to a state that does not belong to any L-fragment in Γ; therefore, the result fragment does not satisfy the condition of AL-fragment.

—*For the third and fourth conditions*. The third condition is used to restrict all the entry transitions of one fragment to be synchronized with all the entry transitions of another fragment to be composed. Similarly, the third condition is for the exit transitions. Consider an entry or exit transition with a sync rule. Assume two synchronized L-fragments with multiple entry transitions and there exists an entry transition in one fragment that does not synchronize with any entry transition of another fragment. Based on the inference rule 3.3 in *Definition 3.9*, the composed entry transition in the synchronized product derived from that transition will never fire since no sync rule is induced on it; therefore, the goal-reachability of the composed fragment is violated. The same problem also occurs in the case of having an exit transition of one fragment without a sync rule on the exit transition of another fragment to be composed. Therefore, the soundness cannot be guaranteed without these two conditions.

This completes the proof of LEMMA 4.1. □

PROOF OF LEMMA 4.2. We can prove it by construction using the ex-lifecycle condition (in *Definition 4.11*), the *B-consistency* condition (in *Definition 4.2*), the condition for atomic composition of SL-fragments (in LEMMA 4.1), and the condition of *B-consistent* refined L-fragment (in THEOREM 4.1) to show that if the conditions of LEMMA 4.2 hold, ℓ is *B-consistent* with $\mathcal{L}'_Y \otimes \ell$. Revisiting the four conditions in LEMMA 4.1, the composition of two SL-fragments is considered as a composite AL-fragment in the synchronized product if the SL-fragments are AL-fragment and the sync rules of entry/exit transitions of one fragment completely synchronize the entry/exit transitions of another fragment. Here in LEMMA 4.2, the AL-fragment condition conforms to the first condition of Lemma 4.1 and the ex-lifecycle condition (in *Definition 4.11*) conforms to the second, third, and fourth conditions of the LEMMA 4.1. Followed from THEOREM 4.1, the composed lifecycle can be considered as a refined, composite AL-fragment (composed of $\mathcal{L}'_{C_y}$ and ℓ) in ℓ; therefore, the composed lifecycle is *B-consistent* with ℓ. □

PROOF OF LEMMA 4.3. Similar to the proof of LEMMA 4.2, we can prove it by construction using *Definition 4.12*, the *B-consistency* condition (in *Definition 4.2*), the condition for the atomic composition of SL-fragments (in LEMMA 4.1), and the condition of *B-consistent* refined L-fragment (in THEOREM 4.1). This proof can be achieved based on the proof of LEMMA 4.1. If the composition of two synchronized fragments in the refined S-region is atomic, then based on THEOREM 4.1 (by considering refined S-region as a refined L-fragment), the *B-consistency* is preserved. □

PROOF OF LEMMA 4.4. The proof can be derived from LEMMA 4.2 and LEMMA 4.3 as the refinement of an existing artifact satisfies the condition of LEMMA 4.3 and the artifact extension satisfies the condition of LEMMA 4.2. □

PROOF OF LEMMA 4.5. The proof can be derived from LEMMA 4.4 and *Definition 4.13* by taking into account the transitivity property of sync rules and the lifecycle composition (in *Definition 3.9*). □

PROOF OF LEMMA 4.6. We can prove it by construction using *Definition 4.14*, *Definition 4.15*, *Definition 4.16*, the *B-consistency* condition (in *Definition 4.2*), and the condition for the atomic composition of SL-fragments (in LEMMA 4.1) to show that if the conditions of LEMMA 4.6 holds, the lifecycle composition of every transition in T^{re} is *B-consistent* with the lifecycle composition of every L-fragment in L^{re} and $\mathcal{L}_Y$. For every reducible L-fragments in L^{re}, it is reduced into a transition. Based on *Definition 4.14*, *Definition 4.15*, and LEMMA 4.1, the composition of reducible L-fragments (with corresponding reduced sync rules) can be considered as a composite AL-fragment; therefore, can be reduced without violating the *B-consistency* condition. □

PROOF OF THEOREM 4.1. The theorem can be proved by checking: for each statement of the theorem, a refined L-fragment (in *Definition 4.5*) does not violate the *B-consistency* condition (in *Definition 4.2*).

—Consider the first statement of the theorem. An AL-fragment that refines a base lifecycle always preserves the *B-consistency* condition as the AL-fragment is atomic and it has a single-entry state and a single exit state. An entire lifecycle of the L-fragment can be completely reduced (or abstracted) in a single transition if such L-fragment is atomic, i.e., being an AL-fragment. We can see that the condition of AL-fragment (in *Definition 4.5*) naturally conforms to Conditions 4.1 and 4.2 of the *B-consistency* (*Definition 4.2*)

—Consider the second statement of the theorem. We can see that the condition of this statement restricts a refined L-fragment to be completely encapsulated within a single state. For every L-occurrence of the L-fragment, it must be originated from a transition fired from an outside state (not in the L-fragment) and must reach to a transition fired to another outside state. We can see that the condition for substituting a state with a refined L-fragment conforms to Condition 4.3 of the *B-consistency* (*Definition 4.2*).

This completes the proof of THEOREM 4.1 □

PROOF OF THEOREM 4.2. The theorem can be proved by construction using the three inference rules for lifecycle composition (in *Definition 3.9*), the soundness condition (in *Definition 3.11*), and the *B-consistency* condition (in *Definition 4.2*). As refined L-fragments do no introduce any synchronization dependencies to specialized artifacts, then the proof follows that the composed lifecycle is always *sound*. As the conditions of THEOREM 4.1 restrict a specialized artifact preserves the *B-consistency* when applying refined L-fragments, the lifecycle composition of every specialized artifact is *B-consistent*. □

PROOF OF THEOREM 4.3. We can prove the theorem as follows.

—*For the if condition*, we must prove that if the two statements are satisfied, then Π' is *B-consistent* with Π. For the first condition, we prove that each specialized artifact needs to be *B-consistent* with its base artifact. Based on *Definition* 4.4, this statement follows from THEOREM 4.2. For the second condition, we prove that why each sync specialization needs to be S-consistent. Based on *Definition* 4.18, this statement follows from LEMMA 4.2, LEMMA 4.3, LEMMA 4.5, and LEMMA 4.6. As each of Lemmas has the condition to preserve the *B-consistency* for each method of sync specialization (*extension*, *refinement*, and *reduction*), therefore, the second statement holds.

This completes the proof of the if direction.

—*For the only if condition*, we must prove that the two statements satisfy if Π' is a behavior-consistent specialization of Π. This can be proved based on the definition of ACP Specialization (in *Definition 4.1*) and the definition of lifecycle specialization *B-consistency* (in *Definition 4.4*), and S-consistency (*Definition 4.18*). In ACP Specialization, we define the three specialization methods: *artifact extension*, *refinement*, and *reduction*. First, the lifecycle *B-consistency* of each specialized artifact must hold as it follows from THEOREM 4.2. Then, consider the three *S-consistency* conditions based on artifact all these specialization methods.

—For *artifact extension*. A newly added artifact is not needed to be *B-consistent*. Either LEMMA 4.2 (sync extension) or LEMMA 4.5 (sync refinement of existing and extended artifacts), or the combination of them is required.

—For *artifact refinement*. If there is no synchronization considered, the *B-consistency* for artifact refinement follows from THEOREM 4.2. With synchronization, THEOREM 4.2 and LEMMA 4.3 (sync refinement) are required.

—For *artifact reduction*. A removed artifact is not needed to be *B-consistent*. But the existing artifact with reduced lifecycle must be *B-consistent* and it follows from THEOREM 4.2 and LEMMA 4.6 (sync reduction).

This completes the proof of the only if direction.

Therefore, the proof of THEOREM 4.3 is complete. □

REFERENCES

Awad, A.: BPMN-Q: A Language to Query Business Processes, in proceedings of the 2nd international workshop on Enterprise modelling and information systems architectures concepts and applications (EMISA) 2007, Gesellschaft für Informatik, Bonn, pp. 115-128.

Awad, A., Sakr, S., Kunze, M., Weske, M.: Design by Selection: A Reuse-Based Approach for Business Process Modeling, in proceedings of the 30th International Conference on Conceptual Modeling (ER) 2011, LNCS 6998, pp. 332–345.

Bagheri Hariri, B., Calvanese, C., De Giacomo, G., De Masellis, R., Felli, P.: Foundations of Relational Artifacts Verification, in proceedings of BPM 2011, LNCS 6896, pp. 379–395.

Bhattacharya, K., et al.: Artifact-centered operational modeling: Lessons from customer engagements, in IBM SYSTEMS JOURNAL, 2007, pp. 703-721.

Bhattacharya, K., Gerede, C, Hull, R., R., Liu, R., Su, J.: Towards Formal Analysis of Artifact-Centric Business Process Models, in: proceedings of the International Conference on Business Process Management (BPM) 2007, LNCS 4714, pp. 288-304.

Bloom, B.: Fundamental study structural operational semantics for weak bisimulations, Theoretical Computer Science 146 (1995), pp. 25-68.

Calvanese, D., De Giacomo, G., Hull, R., Su, J.: Artifact-centric workflow dominance, in proceedings of ICSOC-ServiceWave 2009. LNCS 5900, pp. 130–143.

Chao, T., Cohn, D., Flatgard, A., Hahn, S., Linehan, M., Nandi, P., Nigam, A., Pinel, F., Vergo, J., Wu, F.: Artifact-Based Transformation of IBM Global Financing, in BPM 2009, LNCS 5701, pp. 261–277.

Chiao, C.M., Künzle, V., Reichert, M.: *Object-aware Process Support in Healthcare Information Systems: Requirements, Conceptual Framework and Examples,* Int'l Journal on Advances in Life Sciences, 2013, 5 (1 & 2). pp. 11-26.

Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Engineering Bulletin 32(3), 3–9 (2009)

Curran, T., Keller, G.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model, Upper Saddle River (1997)

Damaggio, E., Deutsch, A., Hull, R., Vianu, V.: Automatic Verification of Data-Centric Business Processes, in proceedings of BPM 2011, LNCS 6896, pp. 3–16

Damaggio, E., Deutsch, A., Vianu, V.: Artifact systems with data dependencies and arithmetic constraints, in proceedings of the International Conference on Database Theory, ICDT (2011), pp. 66-77

Damaggio, E., Hull, R., Vaculin, R.: On the Equivalence of Incremental and Fixpoint Semantics for Business Artifacts with Guard-Stage-Milestone Lifecycles, in proceedings of BPM 2011, LNCS 6896, pp. 396–412.

Decker, G., Overdick, H., Weske, M.: Oryx – sharing conceptual models on the web, in proceedings of ER 2008, LNCS 5231, pp. 536–537

Deutsch, A., Hull, R., Patrizi, F., Vianu, V.: Automatic verification of data-centric business processes. In: proceedings of the International Conference on Database Theory, ICDT (2009), pp. 252-267

Dijkman, R., Dumasm M., van Dongen, B., Kaarik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation, Information Systems, 2011, vol. 36, pp. 498–516

Fahland, D., Leoni, M.D., van Dongen, B.F., van der Aalst, W.M.P.: Conformance Checking of Interacting Processes with Overlapping Instances, in proceedings of BPM 2011, LNCS 6896, pp. 345-361

Fahland, D., Leoni, M.D., van Dongen, B.F., van der Aalst, W.M.P.: Behavioral Conformance of Artifact-Centric Process Models, in proceedings of BIS 2011, LNBIP 87, pp. 37–49, 2011

Fritz, C., Hull, R., Su, J.: Automatic construction of simple artifact-based business processes, in proceedings of ICDT 2009. ACM, vol. 361, pp. 225–238

Gerede, C.E., Su, J: Specification and Verification of Artifact Behaviours in Business Process Models, in proceedings of ICSOC 2007, LNCS 4749, pp. 181-192

Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M., Rosa, M.L.: Configurable workflow models, International Journal of Cooperative Information Systems, 2008, vol. 17 (2), pp. 177–221

Harel, D., Kupferman, O.: On Object Systems and Behavioral Inheritance, IEEE Transactions on Software Engineering, 2002, vol. 28, no. 9, pp. 889-903

Hariri, B.B., Calvanese, D., Giacomo, G.D., Masellis, R.D., Felli, P.: Foundations of Relational Artifacts Verification, in proceedings of BPM 2011, LNCS 6896, pp. 379–395

Hull, R., Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges, in proceedings of On the Move to Meaningful Internet Systems: OTM 2008, LNCS 5332, pp. 1152-1163

Johnson, R., Pearson, D., Pingali, K.: The program structure tree: Computing control regions in linear time, in: PLDI. Proceedings of the ACM SIGPLAN'94 Conference on Programming Language Design and Implementation, 1994, ACM Press, pp. 171–185

Klai, K., Tata, S., Desel, J.: Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes, Data & Knowledge Engineering, 2011, vol. 70, pp. 467-482

Koutsos, A. and Vianu, V.: Process-centric views of data-driven business artifacts. J. Comput. System Sci., 86:82–107, 2017

Kucukoguz, E., Su, J.: On Lifecycle Constraints of Artifact-Centric Workflows, in proceedings of WS-FM 2010, LNCS 6551, pp. 71–85

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

Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object aware process management. Journal of Software Maintenance and Evolution: Research and Practice (2011), vol. 13, issue 4, pp. 205-244.

Küster, J. M., Ryndina, K., and H. Gall: Generation of Business Process Models for Object Life Cycle Compliance, in proceedings of the International Conference on Business Process Management (BPM) 2007. LNCS 4714, pp. 165-181

Lee, J., Wyner, G.M.: Defining specialization for dataflow diagram, Information Systems, 2003, vol. 28, pp. 651–671

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

Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling business contexture and behavior using business artifacts, in proceedings of CAiSE 2007. pp. 324-339

Lohmann, N.: Compliance by Design for Artifact-Centric Business Processes, in proceedings of BPM 2011, LNCS 696, pp. 99-115

Lohmann, N.: Compliance by design for artifact-centric business processes, Information systems, 2013, vol. 38, pp. 606-618.

Lohmann, N., Weinberg, D.: Wendy: a tool to synthesize partners for services, in proceedings of the International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, 2010, LNCS 6128, pp. 297–307

Lohmann, N, Wolf, K.: Artifact-centric Choreographies, in proceedings of ICSOC 2010. LNCS, vol. 6470, pp. 32-46Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Systems Journal 42(3), 2003, pp. 428–445

Lohmann, N., Massuthe, P., Wolf, K.: Operating Guidelines for Finite-State Services, in proceedings of ICATPN 2007, LNCS 4546, pp. 321–341

Massuthe, P., Schmidt, K.: Operating Guidelines - an Automata-Theoretic Foundation for the Service-Oriented Architecture, in proceedings of the Fifth International Conference on Quality Software (QSIC'05), pp. 452- 457

Muller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures, in proceedings of CAiSE 2008, LNCS 5074, pp. 48-63

Object Management Group (OMG), UML 2.0 OCL final adopted specification, 2003, Tech. rep. ptc/03-10-14,

Object Management Group (OMG), UML 2.0 superstructure final adopted specification, 2004, Tech. rep. ptc/04-10-02

OWL Services Coalition, OWL-S: Semantic markup for web services, 2003

Process Mining Group, ProM Framework, 2012, http://www.processmining.org/prom/start

Ryndina, K., Küster, J. M., and H. Gall: Consistency of Business Process Models and Object Life Cycles, in proceedings of MoDELS 2006 Workshops, 2007, LNCS 4364, pp. 80–90.

Reisig, W.: Petri Nets, EATCS Monographs on Theoretical Computer Science, 1985, Springer, Berlin, Heidelberg, New York

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

Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language, Information Systems, 2007, vol. 32 (1), pp. 1–23

Sakr, S., Awad, A.: A Framework for Querying Graph-Based Business Process Models, in proceedings of WWW 2010, ACM, pp. 26–30

Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing, Communications of the ACM, 1975, vol. 18(11), pp. 613–620

Schrefl, M., Stumptner, M.: Behavior-Consistent Specialization of Object Life Cycles, 2002, ACM TOSEM, vol. 11, No. 1, pp. 92-148

Stephens, S.: The Supply Chain Council and the SCOR Reference Model. Supply Chain Management - An International Journal 1(1), 9–13 (2001)

Taylor, C., Probst, C.: Business Process Reference Model Languages: Experiences from BPI Projects, in proceedings of INFORMATIK 2003, Jahrestagung der Gesellschaft f¨ur Informatik e. V (GI), pp. 259–263 (2003)

Uchitel, S., Kramer, J., Magee, J.: Synthesis of Behavioral Models from Scenarios, IEEE Transactions on Software Engineering, 2003, vol. 29(2), pp. 99–115

van der Aalst, W.M.P.: Inheritance of Business Processes: A Journey Visiting Four Notorious Problems, Petri Net Technology for Communication-Based Systems, 2003, pp. 383-408

van der Aalst, W.M.P., Basten, T.: Life-cycle Inheritance: A Petri-net-based Approach, Application and Theory of Petri Nets 1997, LNCS 1248, pp. 62–81

van der Aalst, W.M.P., Basten, T.: Identifying Commonalities and Differences in Object Life Cycles using Behavioral Inheritance, Application and Theory of Petri Nets 2001, LNCS 2075, pp. 32–52

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

van der Aalst, W.M.P., Dumas, M., Gottschalk, F., ter Hofstede, A.H.M., Rosa, M.L., Mendling, J.: Preserving correctness during business process model configuration, Formal Aspects of Computing, 2010, vol. 22 (3), pp. 459–482

van der Aalst, W.M.P, Lohmann, N, La Rosa, M.: Ensuring correctness during process configuration via partner synthesis, Information Systems, 2012, vol. 37, pp. 574–592

van der Aalst, W.M.P, Lohmann, N, Massuthe, P, Stahl, C, Wolf, K: From Public Views to Private Views – Correctness-by-Design for Services, in proceedings of WS-FM 2007, LNCS, vol. 4937, pp. 139-153

van der Aalst, W.M.P, Lohmann, N, Masuthe, P, Stahl C, and Wolf, K.: Multipart Contracts: Agreeing and Implementing Interorganizational Processes, The Computer Journal, 2010, Vol. 53, No. 1, pp.90-106

van der Aalst, W.M.P, Dumas, M., Gottschalk, F., ter Hofstede, A.H.M., Rosa, M.L., Mendling, J.: Correctness-Preserving Configuration of Business Process Models, in proceedings of Fundamental Approaches to Software Engineering (FASE 2008), LNCS 4961, pp. 46–61

van der Aalst, W.M.P, Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language, in proceedings of WS-FM 2006, LNCS 4184, pp. 1–23

van der Aalst, W.M.P, ter Hofstede, A.H.M.: YAWL: yet another workflow language. Information System (IS), 2005, vol. 30(4), pp. 245-275

van der Aalst WMP, ter Hofstede AHM, Kiepuszewski B, Barros AP: Workflow patterns, 2013, Distributed Parallel Database, vol. 14(1), pp. 5–51

van der Aalst, W.M.P., Weske, M.: The P2P Approach to Interorganizational Workflows, in proceedings of CAiSE 2001, LNCS 2068, pp. 140–156

van der Aalst, W.M.P., Weske, M.: Case handling: a new paradigm for business process support. Data and Knowledge Engineering, 2005, vol. 53(2), pp. 129–162

van Dongen, B., Dijkman, R., Mendling, J.: Measuring Similarity between Business Process Models, in proceedings of CAiSE 2008, LNCS 5074, pp. 450–464

Vanhatalo, J., Volzer, H., Leymann, F.: Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition, in proceedings of ICSOC 2007, LNCS 4749, pp. 43–55

Weidlich, M., Dijkman, R., Weske, M.: Deciding Behaviour Compatibility of Complex Correspondences between Process Models, in proceedings of BPM 2010, LNCS 6336, pp. 78-94

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.

Wyner, G.M., Lee, J.: Process Specialization: Defining Specialization for State Diagrams, Computational & Mathematical Organization Theory, 2002, vol. 8, pp. 133–155

YAWL Foundation, 2012, http://ww.yawlfoundation.org/

Yongchareon, S., Liu, C., Zhao, X.: A Framework for Behavior-Consistent Specialization of Artifact-Centric Business Processes, in: BPM 2012, LNCS 7481, pp. 285-301

Yongchareon, S., Liu, C., Yu, J., Zhao, X.: A View Framework for Modeling and Change Validation of Artifact-Centric Inter-Organizational Business Processes, Information systems, 2015, vol. 47, pp. 51-81.

Zhao, X., Su, J., Yang, H., Qiu, Z.: Enforcing Constraints on Life Cycles of Business Artifacts, in proceedings of the 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering, 2009, pp. 111-118

Zdun, U., Hentrich, C., Dustdar, S.: Modeling Process-Driven and Service-Oriented Architectures Using Patterns and Pattern Primitives, 2007, ACM Transactions on the Web, Vol. 1, No. 3, Article 14