On Extending Support for Modeling Artifact-Centric Business Processes

by

Sira Yongchareon

A thesis submitted for the degree of
Doctor of Philosophy
(Information and Communication Technologies)

Faculty of Information and Communication Technologies
Swinburne University of Technology
Victoria, Australia

December 2012
Abstract

In the past few years, a new approach to modeling business processes has emerged – that is artifact-centric business process modeling. Rather than primarily focusing on the activity flows of a business process as in a conventional activity-centric approach, the artifact-centric approach focuses on key business-relevant entities that evolve as they progress through the operations of the process. Business artifacts integrate both data aspects and process aspects into a holistic unit, and serve as basic building blocks in a process model. The approach provides an intuitively natural, robust structure for understanding and specifying business processes and operations, thus, enabling a natural modularity and componentization of business operations and varying levels of abstraction.

Since 2003, several research works have been conducted for the development of meta-models, methods, tools, user-centric and other technologies in support of the artifact-centric paradigm. They show a promising trend in business process management and the state of the art of the artifact-centric approach. However, there is still a need for more research in various dimensions to help improving the wider usefulness and strength of this modeling approach. Particularly, in this thesis, we identify three key challenging issues for the support of artifact-centric business process modeling: (1) support for business process reuse, (2) support for efficient inter-organizational business process collaboration, and (3) support for process visualization and user interfaces development.

In this thesis, we develop frameworks to tackle the above three issues. Firstly, we propose a specialization framework to address the issue of how organizations can systematically reuse their existing business processes in a more efficient way. We propose the behavioral consistency notion called B-consistency for analyzing dynamic behavior of artifacts that interact with each other in a business process. This notion is used for guaranteeing that a specialized process is consistently observable from the base process it derives from. Secondly, we propose a view framework for artifact-centric inter-organizational business processes to address three major requirements for efficient collaboration: compliance, flexibility, and
autonomy. The framework consists of an artifact-centric inter-organizational business process model, the notions of public and private views, view consistency rules, and an algorithm for automatically finding minimal, consistent abstraction from the model. Lastly, we propose a User Interface Flow (UIF) model and algorithms for automatically generating UIF model for a business process from an underlying ACP-i model to support the visualization of business processes and the MDA-based development of process-oriented user interfaces.
Declaration

This is to certify that this thesis contains no material which has been accepted for the award of any other degree or diploma and that to the best of my knowledge this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis. Where the work is based on joint research or publications, I have disclosed the relative contributions of the respective workers or authors.

Signed by: Sira Yongchareon

December, 2012

Melbourne, Australia
Acknowledgement

There are a lot of people I would like to thank for a variety of reasons.

It is difficult to overstate my gratitude to my supervisor, Professor Chengfei Liu. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped me initiate and forward this research. I deeply appreciate his considerable effort and invaluable comments in all of my past research papers and this thesis. I do thank my associate supervisor, Dr. Xiaohui Zhao, for his support and guidance, especially for his time in many critical advices on how to solve difficult issues and improve the quality of my papers. I also thank Dr. Jian Yu for giving me valuable feedbacks on a journal article submission.

I appreciate useful discussions from several friends of mine who have been walking together in the same direction with me during my past three and a half years study – Jianxin Li, Rui Zhou, Jiajiei Xu, Hai Huang, Shangfeng Hu, and Liang Yao – you guys are such wonderful friends. Especially, Saiful Islam, Tuan Nguyen, Tuong Huan Nguyen, and Luan Lam are ones of my unforgettable friends that need to remember. With all of these guys, my research was not just only the thing that I wanted to do but also a part of my life that I have enjoyed.

Many thanks to Professor Doug Grant, Ken Mcinnes, Peter Eden, Peter Sala, Phil Joyce, and Robert Tipping who I have worked with, I have gained valuable academic experience from being a tutor for your subjects during my research life. I have learnt a lot of things in the academia.

Leaving the best to last, I wish to thank my beloved parents, Wittaya Yongchareon and Vannapha Yongchareon. They bore me, raised me, supported me, taught me, and loved me. I am also very deeply grateful to my love, Arjcharaporn Pithaksa, for her love, understanding, patience, encouragement, sacrifice, and help. To them I dedicate this thesis.

Thanks to everyone who has ever thought about me. I wish you all the best.
# Table of contents

List of figures ................................................................................................................... vi
List of tables ................................................................................................................... viii
List of publications ........................................................................................................... ix
Chapter 1. Introduction ...................................................................................................... 2
  1.1 Artifact-Centric Approach to Modeling Business Processes ............................. 2
  1.2 Key Issues in Supporting Artifact-Centric Process Modelling ..................... 6
  1.3 Research Contributions .................................................................................... 11
  1.4 Thesis Outline .................................................................................................. 13
Chapter 2. Literature Review ........................................................................................... 16
  2.1 Artifact-Centric Approach to Modeling Business Processes ........................... 17
  2.2 Business Process Reuse .................................................................................... 27
  2.3 Facilitating Inter-Organizational Business Process Collaboration ................... 35
  2.4 User Interfaces for Business Processes ............................................................ 43
  2.5 Summary and Discussion ................................................................................. 49
Chapter 3. Modeling Artifact-Centric Business Processes .............................................. 50
  3.1 Running Example ............................................................................................. 51
  3.2 Syntax of Artifact-Centric Process (ACP) model ............................................ 52
  3.3 Behavioral Properties and Model Verification ................................................. 57
  3.4 Summary and Discussion ................................................................................. 62
Chapter 4. Specializing Artifact-Centric Business Processes .......................................... 64
  4.1 Motivation to Process Specialization and Problem Statements ....................... 65
  4.2 Specializing Artifact-Centric Business Process Models .................................. 68
  4.3 Behavioral Consistency Analysis ..................................................................... 71
  4.4 Behavior-Consistent Specialization ................................................................. 80
  4.5 Case Study ........................................................................................................ 99
  4.6 Summary and Discussion ............................................................................... 101
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration .............. 104
  5.1 Motivating Example and Problem Analysis ................................................... 105
  5.2 Artifact-Centric Process View Framework .................................................... 110
  5.3 Public View Construction .............................................................................. 117
  5.4 Process Changes and Change Validation ....................................................... 137
  5.5 Open Issues .................................................................................................... 144
  5.6 Summary and Discussion ............................................................................... 145
Chapter 6. Deriving Process-Oriented User Interfaces .................................................. 148
  6.1 Problems and UI Derivation Framework ....................................................... 149
  6.2 User Interface Flow (UIF) Model ................................................................. 152
  6.3 Deriving UIF Model for ACP Model ............................................................ 155
  6.4 ACP Execution System Architecture with UIF Support .............................. 165
  6.5 Summary and Discussion ............................................................................... 169
Chapter 7. Conclusion and Future Work ........................................................................ 172
  7.1 Summary of Contributions ............................................................................. 172
  7.2 Open Problems and Future Work ................................................................... 176
List of figures

Figure 1.1: Four “dimensions” in artifact-centric process modeling [Hull, 2008] ............5
Figure 2.1: ER Model for entities used in a purchasing process........................................18
Figure 2.2: Example of artifact lifecycles and their dependencies ..................................20
Figure 2.3: Three logical levels of Artifact-Centric BPM [Bhattacharya et al., 2009] ....21
Figure 2.4: Direct approach for workflow realization [Ngamakeur et al., 2012] ..........26
Figure 2.5: ACP System [Ngamakeur et al., 2012] .......................................................26
Figure 2.6: An example of a purchasing process [Chebbi, I, et al., 2006] .......................37
Figure 2.7: Public view for a purchasing process [Chebbi et al., 2006] ..........................37
Figure 2.8: View approach to inter-organizational workflow [Jiang et al., 2010] ............39
Figure 2.9: Public view as a process-oriented contract [van der Aalst et al., 2010] ......40
Figure 2.10: MDA approach to generating process-oriented UIs [Sousa et al., 2008] ....46
Figure 3.1: An overall view of artifact-centric purchasing processes........................52
Figure 3.2: An example of lifecycle composition ............................................................60
Figure 4.1: Generic purchasing process with its two specializations .........................66
Figure 4.2: Overview of a framework for behavior-consistent process specialization....68
Figure 4.3: ACP specializations with four different specialization methods ..........69
Figure 4.4: Examples of derived lifecycles and its base lifecycle ...............................73
Figure 4.5: Examples of L-fragments ............................................................................75
Figure 4.6: An example of S-regions and SL-fragments. .............................................76
Figure 4.7: Composite S-regions of SL-fragments in Figure 4.6 .................................77
Figure 4.8: More examples of S-regions and their composition .................................79
Figure 4.9: S-region with a sub SL-fragment ...............................................................80
Figure 4.10: Sync specializations of extended artifact lifecycles ...............................86
Figure 4.11: Open-ended termination inconsistency ....................................................89
Figure 4.12: Sync refinements of existing artifacts. .....................................................90
Figure 4.13: Examples of reduced lifecycles based on sync reduction .....................95
Figure 4.14: Example of propagated sync reduction ..................................................97
Figure 4.15: Generic purchasing process .................................................................99
Figure 4.16: Offline ordering process ..........................................................................100
Figure 5.1: An overall view of artifact-centric purchasing processes ....................107
Figure 5.2: View framework for artifact-centric inter-organizational processes ....111
Figure 5.3: The local ACP model for Supplier .............................................................116
Figure 5.4: Two types of applications for the B-Consistency notion ..........................117
Figure 5.5: Example of an agreed public view .........................................................120
Figure 5.6: Abstractions for abstract transitions and abstract states ......................122
Figure 5.7: Abstraction on expansion of NAL-fragment ..........................................125
Figure 5.8: Examples of sync abstraction .................................................................128
Figure 5.9: Example of abstraction of more than two ASL-fragments ....................129
Figure 5.10: Example of abstraction on nested sub-SL-fragment ............................129
Figure 5.11: A running example for find_minASR function ....................................134
Figure 5.12: An overview of private views and view conformance ...........................140
Figure 5.13: Supplier’s private view $\Pi 1Supplier$ ......................................................141
Figure 5.14: Supplier’s modified private view $\Pi 2Supplier$ .......................................142
Figure 6.1: Automatic UI Derivation Framework for ACP-i model .......................151
Figure 6.2: A UIF Model and UIC with its artifacts ...................................................... 152
Figure 6.3: An example of artifacts in a purchasing process ....................................... 158
Figure 6.4: ACP lifecycle of the simplified purchasing process .................................. 159
Figure 6.5: The behavioral aspect of UIF-base model ............................................... 160
Figure 6.6: An example of UIF-role models for Supplier and Buyer .......................... 165
Figure 6.7: ACP execution system architecture with UIF support ............................ 166
Figure 6.8: ACP business rules format ................................................................. 167
Figure 6.9: Example of ACP business rules in XML format ................................... 168
Figure 6.10: An example of UIFML specification format ...................................... 169
List of tables

Table 3.1: Example of business rules ................................................................. 54
Table 4.1: Relations between ACP specialization methods and Sync specialization 85
Table 5.1: Example of business rules for inter-organizational purchasing process 114
List of publications

Main author


Coauthor


Chapter 1.

Introduction

We organize this chapter as follows. Section 1.1 provides an overview of artifact-centric approach to modeling business processes. Section 1.2 discusses key issues to support the modeling of artifact-centric business processes. Section 1.3 summarizes our research contributions. Lastly, Section 1.4 provides an outline of the thesis.

1.1 Artifact-Centric Approach to Modeling Business Processes

Nowadays, the shift of business strategies from the product focus to customers focus makes businesses to think how their business operations can be efficiently organized as to deliver business values to their customers and accelerate their growth. Business strategy can be operationalized and described by the mean of business process (which is a set of activities executed by various stakeholders to provide value to a customer). Therefore, business processes have significantly
turned out to be one of important aspects that leverage businesses in a completive market. Business process modeling is the method for representing a business process in an intelligible format (e.g., graphical notation). A business process model is an essential tool for businesses to formulate and reason about how desired business goals are achieved, and to serve as the communication mean among business stakeholders. In general, the process model describes how activities are performed (by organizational resources – which can be humans or automated systems) in order to achieve business goals.

In the recent years, businesses are forced by economic constraints and competitive business environment; therefore, they seek for an innovative improvement of their business operations in order to outperform their competitors. Apart from that, businesses have often grown from mergers and acquisitions which can lead them to face redundancies and inconsistencies of their operations and processes [Liu et al., 2007]. The traditional workflow modeling approach turns out to be increasingly difficult to implement as business processes become more sophisticated and large in size. This brings in challenges to the modeling, design, and management of complex and dynamic requirements of business processes. As such, a business process model should not only be used to ensure that the work can be completed as desired but also to facilitate operational innovations and cost savings. A traditional business process (or workflow) modeling approach (e.g., Business Process Modeling Notation (BPMN) [OMG, 2011], Business Process Execution Language (BPEL) [OASIS, 2005], Event-driven Process Chains (EPC) [Keller et al., 1992], and Yet Another Workflow Language (YAWL) [van der Aalst and ter Hofstede, 2005]) emphasizes on the sequencing of activities (i.e., control flows) in a process as a first-class citizen. However, it pays less attention to data aspects of business processes. Due to the lack of holistic view of the information and activity contexts, business actors and process modelers are often riveted by what should be done instead of what can be done, thus, hindering operational innovations since it is difficult to comprehend the possible effects of the sequence of processing steps on key business entities [Bhattacharya et al., 2007, 2009]. In addition, as constructing processes with sequenced activities leads to highly-
cohesive and tightly-coupled process structures, process componentization and extension are difficult to be achieved in a natural way [Kumaran et al, 2008].

In the past few years, a new approach to modeling business processes has emerged – that is artifact-centric (operational) business process modeling [Nigam and Caswell, 2003]. Rather than primarily focusing on the activity flows of a business process, the artifact-centric approach focuses on key business-relevant entities that evolve as they progress through the operations of the process. This approach enables 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, lifecycle of artifact, services (a.k.a. tasks), and associations (between artifacts and services) [Hull, 2008], as illustrated in Figure 1. Artifacts integrate both data aspects and process aspects into a holistic unit, and serve as basic building blocks in a process model. The (macro) lifecycle of an artifact is described in terms of “business stage” and the possible evolution of the artifact. The specification of how artifacts are involved and how services operate on them is described in terms of their associations, which can be expressed in declarative styles (e.g., Event-Condition-Action (ECA)-styled business rules).
In the overall picture, the artifact-centric approach has been successfully proven for providing a higher level of robustness and flexibility in describing process specification compared with traditional process-centric approaches. Since 2003, several works have been carried out for the development of meta-models, methods, tools, user-centric, and other technologies in support of the artifact-centric paradigm. The benefits of the artifact-centric approach can be summarized as follows.

—It enables rich and natural communication among various types of stakeholders about the operations and processes of a business. Therefore, human resources, time, and operating cost needed for business transformations are significantly reduced while enabling new capabilities [Bhattacharya et al., 2005, 2007, 2009; Chao et al., 2007; Cohn and Hull, 2009].

—It utilizes the SOA technology (e.g., IBM’s Web-Sphere Process Server) as high-level artifact-centric process models can be mapped to traditional workflow execution models [Liu, G et al., 2009; Cohn and Hull, 2009; Narendra et al., 2009; Xu et al., 2011; Li and Wu, 2011; Ngamakeur et al., 2012].
—It serves as an organizing foundation for related BPM capabilities, such as business rules [Liu et al., 2007; Gerede and Bhattacharya, 2007; Gerede and Su, 2007; Kumaran et al., 2008; Deutsch et al., 2009; Fritz et al., 2009; Zhao et al., 2009; Cangialosi et al., 2010; Damaggio et al., 2011; Hariri et al., 2011], the development of user interfaces [Sukaviriya et al, 2007; Yongchareon et al., 2010], business intelligence and monitoring [Liu, R et al., 2011; Fahland et al., 2011], design and reuse [Lohmann, 2011, Yongchareon et al., 2012].

1.2 Key Issues in Supporting Artifact-Centric Process Modelling

As introduced, the artifact-centric approach provides significant benefits to the modeling and management of business processes. Several existing works have been carried out to provide and formulate ways to model business processes in an artifact-centric paradigm, thus, showing a promising trend in business process management and its state of the art. However, there is still a need for more research in various dimensions to emphasize the importance, value, and wider usefulness of this modeling approach. Particularly, in this thesis, we identify three key challenging issues for the support of modeling artifact-centric business processes: (1) support for business process reuse, (2) support for efficient inter-organizational business process collaboration, and (3) support for process visualization and user interfaces development. We separate the discussions for the motivations of these issues in the following sub sections.

1.2.1 Business process reuse

Due to the growth and expansion of businesses, business process requirements from different customers, government regulations, outsourcing partners, etc., result in frequent changes and revision to business processes. With business processes becoming more complex and dynamic, reusability of business processes is highly sought after to improve process modeling efficiency. In this background, organizations strive for a more efficient and systematic approach to flexibly define and extend their business processes. In object-oriented software reuse, several benefits have been shown in practices 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 on-demand customization and extension of existing business processes.

Artifact-centric approach to modeling business processes naturally lends itself well to both object-orientation and service-orientation design principles, as it focuses on the design of both business artifacts involved in a process and services performing operations on such artifacts. Owning to the object-oriented nature, the artifact-centric model supports higher level of flexibility, extensibility, and reusability. 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, 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 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 the 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 needs to be considered to ensure that the process can be consistently observed from its base process, therefore, allowing processes to be reported at different levels of generality and to be compared across the specializations [Hull, 2008].
1.2.2 Inter-organizational business process collaboration

Nowadays, business processes become more cooperative and span across organizational boundaries. One of the key challenges is how the artifact-centric approach can be realized and used for facilitating inter-organizational business process collaboration.

The growing number of scenarios for inter-organizational cooperation has been demonstrated in several industry practices and it is stimulated by the advanced business process management technologies and Service-Oriented Architecture (SOA) that provide and foster the more effective and efficient communication and collaboration between organizations. Particularly, SOA enables the business collaboration across organizations by the means of web service composition to achieve the mutual goal of collaboration as well as the individual goal of each participant. Despite the beneficial support from the current technologies, some important issues are raised when organizations try to initiate, design, plan, and manage the cooperation. As they want to be autonomous as much as possible in the collaboration and the mechanism to promote the trustworthiness among participants is unclear, privacy and security of their internal information/processes is their major concern. Organizations attempt not to reveal their private or sensitive information to others and prefer high level of autonomy when participating in the collaboration. As such, organizations seek for efficient approaches for the design, modeling, implementation, and management of inter-organizational business process collaboration [Liu et al., 2008]. Recently, literature [Desai et al., 2009; Ghattas, 2009; Montali et al., 2010; van der Aalst et al., 2010] showed that the efficient and desired means of coordination among participating organizations in a service-oriented collaboration factors in three major requirements: compliance, flexibility, and autonomy.

—**Compliance** refers to the expectation that all parties must provide the services as they have mutually agreed in the collaboration commitment, e.g., a form of contract.

—**Flexibility** means that each party should have a preferred level of freedom to change and implement its own part in the collaboration.
Autonomy refers to the level of isolation of an organization that can participate in the collaboration with the minimal requirement of revealing its own private information (or process) to other parties.

Although service choreographies intend to provide a global view of the collaboration and allow each party to design and implement its own portion, the choreography modeling approach and languages (e.g., WS-CDL [W3C, 2005]) mainly describe the collaboration from the procedural perspective, and focus on control-flow, message sequencing, etc. As such, the degree of flexibility and autonomy is still limited by the nature of current choreography modeling.

In artifact-centric approach, flexibility is naturally achieved as it is deemed as one of the benefits of artifact-centric models. However, a comprehensive study on supporting organizations to achieve all the three collaboration requirements is still missing. Based on literature and practices, we observed that the existing view-based solutions for inter-organizational business process management can provide a promising and efficient way of modeling and change validation to address such requirements (e.g., in [Van der Aalst and Weske, 2001; van der Aalst et al., 2010]); nevertheless, apart from the artifact-centric hub, which utilizes the database view concept in the context of artifacts, presented in [Hull et al., 2009], there is no comprehensive study in the context of artifact-centric inter-organizational business processes.

Process views can provide a coordinating mechanism and privacy policy to ensure that the privacy of process information of each organization in the collaboration is preserved as much as possible. In other words, the privacy of business artifacts which evolve through the processes across the organizational boundaries should be aware of. We believe that two kinds of artifacts, which are local artifacts and shared artifacts, need to be considered in an artifact-centric inter-organizational process. The local artifacts need more strict privacy policy, while shared artifacts may have relaxed privacy policy as they are used to transfer and coordinate between organizations. How these two kinds of artifacts are identified requires a further study, especially for the shared ones. In the coordination, we observed that at a particular state a message exchanged between
organizations reflects the attribute changes of the artifacts used in an organization’s internal process. Some processing steps of the local artifacts of each individual party may link with the global constraints in the collaboration. To address these issues, the idea of process views should be exploited from an artifact-centric perspective to help organizations to meet the three major collaboration requirements (compliance, flexibility, and autonomy) of modeling of artifact-centric business processes and validating process changes in an inter-organizational collaboration environment.

1.2.3 Process visualization and user interfaces development

Unlike a conventional activity-centric approach, in the artifact-centric approach, expressing business process logic by using declarative-style business rules makes process modelers difficult to see and understand the structure and flow of the process, especially when a business process is distributed in cross-organizational environment. In the user-centric aspects, this brings in a key challenge to have a natural approach for representing such declarative process models to non-technical stakeholders and users of the processes [Hull, 2008]. There should be a more natural way to support them to better comprehend artifact-centric process models from both process flow and data perspectives.

As we know that user-interfaces (UIs) are served as a media bridging between systems and users, and the systems run based on its underlying processes. By having a closer look at artifacts and user-interfaces, we can see that on the one hand a user-interface is used for users to view/input business artifact data and invoke a related function that will affect to the process; on the other hand, the artifact data and business rules decide which user-interface should be brought for the users to fulfil a certain procedure of the process. Therefore, the data inputs and navigation flows of UIs can be used as a media for representing (or visualizing) an artifact-centric process model in a task-based/control-flow structure with data perspective. Apart from the representation benefit, UI designers or application developers can use these UIs for generating actual UIs (e.g., web pages) for artifact-centric business processes. As such, we can see that the benefit of generating
artifact-centric UIs is twofold: visualizing business processes and supporting the
development of process-oriented UIs.

A (semi or fully) automatic generation of UIs is one of the features of workflow-based applications provided to support the execution of human tasks in a business process. However, these UIs are designed and developed when workflows are modeled at some abstract level, thus, causing an issue of loosely coupled alignment between business processes and user interfaces, i.e., changes of business processes that impact on UIs are to be managed in an ad-hoc manner [Sousa et al, 2008]. One solution to this issue is to develop UIs based on existing Model-Driven Architecture (MDA) approaches (e.g., OOWS [Torres and Pelechano, 2006]) for the semi-automatic generation of user interfaces by deriving task-based models from activity-centric business process models. Sukaviriya et al. [2007] initiated a study on a MDA-based generation of UIs with the consideration of business artifacts. However, there is no study on how to automatically generate UIs from an artifact-centric process model, where the set of generated UIs needs to present both control and data perspectives of a business process. Due to the distribution of process specification across organizations, the UIs’ generation should also support different roles of users in different organizations that participate in business collaboration. In addition, the generated UIs should be further customized by UI modelers without a concern of the integrity of business logic, and changes of data requirement specified in the model should be reflected on UIs.

1.3 Research Contributions

In this thesis, we study the artifact-centric approach to modeling business processes, and propose novel approaches to tackle the three key issues for supporting artifact-centric business process modeling as introduced in Section 1.2. We summarize our contributions of the thesis in this section. Section 1.3.1 presents our contribution of the support for business process reuse. Section 1.3.2 presents our contribution of the support for efficient inter-organizational business process collaboration. Lastly, the contribution of the support for process visualization and user interfaces development is summarized in Section 1.3.3.
1.3.1 Support for business process reuse

We propose a novel framework for specializing Artifact-Centric Process (ACP) model, which comprises several dependent artifact classes and three construction methods for process specialization: artifact refinement, artifact extension, and artifact reduction. We study the dynamic behavior of artifact class and the interaction (via synchronization dependency) behavior of artifacts in a process, and formulate a set of rules that is used to preserve the behavioral consistency, called $B$-consistency, between a specialized process and its base process. Particularly, we present our fragmental behavior analysis technique with supporting theorems to check the interaction behavioral consistency between synchronized fragments of artifacts in the specialized process and in the base process. With our notion of specialization, process modelers can reuse existing artifact-centric process models in both artifact level and process level. All different specialized process models of a particular generic process model can be consistently observed at different level of specialization hierarchy, thus, allowing both artifact and process instances of a specialized process to be aggregated and reported at the more generic level.

1.3.2 Support for efficient inter-organizational business process collaboration

We propose a novel view framework for artifact-centric inter-organizational business processes. The framework comprises artifact-centric model for inter-organizational business processes called ACP-i model, the notions of public views and private views, and the view consistency rules. An ACP-i model is used to capture the specification of a complete inter-organizational process. A public view can be used to serve as an agreed contract of a global process (defined in ACP-i model) at the more abstract level. A private view is used to capture an organization’s local process and to permit internal changes of the local processes while ensuring that local changes preserve the correctness and behavioral consistency of the global process. Based on our $B$-consistency notion, we propose mechanisms to construct a behavior-consistent public view and an algorithm that helps organizations to automatically find the most abstract public view based on their local processes. The most important thing is that our public/private view
approach permits process verification to be achieved locally, thus, avoiding the state exposition problem that can occur in the global verification.

1.3.3 Support for process visualization and user interfaces development

We propose a *User Interface Flow (UIF)* model and algorithms to automatically derive a *UIF model* from an underlying *ACP-i model* with a view-based support (for different roles of users). The UIF model is a conceptual model used to represent the logical structure of user interfaces (e.g., pages, forms, and input fields that need to be filled by users) and their navigation flows (from a form in one page to another page). It can be used for visualizing business processes as well as for facilitating the user-centric model-driven development and implementation of concrete user interfaces for business processes. It provides an intuitive process structure, likewise an activity (control-flow) structure, for artifact-centric models, which helps process modelers and non-technical stakeholders who are interested in the process understand the flow of a process easily. On the other hand, it supports role-based settings, thus, allowing different roles to see and navigate through only their accessible user interfaces based on the specification of a process. In addition, this work complements the MDA-based approach for developing and generating actual user interfaces. We serialize a UIF model in XML format, and therefore it can be used to generate actual HTML webpages via the use of XSLT together with CSS design templates.

1.4 Thesis Outline

In this section, we describe the overall structure of the thesis as follows.

Chapter 2 presents the necessary background for the remainder of the thesis and gives a general overview of existing related works. Some references to related work are given in the later chapters to provide a detailed comparison of our work to the state of the art in more details.

Chapter 3 discusses an artifact-centric approach to modeling business processes. In this chapter, we present a simple Artifact-Centric Process (ACP) model that will
be used as a base for the discussion of the three kinds of supports we proposed for artifact-centric process modeling. The discussion on the behavioral aspect of a process and artifacts involved in a process is provided this chapter.

Chapter 4 presents a framework for the specialization of ACP models. The specialization methods are proposed for the construction of specialized processes from existing processes. The notion of behavioral consistency, $B$-consistency, is developed for checking whether the specialized process is consistently observable from its super process where synchronization dependencies between artifacts are taken into account.

Chapter 5 proposes a view framework for artifact-centric inter-organizational business processes. It addresses the three collaboration requirements based on the notions of public and private views. The public view construction and validation techniques are also discussed with the automatic construction algorithm.

Chapter 6 presents a model for user interfaces called UIF model, and algorithms to automatically derive a UIF model from an underlying process model. The UIF model is used for the purpose of visualizing artifact-centric business processes as well as supporting the step-wise MDA-based development of actual user interfaces.

Chapter 7 summarizes our contributions made in the thesis and discusses future works.
Chapter 2.

Literature Review

In this chapter, we present the necessary background for the remainder of the thesis and discuss related work. We begin with the background of artifact-centric approach to modeling business processes in Section 2.1. Then, we classify the rest of related work discussions into three main categories corresponding to the three major research issues to be addressed in this thesis in supporting artifact-centric business process modeling. Section 2.2 discusses existing approaches to reusing business processes. Section 2.3 discusses approaches to facilitating inter-organizational business process collaboration. Section 2.4 discusses approaches to deriving user interfaces for business processes.
2.1 Artifact-Centric Approach to Modeling Business Processes

2.1.1 Business artifacts

Nigam and Caswell [2003] were the first to introduce the concept of modeling business processes from business artifacts and their lifecycles. A business artifact (or artifact for short) is defined as a concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business. Artifacts can be referred as business records and have to be recognizable, i.e., to contain information in one place, and are taken to be the only explicit information contained in the business, i.e., a set of business records that represents the content of the business.

"Business artifacts constitute concrete information chunks that the business creates and maintains. In other words, business artifacts provide the mechanism for information localization. [Nigam and Caswell, 2003]"

An example of key business artifacts in a purchasing process is a Purchase Order (PO) which is used to specify what goods a buyer wants to purchase from a seller. The PO may contain several good items and each item can be ordered in a specific amount. The total cost of the goods purchased, purchasing time, and other information may be recorded (in separate fields) in the PO if they are used in the business. The PO should be assigned with a unique identification (ID) for the reference of itself to other related artifacts, e.g., Invoice, Shipping Order, etc.

Technically in the (object-oriented) database area, artifacts correspond to complex objects having structured data entities and their relations. In fact, an Entity Relationship (ER) model [Chen, 1995] can be used for presenting artifacts and their attribute relations in the data perspective. An example of ER model for business artifacts used in a purchasing process is illustrated in Figure 2. Based on a business narrative and ER model of information used in a business, we can derive key artifacts. Not all entities drawn in the ER model are considered as artifacts. If an entity is weak and has one-to-many relation(s), then it may not be considered as
an artifact but as a part of a strong entity that it has the relation with. For instance, the *OrderItem* entity can be structured as a part of the *Order* artifact.

![Figure 2: ER Model for entities used in a purchasing process](image)

Apart from the relational model, Extensible Mark-up Language (XML) [W3C, 2008] can also be used to represent the data structure of artifacts with the Data Type Definition (DTD). Recently, the work on Active XML (AXML) [Abiteboul et al., 2008, 2009; Marinoiu et al., 2010], which is an extension of XML with embedded service calls, has been proposed and can be used for supporting business artifacts and their processing in the SOA environment. Their AXML system specifies a set of interacting AXML documents embodied with declarative specifications based on constraints on the evolution of artifacts which can represent a process that evolves in time.

### 2.1.2 Artifact lifecycle and dependency

We can describe the operations of a business by considering how artifacts are processed. The lifecycle of an artifact captures the end-to-end processing of the artifact from creation to completion and archiving. In [Nigam and Caswell, 2003], to describe how artifacts are processed, two constructs are required: tasks and repositories. Tasks provide a localization of function and execute read/write/update operations on artifacts. Repositories provide a means for storing and retrieving artifacts. In this thesis, we consider only tasks that perform some function (i.e., operation) on artifacts, not the repositories. We assume that whenever artifacts are
required to be created/read/updated, they can be immediately stored or retrieved from accessible repositories.

When describing the lifecycle of an artifact in a business process, many tasks may need to read or modify data of other artifacts where these artifacts have their own lifecycles. Reference to another artifact by a particular task creates a dependency between artifacts that such task performs on. If a task performs a modification on several artifacts, the synchronization dependency between these artifacts occurs in a business process.

A Labeled Transition Systems (LTSs) is one of the behavioral representations that can be used to describe and formalize a lifecycle of artifact. An LTS consists of a set of states and a set of transitions between those states. These transitions are labeled by actions. One state is designated as the initial state. However, there can be multiple final states. We can describe the dependency between two LTSs as follows: a synchronization dependency between two LTSs occurs when a transition of one LTS and a transition of another LTS are fired simultaneously (on the same action). For both LTSs, the source state of each LTS changes to the target state via the transition that is fired.

Figure 2. illustrates an example of LTSs of business artifacts in a purchasing process. We can see that the \( R.\ create\text{Shipment}(a,s) \) task updates on both the Order \( o \) artifact and the Shipment \( s \) artifact and changes the states of both artifacts (from \textit{ready\_for\_shipping} to \textit{in\_shipping} of the Order artifact and from \textit{init} to \textit{open\_for\_shipItem} of the Shipment artifact).
LTSs can be verified by using LTS Analyzer (LTSA) developed by [Magee and Kramer, 2006]. The LTSA performs compositional reachability analysis to exhaustively search for violations of the desired behavioral properties.

### 2.1.3 Process design methodology

Modeling business processes based on artifact-centric approach shows some level of similarity compared with case management [Van der Aalst and Weske, 2005], document-oriented workflows [Wang and Kumar, 2005], product-based workflows [Muller et al., 2008], and business objects approach [Nandi and Kumaran 2005; Redding et al., 2008, 2009, 2010]. These approaches focus on evolving business entities and the support for the management of activities in an ad hoc manner.

An existing work in [Maamar et al., 2010] discussed how to derive artifacts out of business requirements based on data and operation perspectives. On the one hand, Küster et al., [2007] proposed an approach for generating business process models from object lifecycles. On the other hand, Kumaran et al. [2008] and Cabanillas et al. [2011] discussed data-centric views of conventional activity-centric processes. By analyzing key business artifacts or data objects involved in a business process, the corresponding artifact lifecycles can be (automatically) generated. Uchitel et al., [2003] studied on the synthesis of state machines from scenarios where various
scenario specifications are used to generate state machines for all involving objects. Liu et al. [2007] formulated nine operational patterns in artifact-centric operational models and transformed them into colored Petri Nets for reachability analysis. Fritz et al. [2009] studied the technical problem of goal-directed declarative artifact-centric workflow construction with the concern of general setting, design time analysis, and the synthesis of workflow schemas from goal specifications. Lohmann [2011] presented an approach to automatically constructing business process models based on policies (interdependencies between artifacts) and compliance rules, which results in reduction of compliant behavior checking in terms of reachability of final states. An artifact composition technique presented in [Lohmann and Wolf, 2010] was used in this approach for the compliance checking.

Based on the above existing works, we can have an insight understanding of how business artifacts and tasks are involved in a business process. The three-layered framework for artifact (data)-centric design and modeling methodology for business processes was proposed by Bhattacharya et al. [2009], as shown in Figure 2.. The methodology is centered on the data being manipulated as a business is managed.

Figure 2.: Three logical levels of Artifact-Centric BPM [Bhattacharya et al., 2009]

—Business Operations Model (BOM) provides a detailed logical specification of business process execution – this includes services specified in terms of their semantics (e.g., pre-conditions and conditional effects) and ECA rules. Note that a “service” term is used instead of a task due to the need to emphasize that there is a close correspondence between services in artifact-centric paradigm and semantic web services.
Conceptual Flow captures the BOM in a procedural manner while concealing implementation details, which is suitable for optimization.

Workflow is a process execution system that allows executable services to communicate with each other through messages and artifacts that are manipulated by such services.

In this thesis, we only focus on the (model) specification level of a business process with a particular level of consideration on how the specification can be mapped to the conceptual flow and how the conceptual flow can be analyzed. Specifically, we study the behavior of a business process and its involving artifacts that are derived from the specification.

2.1.4 Modeling and model verification approaches

There are several existing works proposed for modeling and verifying artifact-centric business processes.

Gerede and Su [2007] presented a formal model named ABSL based on Computational Tree Logic (CTL) language [Emerson, 1990] for artifact-centric business process specification. Artifact classes are represented by adding object-oriented classes with states, and guarded finite state automata is used to capture the logic of entities that carry out the work in a business model. Similarly, Bhattacharya et al. [2007] focused on the evolution of business process logic by using semantic web services (in the spirit of OWL-S [OWL Services Coalition, 2003]) to model activities and a set of business rules to declaratively capture and represent a business model with the study of necessary properties such as reachability of goal states, absence of deadlocks, and redundancy of data.

Zhao et al., [2009] proposed TiLe language (a subset of Linear-time Temporal Logic (LTL) with Past) for specifying constraints for artifact-centric processes with complexity results for the satisfiability of a set of constraints. Similarly, Damaggio et al. [2011] studied on the automatic verification of artifact-centric business processes based on their earlier works presented in the International Conference on Database Theory (ICDT) [Deutsch et al., 2009; Damaggio et al., 2011] which
addressed the verification problem for artifact systems (with arithmetic and data dependencies) by statically checking whether some/all runs of the system satisfy desirable properties expressed in a Linear-time Temporal First-Order Logic (LTL-FO).

Cangialosi et al., [2010] proposed a formal model for a family of artifact-centric services based on the notion of conjunctive queries to define preconditions and effects of services. They considered the current state of data and their state after the execution of a task as two databases related through a set of tuple-generating dependencies (defined in [Abiteboul et al., 1995]). The data schema of artifact is a full-fledged relational database. The lifecycle schema of artifact is specified as a set of condition-action rules, where both pre-conditions and post-conditions are expressed as conjunctive queries. Their formalism, which is quite expressive and inherently infinite state, is decidable under a weak-acyclicity restriction [Fagin et al., 2005] on the form of the tuple-generating dependencies expressing the effects of actions. Similarly, Hariri et al., [2011] studied relational artifacts, where data are represented by a fully-fledged relational database, and a lifecycle is described by a temporal, dynamic formula expressed in $\mu$-calculus [Luckham et al., 1970; Emerson, 1996]. Compared with the work presented in [Cangialosi et al., 2010], this work used negation and more generally full first-order queries in defining the pre-conditions of actions in which the theory of conjunctive queries and homomorphism is ignored.

Apart from the above works, there are some works with a particular focus on defining the lifecycle of artifacts by using state transitions as a basis. ArtiNet model [Kucukoguz and Su, 2010] and Guard-Stage-Milestone (GSM) approach [Hull et al. 2010, 2011; Damaggio et al., 2011] have been introduced for specifying business artifact lifecycles and their constraints.

ArtiNet model was developed based on declarative specifications/constraints (expressed in LTL) of artifact lifecycle in the spirit of DecSerFlow [Van Der Aalst and Pesic, 2006]. In ArtiNet, the notion of lifecycle is formulated based on the set of all possible paths that an artifact can navigate through, and two technical problems are addressed. (1) Does a given workflow (schema) contain only lifecycle allowed by
a constraint? And (2), is it possible to construct a compliant workflow from a given lifecycle specification (constraint)? Problem (1) is decidable when workflow is atomic or constraints are regular. For problem (2), a workflow that satisfies the constraint can be always constructed and sufficient conditions where atomic workflows can be constructed.

The GSM model was developed in a state-machine style based on the Business Entities with Lifecycles (BEL) framework used in IBM's Service Oriented Modeling and Architecture (SOMA) method [Strosnider et al., 2008]. The behavior specification of GSM models can be declaratively expressed in terms of ECA-like rules with the support for parallelism within artifact instances and modularity through hierarchical constructs, thus, supporting a basis for formal verification and reasoning. A GSM's lifecycle model is specified by using stages, where each stage consists in one or more milestones, a stage body, and guards. Nesting of stages is supported, thus, allowing parallel execution of stages at the same nesting level.

The GSM model is intuitive and actionable as the direct mapping from the specification to an implementation of running systems is reasonably achievable. GSM models can provide a well-support to managing business operations in highly prescriptive approaches (e.g., BPMN and EPC) and highly descriptive approaches (e.g., case management systems [Van der Aalst and Weske, 2005]). Several constructs defined in conventional process-centric approaches can be naturally simulated and included into GSM models. A prototype engine, called Barcelona, which is an extension of the Siena system [Cohn et al., 2008], provides a simple graphical design editor, and captures the GSM models directly into an XML format.

As stated in [Hull et al. 2010], current research carried out based on the GSM meta-model includes the followings.

— a framework and tool to support GSM models for business-level stakeholders, in which they can specify business scenarios and other intuitive, imprecise, incomplete specifications

— tools to capture the specification of guards, milestones, and derived attributes based on Semantics of Business Vocabulary and Business Rules (SBVR) [OMG, 2008]
—formal foundations and algorithms for verification and reasoning

In our work, we do not focus on a declarative-styled modeling language to address the three issues for supporting artifact-centric process modeling. As previously mentioned, we use conventional modeling approach based on LTS, which is comparable to state machine-based model used in the GSM models, to model the behavior-based specification of an artifact-centric process.

2.1.5 Workflow realization

In the activity-centric approach, Ouyang et al. [2009] proposed a technique to translate BPMN to BPEL for the workflow implementation in SOA. However, artifact-centric models require different realization technique. Here, we discuss two different approaches for the realization.

The first approach is a model-to-model transformation-based realization. As discussed in [Cohn and Hull, 2009], artifact-centric models (which are already operational models) can be transformed into conventional workflow models; therefore, the models can be implemented on existing workflow management systems. It is desirable that the workflow models should support constraint-based specification (e.g., [Pesic et al., 2009]) or rule-based languages (e.g., OMG’s SBVR). Model transformation techniques presented in [Küster et al., 2007; Redding et al, 2008] for generating process models from object lifecycles can be used for this purpose. Liu et al. [2010] proposed ArtiFlow models to capture artifact-centric processes and the mapping from ArtiFlow to BPEL. A similar work for artifact-centric model to BPEL translation was presented in [Li and Wu, 2011].

Cohn and Hull [2009] provided a discussion on IBM BELA tool that can be used to support the mapping from artifact model to an executable workflow that can run on IBM’s WebSphere Process Server. The advantage of this approach is an ease of implementation as workflow technologies and standards based on the traditional model have been developed. However, it is argued in [Ngamakeur et al., 2012] that the model conversion approach has several drawbacks as the transformation, which is unidirectional, poses loss of information (depending on the target model’s
specification) and degrades the run-time flexibility and monitoring ability as the reverse conversion is required.

In contrast to the first approach, Ngamakeur et al. [2012] proposed the realization approach to directly execute artifact-centric models in the SOA without model conversion in order to overcome the drawbacks of the first approach. The overview of this approach is illustrated in Figure 2. with the ACP system architecture presented in Figure 2.. Their system is developed and equipped with Drools business rule engine [JBoss, 2012]. The process and artifacts schema are stored in the XML format, which can be retrieved an executed directly into the system. The ACP system supports Web Service Definition Language (WSDL) [W3C, 2001] to allow external web-services calls for interacting with a running process.

Figure 2.: Direct approach for workflow realization [Ngamakeur et al., 2012]

Cohn et al. [2008] presented Seina prototype that allows users to model business artifacts and process as XML documents in order to create a composite web application to be deployed and executed on an execution engine. The Siena includes
a user interface for designing artifact-based workflow schemas (based on state-machine models), a capability to represent the workflow schemas as an XML file along with XSDs (XML Schema Definitions) [W3C, 2004] for holding the artifact information models, and an engine that directly executes the XML file upon incoming events and performing tasks. Siena provides REST (Representational State Transfer) and WSDL interfaces to external parties that want to access artifact values and move artifacts along their lifecycle. In addition, it provides the capabilities of sending notifications and showing artifact instances to users. Siena schemas can be specified from Microsoft PowerPoint.

Based on the above existing works, artifact-centric process models are feasible to be implemented on the current workflow technologies. However, we observe that the direct realization approach is more appropriate as the detailed lifecycle of artifacts (expressed by business rules) should be captured in the system to enable monitoring, tracking, and verification at run-time. The artifact-centric monitoring framework [Liu, R et al., 2011] and the artifact-centric process conformance checking [Fahland et al., 2011] were proposed for capturing and validating running instances in the context of artifacts and the process they are involving with, thus, supporting the run-time management of artifact-centric processes.

2.2 Business Process Reuse

In the context of business process, 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 thesis is based on the inheritance (or specialization) concept, we acknowledge other related works that aim at supporting the reuse of business processes. Section 2.2.1, 2.2.2, and 2.2.3, discuss existing works based on workflow inheritance, configurable process models, and design by selection and patterns, respectively.
2.2.1 Inheritance of workflows

Business process design and modeling in today has 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 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. Given two workflow nets $x$ and $y$ such that $x$ inherits $y$, the following inheritances are defined.

(1) *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.

*If it is not possible to distinguish $x$ and net $y$ when arbitrary tasks of $x$ are executed, but when only the effects of tasks that are also present in $y$ are considered, then $x$ is a subclass of $y$ with respect to projection inheritance.*

(2) *Protocol inheritance* is defined based on encapsulation.

*If it is not possible to distinguish $x$ and $y$ when only tasks of $x$ that are also present in $y$ are executed, then $x$ is a subclass of $y$.*

(3) *Protocol/projection inheritance.*

*If $x$ is a subclass of $y$ with respect to protocol/projection inheritance, then $x$ is a subclass of $y$ with respect to protocol inheritance and projection inheritance.*

(4) *Life-cycle inheritance* is either a protocol and/or projection inheritance.
Similarly, Schrefl and Stumptner [2002] studied the consistency criteria of the inheritance of object life cycles based on refinement, extension, and deletion of states/activities in the lifecycle. They distinguished weak and strong invocation consistencies and observation consistency, and then proposed necessary and sufficient rules for checking behavioral consistency between object lifecycles. Their work corresponds to the inheritance notions proposed in [Basten and van der Aalst, 2001]. Harel and Kupferman [2002] argued that object-oriented system design should incorporate a concept of behavioral inheritance for classes in such way that a system refinement should either preserve trace inclusion or simulation for the language built by a system's protocol – which can be done by using the inheritance notion in [Van Der Aalst and Basten, 2002].

At similar time, 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) for workflow processes which can be used to tackle four problems: 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 evolitional 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 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] 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. Their approach can be used as a method for categorizing and analyzing processes. Their process specialization definition is compatible with the traditional notion of specialization previously discussed.

Recently, Weidlich et al. [2010] introduced the notions of projection and protocol compatibility of correspondences between process models to decide the compatibility of two business process models based on behavior inheritance. Later, Weidlich et al. [2011] proposed the concept of a behavioral profile that captures the essential behavioral constraints of a process model in contrast to the existing notion of trace equivalence and consistency measures. The profile allows the quantified measurement of differences between processes ranging from 0 to 1.0.

All of the above works focused on the inheritance of single object lifecycle or 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. For artifact-centric business processes, specializations should not only apply on each individual artifact but also on their interactions. Some works have initiated the study of object lifecycles and their interactions within (or between) business processes in various areas, e.g., process adaptation and dynamic changes [Muller et al, 2008], design compliance [Küster et al, 2007; Lohmann, 2011], scenario-based specification [Uchitel et al., 2003], conformance checking [Fahland et al, 2011], and
service contract for inter-organizational business processes [Van Der Aalst et al., 2010]. However, a specialization mechanism that takes into account the interactions of objects and the guarantee of behavioral consistency between a specialized process and its base process brings in technical challenges and requires further study.

### 2.2.2 Configurable process models

The design of business process models 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 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 identifying 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
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 each 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 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.

2.2.3 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 chain 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 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 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 relation and dependencies among them needs further investigation.

In artifact-centric setting, Calvanese et al. [2009] addressed the problem of comparing artifact-centric workflows by proposing a notion of dominance between workflows that captures the fact that all executions of one workflow can be emulated by another workflow. However, their work focused on the initial and final snapshots of the workflow execution to be compared and did not take the behavior of artifact and process into account.
Chapter 2. Literature Review

2.3 Facilitating Inter-Organizational Business Process Collaboration

In this section, we discuss existing works related to the support for efficient inter-organizational business process collaboration. Firstly, Section 2.3.1 reviews process view works carried out to support conventional activity-centric business processes in the cross-organizational context. Then, we discuss existing approaches in the context of artifact-centric business processes in Section 2.3.2. Lastly, Section 2.2.3 reviews existing specification languages and standards that support inter-organizational business processes.

2.3.1 Conventional activity-centric process views approaches

There are several works showing the significance of using process view approaches to facilitate the modeling and management of inter-organizational business processes. Initially, van der Aalst and Weske [2001] introduced the notion of public and private views with the discussion of how these views can help the coordination in a dynamic collaboration. The terms used in public/private view approach are defined as the following.

—A total workflow is a whole inter-organizational business process as it is actually executed.

—A public workflow is a business process that all involving partners agreed on, which only consists of tasks that are of interest to all of them. The public workflow is an abstraction of the total workflow.

—A private workflow is a part of the total workflow that belongs to a particular partner.

Process views are used to protect the privacy of collaborating organizations while maintaining high level of flexibility and autonomy. Later, van der Aalst and Basten [2002] formulated the inheritance-preserving transformation rules to support private process changes. The rules can be used to guarantee that the behavior of a public process is not affected by local extensions and modifications. This means that the execution of the overall process is actually consistent with the execution of the
process agreed by all parties in a collaboration in the first place. The design of inter-organizational business process can be achieved based on the following four steps.

(1) Design public workflow process.

(2) Partition the public workflow process definition amongst business partners

(3) Create a private workflow for each business partner

(4) Modify private processes using inheritance preserving transformation rules

Influenced by the concept of public/private views introduced above, there are several follow-up works carried out to support them in both theoretical and practical aspects, e.g., in [Liu and Shen, 2003, 2004; Schulz and Orlowska, 2004; Chiu et al., 2002, 2004; Chebbi et al., 2006; Lin, 2007; Eshuis and Grefen, 2008; Zhao et al., 2005, 2006, 2008, 2011; Yongchareon et al., 2010; Jiang et al., 2010; Eshuis et al., 2011].

Figure 2. shows an example of an inter-organizational purchasing process, and Figure 2. illustrates the public view as (global) coordination means for the process. All partners participating in the process must agree and provide their services based on the public view towards the completion of the process. The detailed internal process of each party is not disclosed to the others.
Chapter 2. Literature Review

Figure 2.: An example of a purchasing process [Chebbi, I, et al., 2006]

Figure 2.: Public view for a purchasing process [Chebbi et al., 2006]
Zhao et al. [2005] proposed a relative workflow framework. Visibility constraints are defined on internal process models to derive partner-specific workflow views. In the framework, each partner combines the workflow views of its partner with its internal process into a relative workflow model. They also proposed a tracking mechanism that enables an organization to track other organizations for its involved parts of collaborative business processes, and allows different organizations to track the same collaborative business process they involve with [Zhao and Liu, 2006].

Ye et al. [2009] studied the analysis of atomicity properties for a set of interacting public process views that use atomicity spheres [Schuldt et al., 2002], and proposed to use axioms from process algebra for the atomicity-preserving construction of public process view from a private process.

Jiang et al. [2010] presented a process view framework based on Timed Colored Petri Net (TCPN) to manage cross-organizational workflows, as overviewed in Figure 2. The framework includes a formal definition, a mapping from TCPN workflow models to process-view workflow models where control flow and data flow are considered together, and (collaborative) execution mechanisms of inter-organizational process instances. They built a hybrid Peer-to-Peer (P2P) based decentralized workflow management system on the top of the open JXTA platform [Sun Microsystem, 2004] to support their view approach providing a flexible and scalable architecture for inter-organizational process management.
Recently, Eshuis et al. [2011] proposed transactional process views by the construction from an internal business process annotated with a support for specification of transactional properties, including nested transactions and chained transactions. Their approach provides a robust and reliable public process behavior, thus facilitating trustworthy, fine-grained collaboration between organizations.

In contrast to the above works which have been developed in orchestration manner, there are several works carried out in a service choreographies perspective.

van der Aalst et al. [2010] proposed to use a public view as a service contract based on open nets (a special class of Petri nets) for choreographic collaboration along with the notion of private/public views accordance and operating guidelines [Massuthe and Schmidt, 2005; Lohmann at el., 2007]. Their overall idea is demonstrated in Figure 2. Their correctness criterion defined as a weak termination can guarantee that the overall process will terminate properly, i.e., a deadlock or livelock will never occur. Their accordance notion can be checked locally guaranteeing that the overall process always satisfies the weak termination property.
Decker and Weske [2007] proposed a unified framework for behavioral compatibility and consistency of services in choreography-driven settings. Consistency checking is used to guarantee that an implemented service is always compatible with other services as to avoid deadlocks and guarantee proper termination of the overall process. They showed that interacting partners only need to agree on a suitable compatibility instead of the consistency relation as it can be determined whether a consistency relation is optimal for the chosen compatibility notion.

All of the above existing works have been intensively studied in the context of traditional activity-centric business processes. However, compared with those works in conventional activity-centric approaches, the research in the area of artifact-centric inter-organizational business processes towards the support of efficient and flexible collaboration in an artifact-centric paradigm is still in its infancy stage. The technique for process view construction and validation needs to be explored and reconsidered in the artifact-centric context. Technically, the mechanisms to consistently construct and validate views for artifact-centric processes are seen to be more complicated due to the interaction relations (i.e., synchronization dependencies) of artifacts that are involved in a business process.
2.3.2 Artifact-centric view approaches

There are some works that initiated the study of how views can be used in the artifact-centric process modeling paradigm based on orchestration and choreography aspects.

The initial attempt has been done by using artifact-centric interoperation hub to facilitate inter-organizational workflows (in orchestration perspective) of multiple autonomous stakeholders who have a common goal [Hull et al., 2009]. The hub provides a centralized, computerized rendezvous point, where stakeholders can access data of common interest and check the current status of an aggregate process. They proposed three types of access restriction for stakeholders, namely window, view, and CRUDE (Create-Read-Update-Delete-Append). Window provides a mechanism to restrict which artifacts a stakeholder can see. View provides a mechanism to restrict what parts of an artifact a stakeholder can see. CRUDE is used to restrict the ways that stakeholders can read and modify artifacts. Their generic interoperation hub capability has been built on top of the Siena prototype system [Cohn et al., 2008] to support the following functions.

— a capability to specify organization roles
— a view of snapshots provided to participants to reflect the condensation of states in interoperation hub views
— an ability to specify access controls based on windows.

This work has been brought forward to the EU-funded project called Artifact-Centric Service Interoperation (ACSI) [2011]. It is promised to support a large number of service collaboration by using artifact-centric inter-operations and to achieve dramatic savings over the conventional approaches.

In contrast to orchestration perspective (e.g., Artifact-centric hub [Hull et al., 2009]), Lohmann and Wolf [2010] presented an approach for artifact-centric choreographies with the concept of agents and location-aware artifacts using Petri-net model with the concepts of agents and locations. They proposed a mechanism for automatic generation of an interaction model that serves as a contract between
the agents ensuring that specified global goal states on the involved artifacts can be reached.

Although the existing works explored the artifact-centric approach in the inter-organizational business context, a comprehensive study on supporting organizations to achieve all the three major collaboration requirements (compliance, flexibility, and autonomy) which are discussed in Section 1.2.2 is still missing.

### 2.3.3 Specification languages and standards

In the current paradigm, several specification languages and standards have been proposed for supporting the automation and coordination of business processes in a cross-organizational environment.

The Business Process Execution Language for Web Services (WS-BPEL or previously BPEL4WS) [OASIS, 2005] is a language for specifying business processes behavior based on Web services and business interaction protocols. A WS-BPEL process allows the definition of two types of business processes: abstract process and executable process. The former defines the business protocol role and describes its public aspects. The latter defines the logic and state of the business process by providing the sequence of the web-service interactions. However, WS-BPEL does not support many concepts for inter-organizational collaboration, such as the integration of manual activities and applications, the collaboration description that links roles to ports to support the heterogeneity of partners, and a standard way to specify how activities in the same process send messages to each other, etc. Importantly, WS-BPEL does not support the concept of artifacts, e.g., data is treated as isolated element as well as in a correlation mechanism, and process logic is defined by control-flow constructs based on the context of activities not data elements.

Web Services Choreography Description Language (WS-CDL) [W3C, 2005] is an XML-based language that describes peer-to-peer collaborations of participants by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common
business goal. Apart from WS-BPEL and WS-CDL, Zaha et al. [2006] proposed Let’s Dance language for modeling behavioral dependencies between service interactions providing a unified framework for capturing interactions both from a local and from a global viewpoint (i.e., behavioral interfaces and choreographies) and addressing abstraction, comprehensibility, and suitability requirements. Let’s Dance language can be transformed into WS-BPEL for local execution. Similarly, Decker et al. [2007] proposed BPEL4Chor language to define choreographies based on the extension of BPEL in which it decouples non-technical specifications from web-service specific configurations, thus, allowing the reuse of choreographies for different technical groundings. Recently, Decker and Weske [2011] proposed an interaction modeling language based on the BPMN called iBPMN, which is claimed to allow easier and less error-prone creation of choreographies (e.g., incompatible control flow dependencies and not properly reflected decisions). However, the above existing works do not provide a support for specifying the behavior of artifacts nor their interactions that constitute specification of local processes and a global process.

2.4 User Interfaces for Business Processes

The use of workflow management system in organizations has been considered as a promising automation approach to enable them to design, execute, monitor and control their business processes. It is conceived that current workflow technologies support organization’s own developed web applications for users to efficiently interact with the processes they involve with. From literature and practices, Model-Driven Web Engineering (MDWE) is considered as a core methodology that provides an efficient and effective way for the development of UIs in the modern web engineering era.

2.4.1 Model-driven web engineering

In the past several years, the area of web engineering, which is seen as a particular direction of software engineering that addresses the development of web applications and systems, has been focusing on the Model-Driven Engineering
(MDE) paradigm. Most of the current web engineering approaches (WebML [Ceri et al., 2000], OOWS [Fons et al., 2003], OOHDM [Schwabe et al., 1996], WSDM [Troyer and Leune, 1997], WebSA [15]) have been proposed to build different views of web systems based on separation of concerns. Model Driven Architecture (MDA) has been conceived as the most well-known approach to MDE that is established by the Object Management Group [OMG, 2004]. With MDA, applications are designed and modeled at a platform independent level, and then are transformed to other models based on independent platform specific requirements and implementations. Model-Driven Web Engineering (MDWE) can be seen as one of the applications of the MDA in the domain of web application and system development. The MDWE provides the separation of design and development layers of web applications which are captured by separate models, e.g., content, navigation, process, and presentation layers. With the use of these models, they can be integrated and transformed to codes for web pages, web configurations, and executable program codes.

Firstly, Fons et al. [2003] presented an Object-Oriented Web Solutions (OOWS) approach providing conceptual modelling facilities to OO-Method [Pastor et al., 2001] in order to model web applications. They proposed two models to properly capture web application requirements: navigation and presentation models. The navigation model contains conceptual modelling primitives for capturing the navigational semantics of web applications at the system specification step. The presentation model captures the presentation requirements specified by means of patterns that are associated to the primitives of the navigational context (navigational classes, navigational links, access structures, searching mechanisms, etc.). These two models are used to generate web applications.

Later, Valderas et al. [2005] introduced a strategy to automate the derivation (model-to-model transformations) of the navigational model of the OOWS method from the OOWS requirements model based on graph transformations. From the automatic transformation between the OOWS navigational model and OOWS requirement models, web application prototypes can be obtained that describe different aspects of a web application. The system static structure and the system behavior are described in three models: class diagram and dynamic and functional
models (which are borrowed from the OO-Method). Finally, web codes can be obtained from the requirement models by performing an automatic transformation.

### 2.4.2 User interfaces and business processes alignment

Traditionally, user interfaces are designed and built based on conventional activity-centric business process modeling approach which can be seen in several MDA-based UI development approaches, e.g., in OOWS-navigational model [Fons et al., 2003], for the semi-automatic generation of user interfaces by deriving task-based models from business process models. Once a business process is designed, a set of workflow constraints specified in the business process must be mapped into: (1) navigation (or page flow) constraints among the pages of activities, and (2) data input (of the form) and queries on the workflow data for checking the status of the process. This mapping is used to guarantee that the data is shown in and required by a particular user interface is consistent with the current user navigation step described by the process specification. With this concern, there are several works developed to support the (semi or fully) automatic mapping and generation of UIs for business processes. In overview, how the MDA approach can be used for the design and code generation of UIs for a business process is illustrated in Figure 2., which is taken from Sousa et al. [2008].
Initially, Torres and Pelechano [2006] proposed a first idea of a MDA method based on web engineering research concentrating on the generation of semantic web service descriptions. They presented a solution to generate web applications, based on a model transformation approach, whose specification is coupled to a business process specification. The OOWS approach was used in order to accept human participant interaction specifications to generate the appropriate web solution apart from automated participants (in the form of web services). Two sets of transformations were introduced: Model-to-Model and Model-to-Text transformations. The former is used for the transformation from BPMN standard to OOWS specification, while the latter is used to transform BPMN into WS-BPEL definition, which can be used for the implementation and execution of a business process. The MDA methodology can guide UI solution design to meet business design goals, and continuing to meet changing goals through UIs design iterations. Sukaviriya et al. [2007, 2009] confirmed a vision that a model-driven UI development environment can reduce the UI production time in the context of business design by automating the tedious part of the design process while still
enabling UI designers to freely develop their design; therefore, the design integrity is passed from UI designers to the solution developers.

Sousa et al. [2008] proposed a model-driven approach for organizational engineering in which user interfaces of information systems are derived from business processes. This approach, which is shown in Figure 2., consists of four steps: business process modeling, task model derivation, task refinement, and user interface model derivation. Each of these steps is used in terms of specifying and refining mappings between the source and the target model where the modification of any model is propagated through the derivation hierarchy. By applying this model-driven approach, the user interfaces of the web application systems can meet the requirements of the business processes. Along with the spirit of MDA paradigm, Brambilla et al., [2007] proposed a MDA approach to design and develop semantic web service applications and their components based on Web Service Modeling Ontologies (WSMO) standard [2006]. They showed that business processes and web engineering models have sufficient expressive power to support the semiautomatic extraction of semantic descriptions (i.e., WSMO ontologies, goals, Web services, and mediators). Their method was developed based on BPMN model combined with WebML model [Ceri et al, 2000], which is a web engineering model for designing and developing semantically rich web applications. Model transformations are performed by means of specialized Extensible Stylesheet Language Transformation (XSLT) stylesheets [W3C, 2010] that process the internal lanes of the BPMN model and produce XML serialization of a WebML model.

Recently, Daniel et al. [2012] proposed an approach to distributed UI orchestration, which is a component-based development technique that introduces a concept into the workflow management and service composition. They proposed a model for UI components with the extension of BPEL to support the orchestration. Technically, they equipped BPEL with a UI modeling environment and a code generator to produce artifacts that can be directly executed in their runtime environment. Based on this approach, intra-page UI synchronization is separated from distributed UI synchronization and service orchestration. A main advantage of the approach is that it recognizes the need for abstraction and more expressive
models and languages at the design time, a new language or system is not needed at the runtime.

All the above existing works considered traditional activity-centric process models and proposed approaches to define the internal components and functionalities of the UIs. There are few works carried out in a data-centric setting.

Deutsch et al., [2007] studied data-driven web applications provided by web sites interacting with users or applications. A web application can access an underlying database, as well as state information updated as the interaction progresses, and receive user inputs. The structure, contents, and task actions to be taken in web pages are determined by current state and inputs of the web page. The sequences of events (inputs, states, and actions) resulting from the interaction are expressed in linear or branching-time temporal logics in order to support to verification of the application. Although this work has not focused on the context of business processes as an initial requirement for the development of web applications, their technique can be used to guide (possibly automatically generate) the specification of an artifact-centric business process from the user-application point of view.

Cohn et al. [2008] developed a Seina prototype to support user-interaction on artifact-centric business processes. With the Seina, artifact classes, their attributes, corresponding services, and rules can be defined for the construction of UIs. At this moment, we cannot find the details of this prototype as it may not be publicly available yet. However, we believe that UIs and their logic have to be created manually by both the process modelers and user interface designer based on the view of users rather than the view of a business process. The separation and connection between two models seems not clear. Thus, some challenges still remain open. The development of user interfaces, especially along the line with MDA, for artifact-centric approach needs to be further investigated to support the step-wise configuration of UIs. UIs should be automatically derived from the underlying artifact-centric process specification. Importantly, the logic of each UI and the flows between UIs (e.g., what input is required, possible steps and navigation flows between UIs) should reflect an underlying process that such UIs derive from. A mechanism should support the generation of template-based UIs (e.g., in web
Chapter 2. Literature Review

forms) for UI designers to design and customize the actual UIs in the way they want.

2.5 Summary and Discussion

This chapter has introduced related works from the past to the recent in detail. From the literature, we can firstly see the state of the art of the artifact-centric approach to modeling business processes. Several works have demonstrated the importance and benefits of this approach in both theoretical and practical aspects. With the current trend of SOA and the force from ever-changing and competitive business environment, artifact-centric approach has been showing itself as a promising solution to support the design and improvement of business operations. Although, several works have been carried out in the area of artifact-centric process modeling so far, yet there is still a need for more comprehensive supports in the modeling. We showed that the three proposed supports for artifact-centric business processes, i.e., business process reuse, efficient inter-organizational business process collaboration, and process visualization and user interfaces development, are demanded and important to the trend of today's business process management. Based on our discussion of existing works, the work carried out for such supports in the artifact-centric paradigm is still missing or in its infancy stage. Therefore, in this thesis, we aim to develop frameworks that effectively provide such three supports, and thus, promote the importance, value, and wider usefulness of the emerging artifact-centric approach.
Chapter 3.

Modeling Artifact-Centric Business Processes

In this chapter, we discuss the artifact-centric approach for modeling business processes. As already introduced in Section 2.1, the artifact-centric business process modeling has a particular focus on business artifacts (or artifacts) and their evolution throughout a business process they are involved with. In contrast to the conventional activity-centric approach, the artifact-centric approach equally lifts relevant business data and activities that perform on the data at the same level of consideration in the modeling phase. We start with the brief description of a major component that is used to specify a business process based on the artifact-centric approach. Artifacts are either physical or digital records/documents that are involved in business processes. To follow the spirit of the paradigm in business process automation and service-oriented architecture, we do not distinguish physical entities with digitalized entities in this thesis. Their evolution from the
**Chapter 3. Modeling Artifact-Centric Business Processes**

Creation to the termination (achieved or destroyed) of their lifecycle is done upon a series of tasks that perform on them. In other words, we assume an artifact and a task are a digital record and a computerized (or web-) service, respectively. To describe the evolution of artifacts, a mechanism to specify how tasks are associated with artifacts is required. We consider the use business rules as their foundation studies have been laid out in terms of suitability and capability to capture the specification of business process in a highly dynamic and flexible way. Intuitively, a complete set of business rules defined for artifact-centric business processes specifies the control logic of the whole process from its beginning to its end. As can be found in the literature, business rules can be defined to express the logic of business processes in different languages. However because our main focus is not on detailed languages, in this thesis we do not consider any specific one but use a simple logical expression in Event-Condition-Action (ECA) rule-based style. A simple yet comprehensive model for artifact-centric business processes, desired behavioral properties, and model verification is discussed in this chapter, which sets a basis for the further study of the three proposed major supports for modeling artifact-centric business processes in the later chapters.

This chapter is organized as follows. In Section 3.1, we introduce an illustrative running example of a purchasing business process based on an artifact-centric perspective. Section 3.2 introduces an Artifact-Centric Process model (ACP model) for capturing the specification of a business process. Then in Section 3.3, we discuss the desired behavioral properties of ACP model and model verification. Finally, the discussion and summary of this chapter is presented in Section 3.4.

### 3.1 Running Example

In this section, we take a purchasing process (between a buyer and a supplier) in a supply chain domain as an example to illustrate and motivate the artifact-centric approach to modeling business processes. As shown in Figure 3., a complete purchasing process model is illustrated based on an artifact-centric perspective. We initiate the discussion of this example 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: *Purchase Order (PO)*, *Shipping Order (SO)*, and *Invoice (IV)*. The interrelations between artifacts can be drawn (dashed-line) as *synchronization dependencies* of such artifacts.

![Diagram of artifact-centric purchasing processes](image)

Figure 3.: An overall view of artifact-centric purchasing processes

Now, let us briefly describe this process based on the artifact-centric approach. At the beginning of the process, a buyer creates a *purchasing order* consisting of a list of goods. Immediately after created, the purchase order is waiting for the approval. Once approved, the purchase order is confirmed and sent to a selected supplier. When the supplier receives the purchase order, it seeks to supply the goods for that order. If the goods run out, then the supplier rejects to supply them and then the purchase order is canceled. Otherwise, the goods are filled, and then the supplier creates a *shipping order* for a delivery. When the goods are in the transit process, at the meanwhile the supplier issues an *invoice*, and then sends it to the buyer and waits for the payment. After the buyer clears the total amount owing in the invoice, consequently, the supplier marks the purchase order as closed. This purchasing process completes when the purchase order is in the *closed* state, the shipping order is in the *arrived* state, and the invoice is in the *cleared* state.

### 3.2 Syntax of Artifact-Centric Process (ACP) model

Here, we formally define the Artifact-Centric Process model (*ACP model*) for a business process. An *ACP model* consists of three core components: *artifacts, tasks,*
and business rules. An artifact is a business entity or an object involved in inter-organizational business processes. A task is used to perform read/update operations on artifact(s). A business rule is defined by a family of constraints in a Condition-Action style to describe which task is executed and which state of artifact is changed under what condition.

Definition 3.: (Artifact class). An artifact class abstracts a group of business artifacts with their attributes and states. Artifact class \( C \) (or artifact if the context is clear) is a tuple \((A, S, s^{\text{init}}, S^f)\) where,

- \( A = \{a_1, a_2, \ldots, a_x\}, a_i \in A(1 \leq i \leq x) \) is a name-value paired attribute of scalar-typed value, or an array list of nested attributes (denoted as \([\ ]a_l(a_j, a_{j+1}, \ldots, a_{j+m}))\),
- \( S = \{s_1, s_2, \ldots, s_y\}, s_i \in S(1 \leq i \leq y) \) is a state,
- \( s^{\text{init}} \) denotes the initial state,
- \( S^f \subseteq S \) is a set of final states.

Example 3. The followings show some of artifact classes that are defined based on the purchasing process introduced in Section 3.1.

1. **Purchase Order (PO)** = \((\{\text{OrderID}, \text{SupplierID}, \text{grandTotal}, \text{SubmitDate}, \text{CompleteDate}\}, \text{init}, \{\text{approving, confirmed, supplying, filled, delivering, billing, closed, canceled}\}, \{\text{canceled, closed}\}).

2. **Shipping Order (SO)** = \((\{\text{ShippingID, OrderID, SubmitDate, ShipDate, CompleteDate}\}, \text{init}, \{\text{in\_transit, arrived}\}, \{\text{arrived}\}).

3. **Invoice (IV)** = \((\{\text{InvoiceID, OrderID, IssuedDate, Amount, ClearedDate}\}, \text{init}, \{\text{issued, unpaid, cleared}\}, \{\text{cleared}\}).

Definition 3.: (Artifact schema). An artifact schema, denoted as \( Z \), contains a set of artifact classes, i.e., \( Z = \{C_1, C_2, \ldots, C_x\} \) where \( C_l \in Z(1 \leq i \leq x) \) is an artifact class.

We also define some basic predicates over schema \( Z \) to be used for defining business rules as followings.
—*defined*(C, a) iff attribute *a* ∈ *C.A* of artifact of class *C* has a value;

—*instate*(C, s) iff state *s* ∈ *C.S ∪ C.S^f* of artifact of class *C* is active. Initially, *instate*(C, s^{init}) implies ∀*x* ∈ *C.A*, ¬*defined*(C, x).

Next, we define business rule to express the processing control logic of a business process.

**Definition 3.: (Business Rule).** A business rule regulates which task can be invoked under what pre-condition. The conditional effect is also defined to restrict the post-condition after performing such task. Business rule *r* is triple (λ, β, υ) where,

—λ and β are sets of pre-condition and post-condition, respectively. For simplicity, we restrict both types of conditions to conjunctive normal form (CNF) containing two types of propositions over schema *Z*: (1) state proposition (the *instate* predicate) and (2) attribute proposition (the *defined* predicate and scalar comparison operators),

—υ ∈ *V* is a task or a composite task (i.e., a service) to be performed, where *V* = \{υ_1, υ_2, ..., υ_ν\} is a set of tasks of which performs operations on some artifacts *C_1*, *C_2*, ..., *C_ν* where *C_j* ∈ *Z* (1 ≤ *j* ≤ *y*).

Table 3. lists some business rules with simple and necessary conditions used in the purchasing process introduced in our motivating example in Section 3.1.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Pre-condition</th>
<th>Task</th>
<th>Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_1</td>
<td><em>instate</em>(po, approving) ∧ <em>defined</em>(po, OrderID) ∧ <em>defined</em>(po.SupplierID)</td>
<td><em>confirm</em>(po)</td>
<td><em>instate</em>(po, confirmed) ∧ <em>defined</em>(po.SubmitDate)</td>
</tr>
<tr>
<td>r_2</td>
<td><em>instate</em>(po, confirmed) ∧ <em>defined</em>(po.OrderID) ∧ <em>defined</em>(po.SupplierID)</td>
<td><em>supply</em>(po)</td>
<td><em>instate</em>(po, supplying)</td>
</tr>
<tr>
<td>r_3</td>
<td><em>instate</em>(po, confirmed) ∧ <em>defined</em>(po.OrderID) ∧ <em>defined</em>(po.SupplierID)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3. Modeling Artifact-Centric Business Processes

<table>
<thead>
<tr>
<th><strong>Order so, and simultaneously issues Invoice iv for po</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-condition</strong></td>
</tr>
</tbody>
</table>
| **Task** | `dispatchGoods(po, so)`  
| | `issueInvoice(po, iv)` |
| **Post-condition** | `instate(po, delivering) instate(so, in_transit) ∧ instate(iv, issued) ∧ defined(so.ShipperID) ∧ defined(iv.InvoiceID) ∧ defined(iv.OrderID)` |

**r_1**: Supplier receives a payment for Invoice iv and closes Purchase Order po

| **Pre-condition** | `instate(po, billing) ∧ instate(iv.unpaid)` |
| **Task** | `closeOrder(po, iv)` |
| **Post-condition** | `instate(po, closed) ∧ instate(iv.cleared)` |

In order to maintain the existence of valid and explicit state changes of an artifact in business rule \( r \), we require the following statement to hold for \( r \). There exists a couple of `instate` predicates of that artifact in both pre-condition and post-condition of \( r \), i.e., for some states \( s_1, s_2 \in C.S \), if there exists `instate(C, s_x)` in \( r.\lambda \), then there exists `instate(C, s_y)` in \( r.\beta \).

The state change refers to either a transition from one state to another state, or to itself. Note that our work assumes one-to-one correspondence between a business rule and a transition of artifact, i.e., transition from one state to another state can be triggered by one and only one business rule. If a transition can be fired by multiple business rules, then those rules should be combined into a single rule.

We also classify state conditioning into two types by determining the existence of the `instate` predicate in the pre- and post- conditions of a business rule. The first type is classified by the case that a change of state occurs in only one artifact, while the second type refers to simultaneous changes of states of multiple artifacts, i.e., more than one pair of `instate` predicates (each pair for each artifact) appears in the pre- and post-conditions of a single business rule. As the second-typed business rule is used for the expression of synchronization between artifacts, we call it as a synchronization (sync) rule.

**Definition 3.** (Sync rule). Business rule \( r \) is called a sync rule for artifact \( C_x \) and artifact \( C_y \) if there exists `instate(C_x, s_i)` and `instate(C_y, s_m)` in \( r.\lambda \) and `instate(C_x, s_j)` and `instate(C_y, s_n)` in \( r.\beta \), where \( s_i, s_j \in C_x.S \) and \( s_m, s_n \in C_y.S \).
As mentioned above, a single sync rule can be used to synchronize more than two artifacts.

**Example 3.** In Table 3., we can see that business rules \( r_1 \) is used to express only the state change of the PO artifact (\( \text{approving} \rightarrow \text{confirmed} \)), while business (sync) rule \( r_2 \) is used to simultaneously change states of the PO artifact (\( \text{billing} \rightarrow \text{closed} \)) and the IV artifact (\( \text{unpaid} \rightarrow \text{cleared} \)). Similarly, sync rule \( r_3 \) is used for the synchronization of three artifacts PO, SO, and IV (notice a composite task is defined in the action part of \( r_3 \)).

In this thesis, we do not focus on the task-level information, i.e., the specification of task is omitted. However, the specification of task in artifact-centric process modeling approach can be defined in the spirit of semantic web-services specified in OWL-S proposal [OWL Services Coalition, 2003] – that is in a form of Input, Output, Pre-condition, and Effect (IOPE). The pre-condition and conditional effect of a task should conform to the pre-condition and post-condition, respectively, of the business rule that specifies the task in its action. A detailed discussion of semantic web-services together with the uses of business rules for modeling business process by using artifact-centric approach can be found in [Bhattacharya et al., 2007; Cohn and Hull, 2009].

**Definition 3.** (Artifact-Centric Process Model or ACP model). Let \( \Pi \) denote for an ACP model, and it is tuple \((Z, V, R)\) where,

- \( Z = \{C_1, C_2, \ldots, C_x\} \) is an artifact schema,
- \( V = \{v_1, v_2, \ldots, v_y\} \) is a set of tasks,
- \( R = \{r_1, r_2, \ldots, r_z\} \) is a set of business rules.

In addition, we define two auxiliary functions over business rules \( R \) and artifact schema \( Z \) in \( \Pi \) to return a set of states of the pre-condition or post-condition of a given business rule.

- function \( \text{pre}_s(r, C) \) returns a set of states \( \{s_1, s_2, \ldots, s_x\} \) where business rule \( r \in Z \) and state \( s_i \in C.S(1 \leq i \leq x) \) is defined in the \textit{instate} predicate of the pre-condition of \( r \).
—function \( \text{post}_s(r,C) \) returns a set of states \( \{s_1, s_2, \ldots, s_y\} \) where business rule \( r \in Z \) and state \( s_j \in C.S(1 \leq i \leq y) \) is defined in the \text{instate} predicate of the post-condition of \( r \).

It is noted that, as previously mentioned, we do not focus on developing an artifact-centric business process model compared to the existing works based on CTL, LTL, or \( \mu \)-calculus specifications. Our model is more based on simple conjunctive ECA-rules describing the behavior of the process without taking into account actual data attribute but considering on state attributes, which represent the current condition of data values at a time. As a base of the study for supporting the modeling in general, we do not focus on any specific language when describing and analyzing the dynamic behavior of artifacts and a process. Our work focuses on artifact lifecycles (which is formalized by using LTSs) and their synchronization dependency (between LTSs).

### 3.3 Behavioral Properties and Model Verification

In this section, we discuss the behavioral properties and model verification of artifact-centric business processes. In general, it is important that the behavior of business processes must be sound in terms of being able to guarantee the reachability of desired goals of the process [Van der Aalst et al., 2010; Lohmann and Wolf, 2010; Klai et al., 2011].

We classify behavioral properties of artifacts in \( ACP \) model into \textit{intra-behavior} and \textit{inter-behavior}. The intra-behavior of an artifact describes how an artifact changes its state throughout its lifecycle. Here, we adopt Labeled Transition System (LTS) to capture the lifecycle of an individual artifact. Second, the inter-behavior describes how the lifecycle of one artifact depends on the counterpart of another artifact, and it can be represented as synchronization dependency between artifacts, i.e., a \textit{sync rule}. 
Definition 3. (Lifecycle of artifact, $\Rightarrow$). Let $C_i = (A_i, s_i^{\text{init}}, S_i, S_i')$ be an artifact class in ACP model $\Pi$. A lifecycle of $C_i$, denoted as $\mathcal{L}_{C_i}$, can be defined as a tuple $(S, s_i^{\text{init}}, \Rightarrow)$ where,

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

—state transition relation $\Rightarrow \subseteq S \times R_i \times G_i \times S$ where,

—$R_i \subseteq \Pi.R$ is a set business rules that are used to induce state transitions of artifact $C_i$ such that,

$$\forall r \in \Pi.R, \exists s_x, s_y \in C.S, s_x \in \text{pre}_s(r, C_i) \land s_y \in \text{post}_s(r, C_i) \rightarrow r \in R_i,$$

—$G_i$ (guards) is a union set of state preconditions of each business rule in $R_i$ such that each precondition references to a state of other artifact in $\Pi$, i.e.,

$$G_i = \bigcup_{j=1}^{n_2} \{ C_j, s | \exists C_j \in \Pi.Z, s \in \text{pre}_s(r, C_j) \land C_j \neq C_i \},$$

We also denote $\Rightarrow^*$ for a reflexive transitive closure of $\Rightarrow$. We write $s_i \Rightarrow^* s_j$ if state $s_j$ can be reached from state $s_i$ by some sequence of business rules in $\Pi.R$.

Next, we define lifecycle occurrence to refer to a particular sequence of states occurring from one state to another state in the lifecycle.

Definition 3. (Lifecycle occurrence or $L$-occurrence). Given ACP model $\Pi$ and lifecycle $\mathcal{L} = (S, s_i^{\text{init}}, \Rightarrow)$ (of either an artifact class in $\Pi$ or $\Pi$), we call 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 state $s_x$ to state $s_y$ is denoted as $\sigma^L_{s_x \rightarrow s_y} = (s_x, \ldots, s_y)$ where $s_x, s_y \in S$ such that,

$$\forall s \in \sigma^L_{s_x \rightarrow s_y}, \exists s \in S, s_x \Rightarrow^* s \land s \Rightarrow^* s_y.$$  

We write transition $s_x \overset{r|g}{\Rightarrow} s_t$ to mean that the state of the artifact will change from source state $s_x$ to target state $s_t$ if business rule $r$ is fired and guard $g$ (state pre-condition of $r$) holds. Note that in a clear context, we may use shorthand $s_x \Rightarrow s_t$
without its superscription, and may use term *artifact* for the mean of *lifecycle of artifact*.

Based on *Definition 3.*, given ACP model $\Pi$, we can derive a lifecycle corresponding to an artifact in $\Pi$ from a set of corresponding business rules that are used to trigger the state transitions of the artifact. We can obtain the lifecycle of an entire process by composing all artifacts in the model. Here, we define ACP lifecycle for describing the behavioral aspect of an ACP model consisting of synchronized lifecycles of artifacts. We adapt a state machine composition technique presented in [Lind-Nielsen et al., 2001] for generating the lifecycle of ACP. The composed lifecycle of ACP model is used for the verification of the model.

*Definition 3.* (Lifecycle composition, composed lifecycle, $\otimes$). Let $L_i = (S_i, s_i^{\text{init}}, \Rightarrow_i)$, and $L_j = (S_j, s_j^{\text{init}}, \Rightarrow_j)$ be two artifact lifecycles in ACP model $\Pi$. *Lifecycle composition* (i.e., synchronized product) of $L_i$ and $L_j$ is denoted as $L_c = L_i \otimes L_j = (S_c, s_c^{\text{init}}, \Rightarrow_c)$ where,

---

$S_c \subseteq L_i \times L_j \times S$ is a set of composed states,

---

$s_c^{\text{init}} = (L_i, s_i^{\text{init}}, L_j, s_j^{\text{init}})$ is the initial state,

---

$\Rightarrow_c \subseteq S_c \times \Pi \times R \times G_c \times S_c$ is a transition relation where $G_c$ is a set of guards (state propositions).

Now, let $g[s_{L_i}^T / \text{state}(L_i, s_x)]$ denote that state $s_{L_i}^T$ in guard $g$ is substituted by *true* or *false* (of state predicate) depending on whether the local state of $L_i$ is $s_x$. We can formulate transition relation $\Rightarrow_c$ of *composed lifecycle* $L_c$, by using the following three inference rules.

\[
(s_{L_i}^T, r, g, s_{L_j}^T) \in \Rightarrow_i \quad \Rightarrow \quad (s_{L_i}^T, s_{L_j}^T) \in \Rightarrow_c \quad \text{with} \quad g_c = g_i \cdot \text{state}(L_i, s_x)
\]

(3.1)

\[
(s_{L_i}^T, r, g_2, s_{L_j}^T) \in \Rightarrow_j \quad \Rightarrow \quad (s_{L_i}^T, s_{L_j}^T) \in \Rightarrow_c \quad \text{with} \quad g_c = g_j \cdot \text{state}(L_i, s_x)
\]

(3.2)
Rule (3.1) and Rule (3.2) are applied when business rule $r$ is fired on only individual lifecycle $L_i$ and $L_j$, respectively. Rule (3.3) is applied when sync rule $r$ is fired on both lifecycles $L_i$ and $L_j$. As the three inference rules apply the substitution of state conditions of two lifecycles in the composition, references to external lifecycle are not replaced.

**Example 3.** Figure 3. shows the composition between the lifecycle of artifact $C_1$ and the lifecycle of artifact $C_2$. The label $r_1[g]$ attached to a transition means that the transition is fired when both the attribute proposition in the pre-condition of business rule $r_1$ holds and all state propositions (of external lifecycles) in $g$ hold. We denote the counter state condition of $C.s_x$ by symbol $-C.s_x$ in the guard. We can also see that state conditions referencing to artifacts $C_3$ and $C_4$ remain in the composed lifecycle but in different forms, which depend on the transition they belong.

Now, we can define the lifecycle of ACP model by using *lifecycle composition*.

**Definition 3.** (ACP Lifecycle, compose_ACP). Given ACP model $\Pi$, a (ACP) lifecycle of $\Pi$, denoted as $L_\Pi$, can be generated by iteratively performing lifecycle composition of every artifact in $\Pi$. We define function $\text{compose}_A$($\Pi$) to return $L_\Pi$ of $\Pi$.

Note that lifecycle composition is associative and commutative, i.e., $L_i \otimes L_j \otimes L_k = L_i \otimes (L_j \otimes L_k) = (L_i \otimes L_j) \otimes L_k$ and $L_i \otimes L_j = L_j \otimes L_i$. Therefore, the final result of the composition of a set of lifecycles is not impacted by their composition order.
Next, we define \textit{soundness} property to describe a desired and correct behavior of artifact lifecycle and the process.

\textit{Definition 3.: (Safe, Goal-reachable, and Sound lifecycle).} Given ACP model $\Pi$ and lifecycle $L = (S, s^{\text{init}}, \Rightarrow)$ (of either an artifact class in $\Pi$ or $\Pi$), we define a set of lifecycle states $S = L.S \cup \{s^{\text{init}}\}$ and a set of final states $S^f \subseteq S$. Lifecycle $L$ is said to be:

—\textit{safe} iff there exists business rule $r \in \Pi.R$ such that $r$ induces one and only one transition in $L$, i.e.,

\[ \forall s_x, s_y \in S, \forall s_m \in S \backslash \{s_x\}, \forall s_n \in S \backslash \{s_y\}, \exists r \in \Pi.R, (s_x, r, g, s_y) \in \Rightarrow \rightarrow (s_m, r, g, s_n) \notin \Rightarrow; \]

—\textit{goal-reachable} iff,

(1) for every non-final state $s$ of $L$, $s$ can be reached from the initial state and $s$ can reach one of the final states of $L$, i.e.,

\[ \forall s \in S \backslash S^f, \exists s_f \in S^f, s^{\text{init}} \Rightarrow^* s \land s \Rightarrow^* s_f, \]

(2) for every final state $s_f$ of $L$, $s_f$ can be reached from the initial state, i.e.,

\[ \forall s_f \in S^f, s^{\text{init}} \Rightarrow^* s_f, \]

—\textit{sound} iff $L$ is safe and goal-reachable.

Note that the goal-reachability property implies deadlock-free and strongly-connected lifecycle of an artifact (or a process). Next we present two Theorems formulated to show how the soundness of two lifecycles and the soundness of their composed lifecycle are related. THEOREM 3. shows that the composed lifecycle of two isolated and \textit{sound} lifecycles is always \textit{sound}; however, in THEOREM 3., if there is any sync rule between \textit{sound} lifecycles to be composed, the result composed lifecycle may not be \textit{sound}. The proofs of these two theorems can be found in the Appendix.

\textbf{THEOREM 3..} Let lifecycle $L_c$ be the \textit{composed lifecycle} of lifecycles $L_x$ and $L_y$ in ACP model $\Pi$. If both $L_x$ and $L_y$ are \textit{sound} and sync rules $\Phi(L_x, L_y) = \emptyset$, then $L_c$ is \textit{sound}. 

61
THEOREM 3.. Let lifecycle $L_c$ be the composed lifecycle of lifecycles $L_x$ and $L_y$. If both $L_x$ and $L_y$ are sound and sync rules $\varphi(L_x, L_y) \neq \emptyset$, then $L_c$ needs not be sound.

In the rest of thesis, we restrict our discussion only to the sound behavior of artifacts and ACP model based on their lifecycles (not the changes of artifact’s data). However, discussions and formal approaches to data verification of artifact-centric business processes (some call artifact systems) can be found in several existing literature, e.g., [Liu et al., 2007; Gerede and Su, 2007; Fritz et al., 2009; Deutsch et al., 2009; Damaggio et al., 2011].

3.4 Summary and Discussion

In this chapter we introduced the Artifact-Centric Process (ACP) model for capturing the specification of a business process with primary focuses on key business artifacts, their lifecycles formalized by Labeled Transition System (LTS), and their interactions expressed by the synchronization dependency between lifecycles. We also presented the lifecycle composition technique (based on synchronized product of artifact lifecycles) to generate the lifecycle of the ACP, and then provided a discussion of the sound behavior property of artifact-centric processes based on the safetyness and goal-reachability conditions of the process. Finally, we formulated two theorems that show how the composed lifecycle of ACP can satisfy the soundness condition.

In our work, we do not focus on the development of modeling language nor restrict our ACP model to any language. In the essence of artifact-centric modeling approaches, rule-based languages are deemed most suitable in its sprit to achieve highly-flexible process specifications. We introduced our ACP specification by using simple condition-action-styled business rules to define the control logic of business processes. This kind of rules can be extended to fully-fledged ECA rules in more declarative style with temporal constraints (e.g., CTL [Gerede and Su, 2007], TiLe [Zhao et al., 2009], and LTL-FO [Damaggio et al., 2011]). More declarative lifecycle specification based on GSM Model [Hull et al., 2010] with SBVR business rules can be incorporated into the ACP model to provide more advanced constructs for the
model specification. As discussed, data verification of business rules is not in our scope. On the other hand, we focused on the verification of the behavior of processes based on the analysis of LTSs; therefore, the interpretation of business rules to generate (LTS) lifecycles of artifacts is required (for our rule syntax, the lifecycle can be constructed based on Definition 3).

The ACP model and the LTS behavior model of an artifact and a process presented in this chapter will be used as a basis for the discussion of our proposed three supports for the modeling of artifact-centric business processes throughout the rest of this thesis.
Driven by complex and dynamic business process requirements, there has been an increasing demand for business process reuse to improve modeling efficiency. Process specialization is an effective reuse method that can be used to customize and extend base process models to specialized models. With the process specialization, a process model can be reused for a more specialized process context. Similar to software reuse, the artifact-centric process reuse can help organizations design their new process models based on existing models in an object-oriented style. In order to achieve this, we borrow the idea of object specialization and study the specialization in the context of processes that consist of several interacting objects. Specifically, we study how a specialized process can be constructed based on the existing process and how the behavior of the specialized process is consistent with its base process. Although, process specialization has been studied for the
traditional models by treating a process as a single object, the specialization of artifact-centric processes that consist of multiple interacting artifacts has not been studied. Inheriting interactions among artifacts for specialized processes and ensuring the consistency of the processes are challenging.

In this chapter, we introduce the notion of behavior-consistent process specialization for artifact-centric business process models. We propose a framework for process specialization that includes artifact-centric process models, methods to define a specialized process model based on an existing process model, and the behavioral consistency rules.

We organize the rest of this chapter as follows. Section 4.1 introduces the motivation of process specialization along with the example and problem statements. Section 4.2 discusses the methods to specialize artifact-centric business process models. Section 4.3 provides the analysis and discussion on the behavioral consistency between two processes. Section 4.4 discusses how the process specialization can be achieved while satisfying the condition of the behavioral consistency. Section 4.5 shows a case study on specializing purchasing process, and the summary and discussion for this chapter is presented in Section 4.6.

4.1 Motivation to Process Specialization and Problem Statements

In this section, we motivate the artifact-centric business process specialization from our purchasing process introduced in Section 3.1. Let consider the case that a retailer and a supplier are agreed to have online purchasing process separate with offline purchasing process based on the reuse of existing generic purchasing process. They look for the methodologies that allow them to 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 process. We can consider a process as a class of a container object consisting of interrelated artifact classes. Figure 4. illustrates our case of having a Generic purchasing process model and its two possible specialized process models, which are Online purchasing process and Offline purchasing.
process. The former offers a service to only retail customers over the Internet, while the latter accepts both retail and wholesale customers. In Figure 4., a round 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 specialization relationship is also applied for artifact classes.

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 4., 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 know only on 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 aggregatable (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 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 flexibility. 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.

Although a specialization of an individual object or treating a process as a single object has been studied in object-oriented research (e.g., in [Van Der Aalst and Basten, 2002; Schrefl and Stumptner, 2002; Wyner et al., 2002, 2003]), however, process specialization that consists of interacting objects requires further support. Dependencies between artifacts become a major concern since they can lead the behavior of a specialized process inconsistent to 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 artifact specializes its base artifact in the ordering process, e.g., *Web PO* specializes *Purchase Order*. Not only the internal behavior of *Web PO* is specialized, but it is possible that some synchronization between *Web PO* and the other artifact(s) may also need to be modified due to the specialization. In addition, it is possible that a newly defined artifact class can be added into the specialized process, e.g. in Figure 4., the *Quote* artifact is added to the *Offline purchasing* process. Extending this artifact, of course, requires some synchronization with the other artifact(s), e.g., *Offline PO*. Technically, the specialization of ACP models brings in the following key questions.

—What is an approach that allows process modelers to define and specialize the process model?

—How can the behavioral consistency between a specialized process model and its base process model be preserved?

To address the above questions, we propose a framework for behavior-consistent business process specialization of ACP models, as overviewed in Figure 4. The framework consists of three core components: (1) ACP models, (2) a method to define and construct a specialization of process models, and (3) a mechanism to
guarantee that the behavior of specialized process model is consistent with the behavior of its base process model.

Figure 4.: Overview of a framework for behavior-consistent process specialization

4.2 Specializing Artifact-Centric Business Process Models

In this section, we present the overview of process specialization, and discuss the specialization methods for artifact-centric business processes.

Figure 4. 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 particular process context. The second layer represents the hierarchy of specializations where each specialization is for a particular 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 generic ACP model. Each generic model is constructed by composing a set of generic artifact classes and a set of generic business rules that are used to coordinate the interaction between those artifacts for achieving the goals of a particular 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 from, which is called supertype. In this thesis, we do not consider multiple inheritances (i.e., one model specializes two or more models) as we observed that this concept is unnecessary complex and not applicable for the process specialization.
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 that for every artifact class in a specialized model, each class is a refined subtype of its supertype 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. In Figure 4., specialization (1) is a full-specialization since each artifact in specialized process $\Pi'_2$ inherits its supertype in generic process $\Pi_1$. Specializations (2), (3), and (4) are partial-specializations. Specialization (2) constructs specialized process $\Pi'_2$ from its super process $\Pi'_1$ by inheriting artifacts $C'_{1-1}$ and $C'_{2-1}$ from artifacts $C_{1-1}$ and $C_{2-1}$, respectively, but not artifact $C'_{1-2}$. Specialization (3) shows an extension of artifact by adding new artifact $C'_{4-2}$ to
specialized process $\Pi_2$. Lastly, specialization (4) shows a substitution (i.e., reduction then extension) of artifact $C_{2-2}'$ by artifact $C_{1-2}''$ in specialized process $\Pi_2'$. Revisiting our example in Figure 4., 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 two specialization categories, we propose that the specialization of ACP models can be achieved by three construction methods: artifact refinement and artifact extension, and artifact reduction.

—Artifact refinement. Process modelers decide to inherit an artifact from a base model by refining (adding/modifying) some corresponding business rules and states to the specialized model. The pre-condition and post-condition of a modified rule may have a state of the supertype refined into new defined state(s) in the subtype. Note that the refinement can be performed on a sync rule that is used to synchronize two or more artifacts.

—Artifact extension. Process modelers decide whether there is a need of any additional artifact for the specialized model (which it does not previously exist in the base model), e.g., the Quote artifact in Figure 4. is added into the Offline purchasing process. Adding new artifacts to a process implies that the process requires not only new business rules (for such artifact) but also synchronization rules between the new artifact and existing artifact(s).

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

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 full-specialized process that has each artifact inherits its supertype in the base process, the behavior of former may be inconsistent to the behavior of the latter as some of 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. In particular, we need to observe the
behavior of the specialized process and its base process, and investigate conditions that make the behavior of the specialized process constructed by any of the above three methods consistent with its base process. To address this requirement, we go through the following steps in the rest of this chapter.

(1) We discuss how the behavior of one process is consistent with the behavior of another process, and formulate the behavioral consistency rules that such two processes must satisfy.

(2) We define how an artifact-centric process model can be specialized based on the three construction methods.

(3) We provide theorems that can be used to support behavior-consistent specialization construction.

4.3 Behavioral Consistency Analysis

In object-oriented design approaches, the consistency of (dynamic) object behaviors between subtype and its supertype can be divided into observation consistency and invocation consistency. Observation consistency ensures that if features added at a subtype are ignored and features refined at a subtype are considered unrefined, any processing of an artifact of the subtype can be observed as correct processing from the view of the supertype. The invocation consistency refers to the idea that instances of a subtype can be used in the same way as instances of the supertype. More detailed discussion about object’s behavior consistencies can be found in [Schrefl and Stumptner, 2002]. In this thesis, we restrict our discussion of business process specialization to observation consistency (for both artifact and process). On the one hand, in the viewpoint of structure, it is ensured that the current processing states of artifact and process are always visible to some higher organizational roles. On the other hand, to preserve the behavioral consistency, it is guaranteed that business rules added at a subtype do not interfere with the business rules inherited from its supertype. Particularly, dealing with changes of synchronization dependencies between artifacts is a major technical issue of ACP specialization.
Chapter 4. Specializing Artifact-Centric Business Processes

We divide the discussion of the behavioral consistency into two sections. Section 4.3.1 introduces the generalized notion of behavior-consistency and the condition that two lifecycles must satisfy. Section 4.3.2 introduces the definitions for capturing a fragment of lifecycle and synchronization region which is used to capture a set of synchronized fragments. Then, Section 4.3.3 analyzes and discusses the atomicity of synchronized fragments which is used for checking the behavioral consistency.

4.3.1 Behavior-consistency (B-Consistency)

As already stated in Section 3.3, in this thesis we only focus on the behavior perspective of ACP models. Here, we propose a notion of behavioral consistency, called B-consistency, to check whether an ACP model (or an artifact) that derives from its base ACP model (or a base artifact) is consistently observable.

Let $L_x$ be a lifecycle of ACP model $\Pi_x$ and $L_y$ be a lifecycle of ACP model $\Pi_y$ such that $\Pi_y$ derives from $\Pi_x$. Assume that $\Pi_y$ is an abstraction of $\Pi_x$. We want to check whether the behavior of $\Pi_y$ is consistent with the behavior of $\Pi_x$. Informally speaking, if every L-occurrence of $L_x$, disregarding the states and transitions of $L_x$ that do not exist in $L_y$, is observable as the same sequence as of $L_y$, then the behavior of $\Pi_y$ is consistent with the behavior of $\Pi_x$.

Here, we define behavior-consistency (B-consistency) relation between two lifecycles to describe the condition to preserve the consistency between them. Our B-consistency relation between two lifecycles is defined by adopting the notion of bi-simulation equivalence relation in process algebras [Bloom, 1995]. By replacing a fragment of states and transitions of one lifecycle that does not exists in another lifecycle with a silent (τ) action, we can apply a weak bi-simulation between the derived lifecycle and its original lifecycle.
Chapter 4. Specializing Artifact-Centric Business Processes

**Definition 4.** (B-consistent, ≃). Let $\mathcal{L}_y = (S_y, s_y^{\text{init}}, \Rightarrow_y)$ and $\mathcal{L}_x = (S_x, s_x^{\text{init}}, \Rightarrow_x)$ be two lifecycles and $S_{xy} = S_x \cap S_y$ be a set of states that exist in both $\mathcal{L}_x$ and $\mathcal{L}_y$. We say that $\mathcal{L}_y$ and $\mathcal{L}_x$ are B-consistent (denoted as $\mathcal{L}_y \approx \mathcal{L}_x$) iff,

\[
\forall s_i, s_j \in S_{xy}, \exists (s_i, r, g, s_j) \in \Rightarrow_x, \forall s_k \in S_y \setminus S_{xy}, (s_i \Rightarrow y s_k \land s_k \Rightarrow y s_j),
\]  

\[
\forall s_i, s_j \in S_{xy}, \exists (s_i, r, g, s_j) \in \Rightarrow_x, \forall s_k \in S_y \setminus S_{xy}, \neg(s_i \Rightarrow y s_k \land s_k \Rightarrow y s_j).
\]

Note that the definition of B-consistency is generalized; therefore, it can be used for checking any construct of lifecycle (i.e., ACP lifecycle, artifact lifecycle, or composite lifecycle).

**Figure 4.** Examples of derived lifecycles and its base lifecycle

**Example 4.** Figure 4 shows a base lifecycle in (a) and its various derivations in (b), (c), and (d). The lifecycle in (a) is not B-consistent with lifecycle in (b), i.e., $\mathcal{L}_a \not\approx \mathcal{L}_b$. This is because, in some L-occurrences of lifecycle in (b), state $a$ can reach state $c$ (through state $x_2$) without passing state $b$; and, state $a$ can reach itself via state $x_4$ without passing state $b$. In contrast, we can see that $\mathcal{L}_a \approx \mathcal{L}_c$ and $\mathcal{L}_a \approx \mathcal{L}_d$ in (c) and (d), respectively.

**4.3.2 Lifecycle fragments and synchronization regions**

Now, we discuss the behavior of a part of an artifact and the behavior of synchronized parts between artifacts. First, we define a fragment of lifecycle called L-fragment that contains a set of sub states and sub transitions for capturing a particular sub-lifecycle. Then, to capture synchronization dependencies between lifecycles, we need to extend the definition of L-fragment for synchronization region.
(called \textit{S-region}) which represents synchronized L-fragments between lifecycles (called \textit{SL-fragments}).

\textbf{Definition 4.: (Lifecycle fragment or L-fragment, }\prec\textit{, findLf).} Given ACP model }\Pi\textit{, L-fragment }\ell_{Lx}\textit{ of lifecycle }L_x\textit{ is a nonempty connected sub-lifecycle of }L_x\textit{. It can be defined as }

\[\ell_{Lx} = (S, \Rightarrow, \Rightarrow_{\text{in}}, \Rightarrow_{\text{out}})\] \textit{where,}

\[- S \subseteq L_x, S \setminus \{s^{\text{init}}\} \cup S^f \text{ is a non-empty set of states of } \ell_{Lx}, \text{ where } S^f \text{ is a set of final states of } L_x,\]

\[- \Rightarrow \subseteq S \times R_{Lx} \times G_{Lx} \times S \subseteq L_x, \Rightarrow \text{ is a set of transitions of } \ell_{Lx}, \text{ where } R_{Lx} \text{ and } G_{Lx} \text{ are subsets of business rules and guards, respectively, defined in } L_x,\]

\[- \Rightarrow_{\text{in}} = L_x, \Rightarrow \cap ((L_x, S \setminus S) \times R_{Lx} \times G_{Lx} \times S)) \text{ is a set of entry transitions into } \ell_{Lx},\]

\[- \Rightarrow_{\text{out}} = L_x, \Rightarrow \cap (S \times R_{Lx} \times G_{Lx} \times (L_x, S \setminus S)) \text{ is a set of exit transitions from } \ell_{Lx}.\]

We denote }\ell \prec L_c\text{ (or }L_c \succ \ell\text{) if L-fragment }\ell\text{ is an L-fragment of lifecycle }L_c\text{ of artifact }C\text{. In addition, we also define function }\text{findLf}(C, S)\text{ to return an L-fragment }\ell\text{ if it can find such L-fragment that consists of a set of states }S\text{ of artifact }C\text{ such that }\ell.S = S\text{ and }\ell \prec L_c;\text{ otherwise, if an L-fragment cannot be found, it returns null.}

Notice that, providing a valid input set of states of an artifact, there is only one case that function }\text{findLf}\text{ returns a }\text{null}\text{ value – that is when such set contains only the }\text{init}\text{ state and the }\text{final}\text{ state(s) of that artifact.}

To ensure that L-fragment }\ell_{Lx}\text{ is correctly formed by a connected sub-lifecycle of its entire lifecycle }L_x\text{, }\ell_{Lx}\text{ is restricted by the following condition:}

\text{for every state }s\text{ in }\ell_{Lx}.S, \text{ there exists a sequence of transitions from some entry transition in }\Rightarrow_{\text{in}}\text{ to }s\text{ and from }s\text{ to some exit transition in }\Rightarrow_{\text{out}}\text{.}

Note that the soundness property can also apply to L-fragment providing that any L-fragment is a sub-lifecycle of its entire lifecycle. Based on the condition of entry and exit transitions of L-fragment, an entry/exit transition must be fired from/to a state inside the L-fragment. However, there is no restriction on the number of the entry state and the exit state of L-fragment.
Next, we identify a specific type of an L-fragment based on its atomicity property which is restricted by means of Single-Entry-Single-Exit (SESE) fragment of lifecycle. The SESE concept has been used for the analysis of program control-flow graphs in compiler theory [Johnson et al., 1994] and business process models [Vanhatalo et al., 2007]. However, we adapt it to our L-fragment by allowing the structure of multiple-entry transitions and multiple-exit transitions instead of single entry and single exit states. Here, we call such L-fragment as atomic L-fragment.

Definition 4. (AL-fragment, NAL-fragment). Given L-fragment \( \ell_{\text{AL-frag}} = (S, \Rightarrow, \Rightarrow_\text{in}, \Rightarrow_\text{out}) \) of lifecycle \( \ell \), \( \ell_{\text{AL-frag}} \) is called an AL-fragment if all entry transitions in \( \Rightarrow_\text{in} \) have the same source state and all exit transitions in \( \Rightarrow_\text{out} \) have the same target state. Otherwise, \( \ell_{\text{AL-frag}} \) is classified as NAL-fragment (non-atomic L-fragment).

Figure 4. shows examples of different types of L-fragments. In Figure (a), L-fragments \( \ell_1 \) and \( \ell_2 \) have single entry state \( s_1 \) and single exit state \( s_4 \); therefore, both \( \ell_1 \) and \( \ell_2 \) are AL-fragments. We can see that L-fragments \( \ell_3 \), \( \ell_4 \), and \( \ell_5 \) in Figure (b) are non-AL-fragments as they have multiple entry states or multiple exit states or both. Now, consider Figure (c), in contrast to those L-fragments, \( \ell_6 \) is not an L-fragment since entry state \( s_1 \) and exit state \( s_3 \) are used for both entry transition and exit transition of \( \ell_6 \).

Next, with the fragment of an artifact and the sync rules used within the fragment, we are able to identify counter-synchronized part(s) of the other artifact(s) that it interacts with.
Definition 4. (Sync rule for synchronized L-fragments, \( \varphi \)). Given ACP model \( \Pi \), a set of sync rules that is defined for two synchronized L-fragments \( \ell_x \) and \( \ell_y \) can be defined as follow:

\[
\varphi(\ell_x, \ell_y) = \{ r \in \Pi.R \mid \exists (s_i, r, g_k, s_j) \in \ell_x \Rightarrow \exists (s_m, r, g_l, s_n) \in \ell_y \Rightarrow \}
\]

It is noted that a sync rule is transitive, i.e., \( \exists r \in \varphi(\ell_x, \ell_y) \cap \varphi(\ell_y, \ell_z) \Rightarrow r \in \varphi(\ell_x, \ell_z) \).

Definition 4. (SL-fragment and S-region). Given ACP model \( \Pi \), we denote \( \omega = (\Gamma, R^{\text{sync}}) \) as a synchronization region (S-region) where,

— a set of synchronized L-fragments \( \Gamma = \{ \ell^{c_1}, \ell^{c_2}, ..., \ell^{c_x} \} \), \( \ell^{c_i} \in \Gamma (1 \leq i \leq x) \) is a synchronized L-fragment, called as SL-fragment, of artifact lifecycle \( L_{C_i} (C_i \in \Pi.Z) \),

— \( R^{\text{sync}} \subseteq \Pi.R \) is a set of sync rules that is exclusively used to synchronize transitions between L-fragments in \( \Gamma \) such that,

\[
\forall r \in \Pi.R, \exists \ell^{c_i}, \ell^{c_j} \in \Gamma, \forall r \in \varphi(\ell^{c_i}, \ell^{c_j}), r \in R^{\text{sync}}.
\]

Example 4.. In Figure 4. (a), we have S-region \( \omega_a \) with SL-fragments \( l_1 \) of artifact \( C_1 \) synchronized with SL-fragment \( l_2 \) of artifact \( C_2 \) via sync rules \( r_1 \) and \( r_2 \). In Figure 4. (b), S-region \( \omega_b \) has two SL-fragments \( l_3 \) and \( l_4 \) with sync rules \( R^{\text{sync}} = \{ r_1, r_2, r_3 \} \). Notice that sync rule \( r_4 \) is excluded from \( \omega_b \) as it is not exclusively used for the synchronization between \( l_3 \) and \( l_4 \), i.e., \( r_4 \) is used to synchronize transition \( s_5 \Rightarrow s_6 \) in \( l_4 \) with transition \( s_2 \Rightarrow s_9 \) which is not in \( l_3 \).

![Diagram](image-url)

Figure 4.: An example of S-regions and SL-fragments.
Next, we study the atomicity property of S-region by determining the combsability of contained SL-fragments and the boundness of their synchronization behavior.

### 4.3.3 Atomicity of synchronization region

We propose a fragmental composition technique to check the atomicity of S-region. As we can understand that the product of the composition between two synchronized L-fragments is a composite L-fragment. Then we can apply atomicity checking to the composite L-fragment. As such, we need to observe the conditions for SL-fragments that make the composite L-fragment atomic. Now, we define composite S-region based on lifecycle composition (in Definition 3.).

**Definition 4.** (Composite S-region). Given ACP model \( \Pi \), let S-region \( \omega = ([\ell^C_x, \ell^C_y], R^{sync}) \) of L-fragment \( \ell^C_x \) of artifact \( C_x \in \Pi.Z \) and L-fragment \( \ell^C_y \) of artifact \( C_y \in \Pi.Z \) where \( \ell^C_x \) and \( \ell^C_y \) are synchronized via business rules \( R^{sync} \). The composite S-region of \( \omega \), \( \otimes \omega = \ell^C_x \otimes \ell^C_y \), is tuple \((S, \Rightarrow, \Rightarrow^{in}, \Rightarrow^{out})\) where each set element in \( \otimes \omega \) has the same definition corresponding to the element of L-fragment, i.e., \( \otimes \omega \) can be considered as an L-fragment (or SL-fragment if the composite fragment still has some synchronization to other fragment of different artifact).

It is noted that a composite S-region is considered as a sub-lifecycle of the composition between two entire lifecycles. To have a (minimal and sufficient) complete view of the composition we draw a dashed arrow for a transition between a composite state excluded from the S-region and a composite state that is an entry or exit state of the S-region, as exemplified in Figure 4.. Figure 4. (a) and (b) show the results of SL-fragment composition, composite S-regions \( \otimes \omega_a \) and \( \otimes \omega_b \), for S-regions \( \omega_a \) and \( \omega_b \) in Figure 4. (a) and in Figure 4. (b), respectively.

![Figure 4.: Composite S-regions of SL-fragments in Figure 4.](image_url)
Example 4. In Figure 4., composite S-region $\oplus \omega_a$ has composite state $(s_2, s_5)$, and $(s_4, s_7)$ as its entry state and exit state, respectively. Likewise, composite S-region $\ominus \omega_b$ has two entry states $(s_2, s_5)$ and $(s_9, s_6)$, and two exit states $(s_4, s_7)$ and $(s_4, s_8)$. Note that composite state $(s_1, s_5)$ is out of scope of $\omega_a$ and $\omega_b$, so it is excluded from $\ominus \omega_a$ and $\ominus \omega_b$, respectively.

Next, we validate atomicity property of S-region by checking whether SL-fragments of the S-region can be composed into an atomic composite S-region. We consider the property of AL-fragment to define atomic S-region (AS-region), i.e., AS-region must have all entry transition fired from the same (composite) source state and all exit transitions fired to the same (composite) target state.

**Lemma 4. (Atomic Composition of SL-Fragments).** Given ACP model $\Pi$, let S-region $\omega = (\Gamma, R^{\text{sync}})$ and $Z^\ell \subseteq \Pi.Z$ be a set of artifacts where of which has its L-fragment defined in $\Gamma$. The synchronized product from the composition of all SL-fragments in $\Gamma$ is sound and satisfies the property of AL-fragment if, for every $\ell^{ci} \in \Gamma$, the following statements hold:

1. $\ell^{ci}$ is an AL-fragment,
2. $\forall \mathcal{C}_j (\mathcal{C}_j \in \Pi.Z \setminus Z^\ell), \varphi(\ell^{ci}, \mathcal{C}_j) = \emptyset$,
3. $\forall \ell^{cx}, \ell^{cy} \in \Gamma, \forall r \in \varphi(\mathcal{L}_{cx}, \mathcal{L}_{cy}), \exists s \in \ell^{ci}, S, s \xrightarrow{r} s \Rightarrow s \in \ell^{ci}$, $\Rightarrow \wedge r \in R^{\text{sync}}$,
4. $\forall \ell^{cx}, \ell^{cy} \in \Gamma, \forall r \in \varphi(\mathcal{L}_{cx}, \mathcal{L}_{cy}), \exists t \in \ell^{ci}, S, s \xrightarrow{r} s \Rightarrow s \in \ell^{ci}$, $\Rightarrow \wedge r \in R^{\text{sync}}$.

The proof of Lemma 4. can be found in the Appendix.

For the case that an S-region contains more than two SL-fragments, as similar to lifecycle composition of multiple artifact lifecycles in an ACP model, we can check the atomicity property of S-region by performing iterative composition for each SL-fragment in that S-region. It is also worth mentioning that the composition of SL-fragments holds as same characteristics as for the lifecycle composition – that is
Chapter 4. Specializing Artifact-Centric Business Processes

commutative and associative, i.e., \( \ell c_x \otimes \ell c_y = \ell c_y \otimes \ell c_x \) and \( \ell c_x \otimes (\ell c_y \otimes \ell c_z) = (\ell c_x \otimes \ell c_y) \otimes \ell c_z \).

**Definition 4.:** (AS-region and ASL-fragment). Given S-region \( \omega = (\Gamma, R^{\text{sync}}) \), if \( \omega \) holds **LEMMA 4.**, then \( \omega \) is called as AS-region and each L-fragment in \( \Gamma \) is called as ASL-fragment.

**Example 4.** In Figure 4. (a), S-region \( \omega_a \) is an AS-region as both L-fragments \( l_1 \) and \( l_2 \) are AL-fragments with all related sync rules \( (r_1 \) and \( r_2 \)) reside within them. As such, the resulted fragment from the composition between \( l_1 \) and \( l_2 \) is then atomic, as shown in Figure 4. (a). In contrast, in Figure 4. (b), we can see that L-fragment \( l_4 \) cannot satisfy the property of AL-fragment, and L-fragment \( l_3 \) does not include transition \( s_2 \xrightarrow{r_4} s_9 \) where sync rule \( r_4 \) exits in entry transition \( s_5 \xrightarrow{r_4} s_6 \) of \( l_4 \). Therefore, S-region \( \omega_b \) cannot be considered as AS-region.

**Example 4.** Figure 4. shows more examples of a variation of SL-fragments of two synchronized lifecycles introduced in Figure 4. (b). In Figure 4. (b), we can see that composite S-region \( \bigoplus \omega_c \) has two exit states \( (s_4, s_7) \) and \( (s_4, s_8) \). Therefore \( \omega_c \) is not an AS-region. In contrast, in Figure 4. (d), \( \bigotimes \omega_d \) has single entry state \( (s_2, s_5) \) and single exit state \( (s_4, s_8) \); thus, \( \omega_d \) is an AS-region.

Figure 4.: More examples of S-regions and their composition
Next, we show an example of the case where a nested L-fragment synchronizes with an L-fragment of another artifact.

Example 4. Now, we illustrate the case that an S-region contains more than two SL-fragments where one of which synchronized on the (nested) sub-fragment of SL-fragment, as shown in Figure 4. Let assume S-region \( \omega_1 \) for SL-fragments \( \{l_1, l_2, l_3\} \), \( \omega_1 \) cannot be considered as an AS-region as \( l_4 \) has some sync rules that are used for the synchronization between its sub L-fragment \( l_5 \) and L-fragment \( l_4 \) of the lifecycle of artifact \( c_4 \). Therefore, we need to include \( l_4 \) into the S-region in order to satisfy the property of AS-region.

### 4.4 Behavior-Consistent Specialization

In this section, we discuss how to specialize an ACP model while preserving the B-consistency of its base ACP model. First, we introduce the ACP specialization function that maps a specialized ACP model to its base ACP model. Then, we discuss how each of the specialization methods introduced in Section 4.2 (refinement, extension, and reduction) can be achieved with the guarantee of B-consistency.

**Definition 4.** (ACP Specialization). Given two ACP models \( \Pi = (Z, V, R) \) and \( \Pi' = (Z', V', R') \), we define a specialization relation between \( \Pi \) and \( \Pi' \) by \( ACP \) specialization function \( ps_{\Pi' \rightarrow \Pi} : Z' \cup V' \cup R' \rightarrow Z \cup V \cup R \cup \{\varepsilon\} \) such that \( ps \) is a total function mapping from each element in specialized model \( \Pi' \) onto the element
Chapter 4. Specializing Artifact-Centric Business Processes

in \( \Pi \) or empty (or null) element \( \varepsilon \). Note that we slightly abuse the use of \( ps \) to map from \( C'.S \) onto \( C.S \) where \( C' \in Z' \) and \( C \in Z \).

The refinement, extension, and reduction of artifact can be expressed by ACP specialization function \( ps \) as follows.

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

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

(b) \( \forall r'_i \in R_x', ps(r'_i) = r \),

(c) \( \forall v'_j \in V_y', \exists r'_i \in R_x', ps(v'_j) = v \land v'_j \in r'_i.V \),

(d) \( \forall s_k' \in S_z', \exists r'_i \in R_x', ps(s_k') = s \land s_k' \in pre_s(r'_i, C') \cup post_s(r'_i, C') \)

—Artifact extension. Let new artifact \( C' \in Z' \) be added in \( \Pi' \) with a set of new business rules \( R_x' \subseteq R' \setminus R \), and a set of new tasks \( V_y' \subseteq V' \setminus V \). 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 artifact \( C \) be removed from \( Z \) in \( Z' \). The following statements must be satisfied for \( ps \):

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

(b) \( \forall r_i \in R, s \in pre_s(r, C) \cup post_s(r, C), \exists r'_i \in R' \rightarrow ps(r'_i) = r_i \),

(c) \( \forall v_j \in R, v \in pre_s(r, C) \cup post_s(r, C), \exists v'_j \in V' \rightarrow ps(v'_j) = v_j \).

It is noted that if artifact, business rule, or task \( x \): \( x \in Z \cup V \cup R \) remain unchanged in the specialized model, then \( ps(x) = x \).
We can consider the specialization of an ACP model as the product of (1) the specialization of individual artifact lifecycles (called lifecycle specialization) and (2) the specialization of their synchronizations (called sync specialization) defined in the model. In Section 4.4.1, we discuss the specialization of isolated (non-synchronized) lifecycles. Then, Section 4.4.2 discusses how the inter-behavior of interacting artifacts (via sync rules) is taken into account for the ACP specialization.

### 4.4.1 Lifecycle specialization

Based on ACP specialization with our focus only on the behavioral perspective, we define lifecycle specialization to map a lifecycle of subtype to its supertype. Note that we generalize its definition to be used for both artifact classes and process models.

**Definition 4.:** (Lifecycle specialization). Let $\Pi'$ be a specialization of ACP model $\Pi$ with ACP specialization $ps_{\Pi' \rightarrow \Pi}$. Given lifecycle $L = (S, s^{\text{init}}, \Rightarrow)$ (of an artifact class in $\Pi.Z$ or $\Pi$) and lifecycle $L' = (S', s'^{\text{init}}, \Rightarrow')$ (of an artifact class in $\Pi'$ or $\Pi'$), we define lifecycle specialization relation between $L$ and $L'$ based on $ps_{\Pi' \rightarrow \Pi}$ by lifecycle specialization mapping function $ls_{\Pi' \rightarrow \Pi} : L \rightarrow L' : S' \cup s'^{\text{init}} \cup \Rightarrow' \rightarrow S \cup s^{\text{init}} \cup \Rightarrow \cup \{\varepsilon\}$, where $\varepsilon$ is an empty element. $ls_{\Pi' \rightarrow \Pi}$ is a total function. We write $ls$ without its superscription or its subscription in a clear context.

Next, we define B-consistent lifecycle specialization based on the lifecycle specialization mapping and the B-consistency (in Definition 4.).

**Definition 4.:** (B-consistent specialization, $\simeq_{ls}$). Let artifact lifecycle $L'_{c_x} = (S', s'^{\text{init}}, \Rightarrow')$ in specialized ACP model $\Pi'$ be a specialized lifecycle that inherits artifact lifecycle $L_{c_x} = (S, s^{\text{init}}, \Rightarrow)$ in ACP model $\Pi$ based on lifecycle specialization mapping function $ls_{\Pi' \rightarrow \Pi} : L'_{c_x} \rightarrow L_{c_x}$. If $L'_{c_x} \simeq L_{c_x}$, then we say that $L'_{c_x}$ is a B-consistent specialization of $L_{c_x}$, denoted as $L'_{c_x} \simeq_{ls} L_{c_x}$. Correspondingly, we also say that $ls_{\Pi' \rightarrow \Pi} : L'_{c_x} \rightarrow L_{c_x}$ is B-consistent.
For ACP specialization method by the refinement of artifact class, a single state (or a transition) is refined into a set of sub states and sub transitions. Then, we use lifecycle specialization function $l_s$ to project every state and transition of a particular fragment in a specialized lifecycle onto a single state or a single transition of its base lifecycle.

We can define lifecycle specialization based on L-fragment (in Definition 4.) called refined L-fragment that is used to refine the lifecycle, and then discuss how refined L-fragments can be used to construct a $B$-consistent specialized lifecycle, as shown in THEOREM 4..

**Definition 4.** (Refined L-fragment). Let lifecycle $L' = (S', s^{{init}'}, \Rightarrow')$ be a specialization of lifecycle $L = (S, s^{{init}}, \Rightarrow)$ by lifecycle specialization $l_s: L' ightarrow L$. We have a set of refined L-fragments that are used to refine $L$ is denoted as $lf(L' \rightarrow L) = \{\ell_1, \ell_2, ..., \ell_y\}$ where $\ell_i (1 \leq i \leq y)$ is a L-fragment in $L'$ such that $\ell_i$ does not exist in $L$.

**Example 4.** Figure 4. illustrates different scenarios of applying refined L-fragments to a base lifecycle. Lifecycles (b), (c), and (d) are different specializations of lifecycle (a). Lifecycles (b) and (c) refine some transitions of lifecycle (a) by applying a refined L-fragment $\ell_1$ and refined L-fragments $\{\ell_2, \ell_3\}$, respectively. In lifecycle (d), we can see that L-fragment $\ell_4$ is used to refine state $b$ of lifecycle (a).

**THEOREM 4.** (B-CONSISTENT REFINED L-FRAGMENT). Let artifact lifecycle $L'_c_x$ in ACP model $\Pi'$ be a specialization of artifact lifecycle $L_{c_x}$ in ACP model $\Pi$ with a set of refined L-fragments $lf(L'_c_x \rightarrow L_{c_x})$. Then, we have $L'_c_x \approx_{l_s} L_{c_x}$ if, for every refined L-fragment $\ell_i \in lf(L'_c_x \rightarrow L_{c_x})$,

- if $\ell_i$ refines transition $s_x \Rightarrow t, s_y \in L_{c_x} \Rightarrow \ell_i$ is an AL-fragment; or,

- if $\ell_i$ refines state $s \in L_{c_x}, S$ then, for every instate $s_x \in L_{c_x}, S$ fired to $s$ and for outstate $s_y \in L_{c_x}, S$ fired from $s$, $s_x$ can reach $s_y$ in some L-occurrences of $\ell_i$.

Next, we show in THEOREM 4. that the $B$-consistent refinement guarantees the soundness of the specialized lifecycle.
Chapter 4. Specializing Artifact-Centric Business Processes

Theorem 4.: (Sound and $B$-Consistent Refinement). Let ACP model $\Pi'$ be a specialization of ACP model $\Pi$ with ACP specialization $ps_{\Pi'} \rightarrow_{\Pi}$ by applying only artifact refinement with a set of refined $L$-fragments $lf(L' \rightarrow L)$ ($L'$ can be the lifecycle of any specialized artifact in $\Pi' \cdot Z$). $\Pi'$ is sound and is a $B$-consistent specialization of $\Pi$ if, for every $\ell_i \in lf(L' \rightarrow L)$, $\ell_i$ holds the condition in Theorem 4. and $\ell_i$ is sound.

The proofs of both Theorem 4. and Theorem 4. can be found in the Appendix.

Definition 4.: (Behavior-consistent specialized artifact and process). Given ACP model $\Pi'$ be a specialization of ACP model $\Pi$ with ACP specialization $ps_{\Pi'} \rightarrow_{\Pi}$, $\Pi'$ is a behavior-consistent specialization of $\Pi$ iff $ls_{L' \rightarrow L}$ is $B$-consistent. Given artifact $C' \in \Pi', Z$ be a specialization of artifact $C \in \Pi \cdot Z$ based on $ps_{\Pi'} \rightarrow_{\Pi}$, $C'$ is a behavior-consistent specialization of $C$ iff $ls_{L' \rightarrow C}$ is $B$-consistent.

4.4.2 Sync specialization

Next, we discuss how changes of artifact interactions (through their synchronization dependencies) affect the behavior of the process in their specialization at both the artifact level and the process level. Here, we classify specialization of synchronizations (sync specialization) into three primitive operations: sync extension, sync refinement, and sync reduction.

—Sync extension is a method for synchronizing new artifact with an existing artifact without refining any existing sync rule. However, it is achieved by adding a new defined set of sync rules, called extended sync rules.

—Sync refinement is a method to refine an individual existing sync rule in the base process to a new set of specialized sync rules in the specialized process. A specialized sync rule can be used to synchronize between existing artifacts or between existing artifact(s) and new (extended) artifact(s) added to the specialized process.

—Sync reduction is a method for removing synchronization between existing artifacts.
Table 4. shows the overall relations between the three construction methods of ACP specialization introduced in Section 4.2 and the three primitive sync specialization operations. Note that artifact extension generally requires both sync extension and sync refinement operations. The details of each operation will be discussed later in this section.

Table 4.: Relations between ACP specialization methods and Sync specialization

<table>
<thead>
<tr>
<th></th>
<th>Sync extension</th>
<th>Sync refinement</th>
<th>Sync reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact refinement</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Artifact extension</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Artifact reduction</td>
<td></td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>

Figure 4. shows an example of results after applying different sync specialization methods to the base process (a). The shaded artifacts represent extended artifacts. Similarly, for an existing artifact, a set of gray-shaded states and their corresponding transitions represents the refinement of its base artifact. The dash line linked between transitions of different artifacts indicates the synchronization (by mean of using sync rules) between such artifacts. More detailed discussion on this example will appear through the rest of this chapter.

The consistency of synchronization dependencies in business process specialization means whatever changes made to the synchronization of artifacts the behavior of the composed lifecycle of such specialized artifacts must be consistent with their composition in the base process. Particularly, adding a new artifact into a specialized process results in unobservable behavior of itself in the base process. Similarly, removing existing artifact from the base process in the specialized process can also have an impact on the behavioral consistency. However, it is desirable that the overall behaviors of such base and specialized processes (with added or removed artifacts) remain consistently observable. For instance, based on the purchasing process in Figure 4., the Quote artifact is added to the Offline Purchasing process should not interfere with the behavior of the Purchase Order, Shipping Order and Invoice artifacts and their interactions in its base purchasing
process. On the other hand, the removed artifact should not impact the overall behavioral consistency of the base process either.

Next, we define the specialization of synchronization between two lifecycles followed by detailed discussion on how synchronization is consistently handled when applying each of the three sync operations.

**Definition 4.: (Synchronization (Sync) specialization).** Let artifact lifecycles \( \mathcal{L}'_{c_x} \) and \( \mathcal{L}'_{c_y} \) in specialized ACP model \( \Pi' \) be a behavior-consistent specialization of artifact lifecycles \( \mathcal{L}_{c_x} \) and \( \mathcal{L}_{c_y} \) in base ACP model \( \Pi \), respectively. We define sync specialization mapping function \( s_{\Pi' \rightarrow \Pi}(\mathcal{L}'_{c_x}, \mathcal{L}'_{c_y}) \rightarrow (\mathcal{L}_{c_x}, \mathcal{L}_{c_y}) : \varphi(\mathcal{L}'_{c_x}, \mathcal{L}'_{c_y}) \rightarrow \varphi(\mathcal{L}_{c_x}, \mathcal{L}_{c_y}) \cup \{\varepsilon\} \) that projects a specialized sync rule between \( \mathcal{L}'_{c_x} \) and \( \mathcal{L}'_{c_y} \) onto its base sync rule between \( \mathcal{L}_{c_x} \) and \( \mathcal{L}_{c_y} \) or empty element \( \varepsilon \). A sync rule is unchanged if for every sync rule \( r \in \varphi(\mathcal{L}'_{c_x}, \mathcal{L}'_{c_y}), r \in \varphi(\mathcal{L}_{c_x}, \mathcal{L}_{c_y}). \)
Chapter 4. Specializing Artifact-Centric Business Processes

Note that $ss^\Pi' \to \sum (L'_{c_x}, L'_{c_y}) \to (L_{c_x}, L_{c_y})$ is a total function and we may write $ss$ without its superscription ($\Pi' \to \Pi$) and subscription $(L'_{c_x}, L'_{c_y}) \to (L_{c_x}, L_{c_y})$ in a clear context.

4.4.3 Sync extension

With a sync extension operation, we can add a new synchronized artifact to the specialized process. By doing so, we need to guarantee that the consistency is not interfered by the behavior of such added artifact. First, we discuss how the lifecycle of extended artifact and its related sync rules are taken into account in the specialization. Then, we check whether the whole lifecycle of such artifact can be completely embedded (or composed) within an L-fragment of an existing artifact it synchronizes with. This is shown in Lemma 4. on how $B$-consistency can be preserved.

We consider the case when an L-fragment of one artifact synchronizes with the entire lifecycle of another artifact. Such whole lifecycle can be considered as fully-embedded external lifecycle called ex-lifecycle in the L-fragment. We say that lifecycle $L_{c_j}$ can be fully-embedded in lifecycle $L_{c_i}$ if there exists L-fragment $\ell_{c_i} < L_{c_i}$ that completely synchronizes the entire lifecycle $L_{c_j}$, i.e., the entry state and the exit state(s) of $\ell_{c_j}$ are the init state and all the final state(s) of $L_{c_j}$, respectively.

**Definition 4. (Ex-lifecycle and $\models$).** Let artifact lifecycle $L'_{c_x}$ in specialized ACP model $\Pi'$ be B-consistent with artifact lifecycle $L_{c_x}$ in base ACP model $\Pi$, and lifecycle $L'_{c_y}$ of extended artifact $L'_{c_y}$ in $\Pi'$ synchronize $L'_{c_x}$ with lifecycle specializations $\{ l_{\ell} : L_{c_x} \to L_{c_x}, l_{\ell} : L_{c_y} \to L_{c_y} \}$ and sync specialization $ss(L'_{c_x}, L'_{c_y}) \to (L_{c_x}, L_{c_y})$ such that $ps(L'_{c_y}) = \varepsilon \land \varphi(L'_{c_x}, L'_{c_y}) \neq \emptyset$ and for every sync rule $r' \in \varphi(L'_{c_x}, L'_{c_y})$, $ss(r') = \varepsilon$. We say $L'_{c_y}$ as an ex-lifecycle of $L'_{c_x}$ if there exists refined L-fragment $\ell_i \in LF(L'_{c_x} \to L_{c_x})$ such that $L'_{c_y}$ has its whole lifecycle synchronized within $\ell_i$, denoted $\ell_i \models L'_{c_y}$, i.e.,
— for all entry transition and exit transition $\Rightarrow_\ell$ in $L'_C$, $\Rightarrow_\ell$ synchronizes with some transition in $\ell_i$; and,

— there does not exist sync rule $r'_e \in \varphi(L'_C, L'_C_y)$ such that $r'_e$ synchronizes any transition that exists in $L'_C$ with any transition that does not exist in $\ell_i$.

The synchronization of whole lifecycle (of an extended artifact) is used to ensure that every L-occurrence of the extended artifact can reach the final state within some refined (and synchronized) L-fragment. It is noted that a lifecycle of one artifact can be considered as fully-embedded external lifecycle of other multiple lifecycles.

Example 4. Extended artifact $C'_3$ in Figure 4. (b) has its whole lifecycle synchronized within artifact $C'_2$. We can see that SL-fragment $\ell'_1$ synchronizes with $\ell'_2$ which represents the whole lifecycle of $C'_3$; therefore, $C'_3$ is an ex-lifecycle of artifacts $C'_2$.

Example 4. We show a counter example of ex-lifecycle. Consider Figure 4. (b). As artifact $C_2$ cannot be entirely synchronized with any SL-fragment of artifacts $C_1$, i.e., $s_7 \Rightarrow s_8$ is not synchronized where $s_8$ is the final state of $C_2$; therefore, $C_2$ is not an ex-lifecycle of artifacts $C_1$.

In addition, we observe that: if an extended artifact terminates after all the existing artifact(s) have terminated, then the composed lifecycle (of these all artifacts) contains some L-occurrences that leads to open-ended termination inconsistency when comparing to the composition of existing artifacts. It is undesirable when such case occurs in the specialized process as the specialized process cannot terminate itself at the same state as its base process does. Therefore, we do not consider this case in our work when checking behavior-consistency. We assume that all extended artifacts must terminate before all existing artifacts terminate. Figure 4. shows an example of the case where the open-ended termination inconsistency occurs in process (b) that specializes its base process (a).
LEMMA 4. Based on Definition 4., given refined L-fragment $\ell \in l(\mathcal{L}'_{C_y} \rightarrow \mathcal{L}_{C_y})$ synchronize with extended lifecycle $\mathcal{L}'_{C_y}$, the composed lifecycle between $\mathcal{L}'_{C_y}$ and $\ell$ is $B$-consistent with $\ell$ if $\ell$ is an AL-fragment and $\ell \supseteq \mathcal{L}'_{C_y}$.

The proof of LEMMA 4. can be found in the Appendix.

From LEMMA 4., if $\ell \supseteq \mathcal{L}'_{C_y}$ and $\ell$ is an ASL-fragment, then $\mathcal{L}'_{C_y}$ is also considered as an ASL-fragment, and therefore we can construct AS-region that consists of $\mathcal{L}'_{C_y}$ and $\ell$. Now, we can see that if we can construct AS-region that consists of a refined L-fragment of one artifact and a whole lifecycle of another artifact, then we are able to have the whole lifecycle fully embedded into the refined L-fragment. We can guarantee 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 we cannot construct the AS-region then the $B$-consistency is violated.

Example 4. From Example 4., we have artifact $A'_3$ (cf. Figure 4. (a)) as an ex-life cycle of artifacts $C'_2$ with SL-fragment $\ell'_1$. Since $\ell'_1$ is considered as an ASL-fragment in $C'_2$, based on Lemma 4., the composed lifecycle between $\ell'_1$ and $C'_3$ is $B$-consistent with $\ell'_1$. The lifecycle of $C'_3$ is considered as an ASL-fragment, and we can construct AS-region of $C'_3$ and $\ell'_1$ (cf. Figure 4. (c)).

Example 4. In contrast to Example 4., in Figure 4. (b), the composed lifecycle between SL-fragment $\ell'_3$ and artifact $C'_3$ is not $B$-consistent with $\ell'_3$ as we can see that $\ell'_3$ (also lifecycle of $C'_3$) is not an ASL-fragment; thus, we cannot construct AS-region for $C'_3$ and $\ell'_3$. 
Chapter 4. Specializing Artifact-Centric Business Processes

One can question that what would be the result if an extended artifact synchronizes with more than one existing artifact. For instance, in Figure 4. (d), where two existing artifacts $C'_1$ and $C'_2$ synchronize with extended artifact $C'_3$. It is possible to see that the lifecycle of $C'_3$ is an *ex-lifecycle* of the lifecycle of $C'_1$ while it does not hold for $C'_2$. Based on *Definition 4.*, although the condition of ex-lifecycle is satisfied for the synchronization between $C'_3$ and $C'_1$, however, it is not held for the synchronization between $C'_3$ and $C'_2$. Therefore, the result of the iterative composition of such three lifecycles should not satisfy *LEMMA 4.*

4.4.4 Sync refinement

We divide sync refinement into three different sub-operations: the refinement of synchronization (1) between two existing artifacts, (2) between one existing artifact and one extended artifact, and (3) between one existing artifact and multiple extended artifacts.

4.4.4.1 Refinement of synchronization between two existing artifacts

We classify specialization patterns of the synchronization between two existing artifacts into two cases. First, one of two artifacts is refined while the other one remains unrefined. Second, both artifacts are refined. With the first case, the effected sync rule(s) of the refinement may have its state condition redefined on either the entry transition or the exit transition of a SL-fragment. For the second case, both artifacts have their SL-fragment refined.

![Sync refinements of existing artifacts.](image)

*Figure 4.*: Sync refinements of existing artifacts.

*Example 4.* Figure 4. (b) shows the sync refinement of single artifact. We can see that sync rule $r_1$ in Figure 4. (a) is refined to $r'_1-1$ and $r'_1-2$ with refined SL-
fragment in artifact $C'_1$ containing states $s_m$ and $s_n$. While Figure 4. (c) and (d) show the sync refinements of both artifacts $C'_1$ and $C'_2$.

For the refinement of two existing artifacts, we can apply AS-region to check whether the refinement of these artifacts preserves the $B$-consistency. However, for single artifact refinement, we consider it as a special case since the refinement is applied on a single transition of one artifact not a SL-fragment. In order to make the transition to qualify SL-fragment, so we expand its boundary to cover the source and target states of the transition. Then, we can validly apply the AS-region to check the $B$-consistency condition.

**Definition 4.** (Refined S-region). Given artifact lifecycles $L'_c_x$ and $L'_c_y$ in specialized ACP model $\Pi'$ be $B$-consistent with artifact lifecycles $L_{c_x}$ and $L_{c_y}$ in base ACP model $\Pi$, respectively, with two lifecycle specializations $\{ls_{L'_c_x \rightarrow L_{c_x}}, ls_{L'_c_y \rightarrow L_{c_y}}\}$ and sync specialization $ss_{(L'_c_x, L'_c_y) \rightarrow (L_{c_x}, L_{c_y})}$, such that for every sync rule $r' \in \varphi(L'_c_x, L'_c_y), ss(r') \neq \varepsilon$. Let sync rule $r \in \varphi(L_{c_x}, L_{c_y})$ be used to synchronize between transition $\Rightarrow i$ in $L_{c_x}$ and transition $\Rightarrow j$ in $L_{c_y}$. A refined S-region can be defined as follows.

—If L-fragment $\ell_m \in lf(L'_c_y \rightarrow L_{c_y})$ refining $\Rightarrow j$ synchronizes with transition $\Rightarrow i \in L'_c_x \Rightarrow$, then we can define refined S-region $\omega = \left\{ \ell_m, \ell_n \right\}$, $\varphi(\ell_m, \ell_n)$ where (unrefined) L-fragment $\ell_n$ contains a single transition $\Rightarrow i$ and its source state and target state, such that for every sync rule $r' \in \varphi(\ell_m, \ell_n), ss(r') = r$. 

—If L-fragment $\ell_m \in lf(L'_c_y \rightarrow L_{c_y})$ refining $\Rightarrow j$ synchronize with L-fragment $\ell_n \in lf(L'_c_x \rightarrow L_{c_x})$ refining $\Rightarrow i$, then we can define refined S-region $\omega = \left\{ \ell_m, \ell_n \right\}$, $\varphi(\ell_m, \ell_n)$ such that for every sync rule $r' \in \varphi(\ell_m, \ell_n), ss(r') = r$.

**Example 4.** In Figure 4. (b), S-region $\omega_b$ can be defined as a refined S-region of its base process in Figure 4. (a) based on the first statement of **Definition 4.**. It consists of the shaded L-fragment of $C'_1$, the L-fragment that is expanded from transition $(s_3 \Rightarrow s_4)$ of $C'_2$, and refined sync rules $\{r'_{1-1}, r'_{1-2}\}$. In Figure 4. (c) and (d), both refined S-region $\omega_b$ and $\omega_c$ are defined based on the second statement of **Definition 4.**.
LEMMA 4. Let artifact lifecycles $\mathcal{L}_c^x$ and $\mathcal{L}_c^y$ in specialized ACP model $\Pi'$ be $B$-consistent with artifact lifecycles $\mathcal{L}_c^x$ and $\mathcal{L}_c^y$ in base ACP model $\Pi$. Given refined $S$-region $\omega = (\ell_m, \ell_n, \varphi(\ell_m, \ell_n))$ where $\ell_m$ refines transition $\Rightarrow_i$ in $\mathcal{L}_c^x$ and $\ell_n$ refines transition $\Rightarrow_j$ in $\mathcal{L}_c^y$, if $\ell_m$ and $\ell_n$ are ASL-fragments, then the composed lifecycle of $\ell_m$ and $\ell_n$ is $B$-consistent with the composed lifecycle of $\Rightarrow_i$ and $\Rightarrow_j$ (both $\Rightarrow_i$ and $\Rightarrow_j$ are considered as L-fragments with one transition).

The proof of LEMMA 4. can be found in the Appendix.

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

Now, we extend the sync refinement between existing artifacts to be able to consider synchronizations between an existing artifact and an extended artifact. Recall sync extension, an extended artifact can be considered as an ex-lifecycle of an existing artifact if the lifecycle of the extended artifact is entirely synchronized within such existing artifact. We can say that if extended artifact $\mathcal{C}$ is used to refine sync rule $r$, then each artifact that is synchronized by $r$ must have $\mathcal{C}$ as its ex-lifecycle.

LEMMA 4. Let artifact lifecycles $\mathcal{L}_c^x$ and $\mathcal{L}_c^y$ in specialized ACP model $\Pi'$ be $B$-consistent with artifact lifecycles $\mathcal{L}_c^x$ and $\mathcal{L}_c^y$ in base ACP model $\Pi$, and let refined $S$-region $\omega = (\ell_m, \ell_n, \varphi(\ell_m, \ell_n))$ where $\ell_m$ refines transition $\Rightarrow_i$ of $\mathcal{L}_c^x$ and $\ell_n$ refines transition $\Rightarrow_j$ of $\mathcal{L}_c^y$ such that $\ell_m$ and $\ell_n$ are ASL-fragments, and extended lifecycle $\mathcal{L}_c^x$ synchronize with $\omega$. The composed lifecycle of $\ell_m$, $\ell_n$, and $\mathcal{L}_c^x$ is $B$-consistent with the composed lifecycle of $\Rightarrow_i$ and $\Rightarrow_j$ iff,

$$\ell_m \Rightarrow \mathcal{L}_c^x \land \ell_n \Rightarrow \mathcal{L}_c^x.$$

The proof of LEMMA 4. can be found in the Appendix.

Example 4. In Figure 4. (e), artifact $\mathcal{C}_3'$ is used to refine sync rule $r_1$ (between $\mathcal{C}_1'$ and $\mathcal{C}_2'$), and we can see that artifact $\mathcal{C}_3'$ is an ex-lifecycle of $\mathcal{C}_1'$ and $\mathcal{C}_3'$ is an...
Chapter 4. Specializing Artifact-Centric Business Processes

ex-lifecycle of \( C'_{2} \); therefore, the composed lifecycle of refined SL-fragments in \( C'_{1} \) and \( C'_{2} \) with \( C'_{3} \) is \( B\)-consistent with the composed lifecycle of \( C'_{1} \) and \( C'_{2} \) (given that \( C'_{1} \) and \( C'_{2} \) are only refined with \( C'_{3} \)). In contrast, in Figure 4. (d), we can see that artifact \( C'_{3} \) is an ex-lifecycle of \( C'_{1} \) but not for \( C'_{2} \). As such, the \( B\)-consistency between \( C'_{1} \) and \( C'_{2} \) cannot be preserved.

4.4.4.3 Refinement of synchronization between one existing artifact and multiple extended artifacts

We now consider the scenario that has to deal with synchronizations for multiple extended artifacts, e.g., extended artifacts \( C'_{3} \) and \( C'_{5} \) with existing artifact \( C'_{1} \) in Figure 4. (f). Similar to the refinement between an existing artifact and an extended artifact, here we extend the sync extension method and the \( B\)-consistency checking to the synchronization for multiple extended artifacts. Informally, it can be understood that if the lifecycle of extended artifact \( C'_{1} \) is entirely synchronized within artifact \( C'_{2} \) such that \( C'_{2} \) is an ex-lifecycle of artifact \( C'_{3} \), then \( C'_{1} \) can be transitively considered as an ex-lifecycle of \( C'_{3} \). Now, we define transitive ex-lifecycle to capture the transitivity of ex-lifecycles.

**Definition 4.** (Transitive ex-lifecycle and \( \sqsupseteq^{+} \)). Let artifact lifecycle \( L'_{c_{x}} \) in specialized ACP model \( \Pi' \) be \( B\)-consistent with artifact lifecycle \( L_{c_{x}} \) in base ACP model \( \Pi \), and \( L'_{c_{y}} \) and \( L'_{c_{z}} \) be lifecycles of extended artifacts \( C'_{y} \) and \( C'_{z} \), respectively, in \( \Pi' \) with lifecycle specializations \( \{ls(L'_{c_{x}}, L'_{c_{y}}), ls(L'_{c_{y}}, L'_{c_{z}})\} \) and sync specializations \( \{ss(L'_{c_{x}}, L'_{c_{y}}) \to (L_{c_{x}}, L_{c_{y}}), ss(L'_{c_{y}}, L'_{c_{z}}) \to (L_{c_{y}}, L_{c_{z}})\} \), such that,

\[
\neg ps(C'_{y}) = \varepsilon \land \varphi(L'_{c_{x}}, L'_{c_{y}}) \neq \emptyset \quad \text{and} \quad ps(C'_{y}) = \varepsilon \land \varphi(L'_{c_{y}}, L'_{c_{z}}) \neq \emptyset,
\]

\[
\forall r' \in \varphi(L'_{c_{x}}, L'_{c_{y}}) \cup \varphi(L'_{c_{y}}, L'_{c_{z}}), ss(r') = \varepsilon.
\]

We say \( L'_{c_{z}} \) as a transitive ex-lifecycle of \( L'_{c_{x}} \) if \( L'_{c_{z}} \) is an ex-lifecycle of \( L'_{c_{y}} \) and \( L'_{c_{z}} \) is an ex-lifecycle of \( L'_{c_{x}} \). Correspondingly, we write \( \ell_{i} \sqsupseteq^{+} L'_{c_{z}} \) if there exists refined L-fragment \( \ell_{i} \in lf(L'_{c_{z}} \to L_{c_{z}}) \) such that \( \ell_{i} \sqsupseteq L'_{c_{y}} \) and \( L'_{c_{z}} \) is an ex-lifecycle of \( L'_{c_{y}} \).
Chapter 4. Specializing Artifact-Centric Business Processes

Example 4. Artifact $C'_5$ in Figure 4. (f) has its whole lifecycle synchronized within the lifecycle of artifacts $C'_3$ (based on Definition 4.) and $C'_3$ is an ex-lifecycle of $C'_1$; so, we have that $C'_5$ is a transitive ex-lifecycle of $C'_1$.

Now, we extend Lemma 4. to be able to take into account the transitivity of ex-lifecycles for the $B$-consistency checking, as shown in Lemma 4..

LEMMA 4. Let artifact lifecycles $\mathcal{L}'_{c_x}$ and $\mathcal{L}'_{c_y}$ in specialized ACP model $\Pi'$ be $B$-consistent with artifact lifecycles $\mathcal{L}_{c_x}$ and $\mathcal{L}_{c_y}$ in base ACP model $\Pi$, and let refined S-region $\omega = (\{\ell_m, \ell_n\}, \varphi(\ell_m, \ell_n))$ where $\ell_m$ refines transition $\Rightarrow_i$ in $\mathcal{L}_{c_x}$ and $\ell_n$ refines transition $\Rightarrow_j$ in $\mathcal{L}_{c_y}$ such that $\ell_m$ and $\ell_n$ are ASL-fragments. Let extended lifecycle $\mathcal{L}'_{c_x}$ synchronize with $\ell_m$ or $\ell_n$ in $\omega$ and a set of extended lifecycles $Z^\text{ex}$ synchronize with $\mathcal{L}'_{c_x}$. The composed lifecycle of all artifacts in $Z^\text{ex}$, $L'_{c_x}$, $\ell_m$, and $\ell_n$ is $B$-consistent with the composed lifecycle of $\Rightarrow_i$ and $\Rightarrow_j$ iff,

$$\forall L'_{c_i}(C_i \in Z^\text{ex}), \ell_m \Rightarrow^+ L'_{c_i} \wedge \ell_n \Rightarrow^+ L'_{c_i}.$$  

The proof of LEMMA 4. can be found in the Appendix.

Example 4. From Example 4. (cf. Figure 4. (f)), as artifact $C'_5$ is a transitive ex-lifecycle of artifact $C'_1$ (via artifact $C'_3$) and $C'_5$ is also a transitive ex-lifecycle of artifact $C'_2$ (via artifact $C'_3$ and $C'_4$), therefore based on Lemma 4., the composed lifecycle of $C'_5$, $C'_3$, and refined SL-fragments of $C'_1$ and $C'_2$ is $B$-consistent with the composed lifecycle of $C'_1$ and $C'_2$ (given that $C'_1$ and $C'_2$ do not have any other refined SL-fragments).

4.4.5 Sync reduction

Based on the artifact reduction method we can remove an artifact in the specialized process model from its base process model. By doing this, all synchronizations between such removed artifact and all other related artifacts that synchronize with should be removed. The deletion of one artifact should propagate to the removal of a part of lifecycle of other artifact(s) that it synchronizes with. The deleted artifact must be guaranteed not violating the $B$-consistency of the overall process. Here, we recall
the definition of *composable SL-fragment* (in Definition 4.) and *atomic S-region* (in Definition 4.) in order to define *reduced lifecycle* of an artifact.

**Definition 4.: (Reducible L-fragment).** Let artifact lifecycle $L'C_x$ in specialized ACP model $\Pi'$ be $B$-consistent with artifact lifecycle $L_Cx$ in base ACP model $\Pi$, and let artifact lifecycle $L_{Cy}$ of $C_y \in \Pi.Z$ that synchronizes $L_Cx$ in $\Pi$ be removed from $\Pi'$ with lifecycle specializations $\{ls_{L'Cx} \rightarrow L_Cx, ls_{L'Cy} \rightarrow L_{Cy}\}$ and sync specialization $ss(L'_Cx, L'_Cy) \rightarrow (L_Cx, L_{Cy})$ such that $\forall C' \in \Pi'.Z \rightarrow ps(C') = C_y$. If there exists L-fragment $\ell$ in $L_Cx$ such that both lifecycles $\ell$ and $L_{Cy}$ are ASL-fragments and can be constructed as AS-region, then $\ell$ is a *reducible L-fragment* of $L_Cx$ for $L_{Cy}$.

**Figure 4.:** Examples of reduced lifecycles based on sync reduction.

**Example 4..** In Figure 4. (a), artifact $A_2$ is to be removed in the specialized process. We can see that L-fragment $\ell_1$ of $C_1$ synchronizes with $C_2$ and both $\ell_1$ and $C_2$ are ASL-fragments that can construct AS-region. Therefore, we have $\ell_1$ as a reducible L-fragment of $C_1$ for $C_2$. Similarly, in Figure 4. (b), L-fragment $\ell_2$ of $C_1$ and L-fragment $\ell_3$ of $C_2$ synchronize with $C_3$. We can construct one AS-region of $\ell_2$ and $C_3$ as well as another one AS-region of $\ell_3$ and $C_3$. Therefore, we have $\ell_2$ and $\ell_3$ as reducible L-fragments of $C_1$ and $C_2$, respectively, for $C_3$.

**Definition 4.: (Reduced transition).** Based on Definition 4., given artifact lifecycle $L_Cy$ be removed from specialized ACP model $\Pi'$ and a set of artifact lifecycles $L^{ex}$ synchronize with $L_{Cy}$, if, for every artifact lifecycle $L_{Ci}$ in $L^{ex}$, there exists *reducible L-fragment* $\ell^{Ci}$ of $L_{Ci}$ for $L_{Cy}$, then $\ell^{Ci}$ is reduced to *reduced transition* $s_i \Rightarrow \ell^{Ci} s_j$, where $s_i$ is the state entering to $\ell^{Ci}$ and $s_j$ is the exit state of $\ell^{Ci}$. 

95
Example 4. Based on Example 4., in Figure 4. (a), ASL-fragment \( \ell_1 \) of \( C_1 \) is reduced to reduced transition \( s_1 \xrightarrow{\ell_1} s_3 \) in \( C'_1 \). Similarly, in Figure 4. (b), ASL-fragments \( \ell_2 \) of \( C_1 \) and \( \ell_3 \) of \( C_2 \) are reduced to reduced transition \( s_1 \xrightarrow{\ell_2} s_4 \) in \( C'_1 \) and \( s_1 \xrightarrow{\ell_3} s_3 \) in \( C'_2 \), respectively.

Once an artifact is removed and an effected lifecycle is reduced, all sync rules used within that lifecycle must also be either reduced or removed.

Definition 4: (Reduced sync rule). Based on Definition 4., given a set of reducible L-fragments \( L^{re} \) for removed artifact \( L_{cy} \), for every reducible L-fragment \( \ell_m \) in \( L^{re} \), for every sync rule \( r \in \varphi(\ell_m, L_{cy}) \), \( r \) needs to be:

— removed if \( \ell_m \) synchronizes with only \( L_{cy} \) and \( \ell_m \) does not synchronize with other L-fragment in \( L^{re} \); or,

— reduced into reduced sync rule \( r_\ell \) where \( r_\ell \) is used to synchronize between transition \( \Rightarrow \ell_m \) and transition \( \Rightarrow \ell_n \) if there exists \( \ell_n \) in \( L^{re} \) such that \( \ell_m \) synchronizes with \( \ell_n \).

Example 4. In Figure 4. (a), artifact \( C_2 \) is removed in the specialized process. The result of removal propagate to the reduction of an L-fragment containing state \( s_2 \) with its entering and existing transitions that synchronize \( C_2 \). We can see that such L-fragment is reduced to a single transition \( (s_1 \xrightarrow{\ell_1} s_3) \) in \( C'_1 \). All sync rules that are used by \( C_1 \) and \( C_2 \) are removed based on the first condition of Definition 4..

Example 4. In Figure 4. (b), artifact \( C_3 \) is removed in the specialized process and the specialized process consists of only artifacts \( C'_1 \) and \( C'_2 \) where the effected ASL-fragments of both artifacts are reduced into reduced transitions. All synchronizations among \( C_1, C_2, \) and \( C_3 \) are reduced to reduced sync rule \( r'_{\ell_x} \) based on the second condition of Definition 4.. This reduced sync rule is used to synchronize between a reduced transition resulted from the reduction of L-fragment in \( C'_1 \) and a reduced transition resulted from the reduction of L-fragment in \( C'_2 \).

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

**Definition 4.** *(Propagated sync reduction).* Based on Definition 4., given a set of reducible L-fragments \( L^r_e \) for removed artifact \( \mathcal{L}_{C_y} \), for every reducible L-fragment \( \ell_m \) in \( L^r_e \), if there exists L-fragment \( \ell_n \) in \( \mathcal{L}_{C_z} \) of artifact \( C_z \) in the base ACP model such that \( \ell_n \) is an ASL-fragment, then \( \ell_n \) is reducible L-fragment of \( \mathcal{L}_{C_z} \) for \( \mathcal{L}_{C_y} \). We can say that the removal of \( \mathcal{L}_{C_y} \) propagates to the reduction of \( \ell_n \) via \( \ell_m \).

![Figure 4. Example of propagated sync reduction.](image)

**Example 4.** In Figure 4., artifact \( C_3 \) is to be removed in the specialized process. We can see that both L-fragments \( \ell_1 \) of artifact \( C_1 \) and \( \ell_2 \) of artifact \( C_2 \) are reducible L-fragments, and \( \ell_1 \) has its sub L-fragment (containing state \( s_4 \)) synchronized with L-fragment \( \ell_3 \) of \( C_4 \). Such sub L-fragment in \( C_1 \) and \( \ell_3 \) can be considered as ASL-fragments and can be constructed as AS-region. The removal of artifact \( C_3 \) not only causes the reduction of \( \ell_2 \) but also propagates to the reduction of \( \ell_3 \) as well. Therefore, we have \( \ell_3 \) reduced into reduced transition \( s_1 \Rightarrow s_3 \) in \( C'_4 \) and all sync rules defined in \( \ell_3 \) reduced to reduced sync rule \( r'_y \) for the synchronization between \( C'_1 \) and \( C'_4 \).

**Lemma 4.** Let artifact lifecycle \( \mathcal{L}_{C_y} \) that synchronizes with artifact lifecycle \( \mathcal{L}_{C_x} \) in base ACP model \( \Pi \) be removed from specialized ACP model \( \Pi' \). Given \( L^r_e \) be a set of all possible reducible L-fragments for \( \mathcal{L}_{C_y} \) and \( T^r_e \) be a set of reduced transitions, the composed lifecycle of all transitions in \( T^r_e \) is \( B\)-consistent with the composed lifecycle of all L-fragments in \( L^r_e \) and \( \mathcal{L}_{C_y} \).

The proof of **Lemma 4.** can be found in the Appendix.
Chapter 4. Specializing Artifact-Centric Business Processes

4.4.6 Sync specialization consistency and B-consistency

Based on our comprehensive discussion on the three operations of sync specializations and their consistency conditions, we now are able to define a complete sync consistency (S-consistency) property of ACP model by taking sync specialization into account.

**Definition 4.** (S-consistent). Given ACP model \( \Pi' \) be a specialization of ACP model \( \Pi \) with ACP specialization \( p_{\Pi' \rightarrow \Pi} \) and sync specialization \( SS(\ell'_c, \ell'_c) \rightarrow (\ell_c, \ell_c) \), \( SS(\ell'_c, \ell'_c) \rightarrow (\ell_c, \ell_c) \) is said to be S-consistent iff,

—Lemma 4. is held for sync extension based on Definition 4.,

—Lemma 4. and Lemma 4. are held for sync refinement based on Definition 4. and Definition 4., respectively,

—Lemma 4. is held for sync reduction based on Definition 4..

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

**Theorem 4.** Let ACP model \( \Pi' \) specialize ACP model \( \Pi \) with ACP specialization \( p_{\Pi' \rightarrow \Pi} \), \( \Pi' \) is a behavior-consistent specialization of \( \Pi \) based on \( ps \) iff,

—for every artifact \( C'_i \in \Pi'.Z \) such that \( ps(C'_i) = C_i \in \Pi.Z \), \( ls_{C'_i} \rightarrow \ell_c \) is B-consistent,

—for every artifact \( C'_x \) and \( C'_y \) in \( \Pi' \), \( ss(\ell'_c, \ell'_c) \rightarrow (\ell_c, \ell_c) \) is S-consistent.

We can see that Theorem 4. has an importance of being able to assert the overall behavioral consistency between a specialized ACP model and its base model while only perform fragmental consistency checking based on a specialization, i.e., for an individual artifact and for only a synchronization between artifacts that is added, modified, or deleted in the specialized process. Notably, the model verification can suffer from the state exposition of compositional lifecycle if there are a number of artifacts having many states. Technically, we avoid the state space exposition problem by not composing all artifacts in the model.
Chapter 4. Specializing Artifact-Centric Business Processes

THEOREM 4.. Given ACP model $\Pi'$ be a specialization of ACP model $\Pi$, if for every artifact $C' \in \Pi'.Z$ and every artifact $C \in \Pi.Z$ such that $C'$ is a behavior-consistent specialization of $C$, then $\Pi'$ needs not to be a behavior-consistent specialization of $\Pi$.

The proofs of both THEOREM 4. and THEOREM 4. can be found in the Appendix.

4.5 Case Study

In this section, we illustrate how our business process specialization approach can be used for reusing existing purchasing processes between buyers and sellers (or suppliers). At the beginning, a supplier generally designs and uses purchasing process for a wide range of processing orders requested from buyers. The supplier and the buyers identify their three core business artifacts that are involved in the process: Purchase Order, Shipping Order, and Invoice. They also define their business rules that specify the logic of the process together with a set of tasks required to create/update those artifacts. The artifacts and their interaction behavior for this process are shown in Figure 4..

![Figure 4: Generic purchasing process.](image)

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 still keeping their original processes for the purpose of historical analysis and
reporting. Based on the new requirements, both the buyers and the supplier have to design 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 4.. 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.

![Diagram of Offline ordering process](image)

**Figure 4.:** Offline ordering process.

In Figure 4., we can see that *Offline Purchase Order* and *Offline Invoice* specialize *Purchase Order* and *Invoice*, respectively, in the generic purchasing process (cf. Figure 4.). The specialization of these two artifacts is achieved by applying artifact refinement method (represented by the use of atomic L-fragments $\ell_1$ and $\ell_2$ on *Offline Purchase Order* and refined L-fragment $\ell_3$ on *Offline Invoice*). Moreover, *Quote*, *Picking List*, and *Shipping List* are introduced and added in the offline purchasing process in which they can be achieved by applying artifact
extension method. The *Quote* and *Picking List* artifacts synchronize with *Offline Purchase Order* within AL-fragments $\ell_1$ and $\ell_2$, 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 similar to 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 composed lifecycle of all artifacts in the generic purchasing process, then we will have the lifecycle of the offline purchasing process covers the lifecycle of the generic purchasing process. Thus, we have the lifecycle of the former *B-consistent* with the lifecycle of the latter, i.e., the former is a behavior-consistent specialization of the latter.

### 4.6 Summary and Discussion

In this chapter, we presented how our process specialization approach that 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 higher degree of flexibility and modularity than traditional activity-centric business process modeling approaches. As discussed, the object specialization considers on 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 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 analyzed the interaction (synchronization) behavior of artifacts and showed how to ensure their consistency in a derived process via the use of atomicity property (of S-region) to preserve the synchronous behavioral consistency between artifacts in their original process and in the derived process. Our notion of
Chapter 4. Specializing Artifact-Centric Business Processes

*B-consistency* can be used to guarantee the weak bi-simulation behavioral consistency between two processes.

By applying our behavior-consistent restriction to the specialization mechanism, it not only improves the modeling and design efficiency of business processes, but also allows efficient runtime management capabilities. We are capable to aggregate data and processes and view them in more abstract level. This feature facilitates consistent monitoring and reporting of business data and processes. The major outcome of this chapter is the formal studies on the conditions for preserving the behavior consistencies of both intra-behavior and inter-behavior of artifacts in a specialized process based on our three proposed specialization methods (extension, refinement, and reduction).
Over the past several years, there have been increasing needs for more efficient approaches for the design, model, implementation, and management of inter-organizational business processes. In general, modeling business processes spanning multiple organizations can be inevitably complex, thus difficult to manage. All involving organizations should be able to agree on their business process at an abstract level and to participate towards the completion of the business process, where each party has its own freedom of modifying its internal operations to meet its private goals while satisfying the mutual objectives with its partners.
Although there are some existing works that explored how the artifact-centric approach can be utilized to support collaborations between organizations, the research in this area is still in its infancy stage. To address the three major collaboration requirements (compliance, flexibility, and autonomy) in an artifact-centric paradigm as introduced in Section 1.2.2, we propose a view-based approach for modeling and change validation of inter-organizational business processes. We present three core components in our approach: (1) artifact-centric inter-organizational business process models based on our ACP model, (2) public/private process views, (3) and changes and consistent change validation mechanisms that can support the participating organization's customization towards their internal operations while ensuring the correctness of collaborating processes. Technically, we use our B-consistency and AS-region notions proposed in Chapter 4 to support process view construction and behavioral consistency validation between a process and its views.

This chapter is organized as follows. In Section 5.1, we introduce our motivating example and discuss the problems addressed for efficient inter-organizational collaboration. Section 5.2 presents our view-based approach and discusses how it can be used to tackle the problems. Section 5.3 introduces the notion of public view for artifact-centric inter-organizational business processes. Section 5.4 introduces change validation mechanism via the notion of private views. Section 5.5 summarizes and discusses the view-based approach.

5.1 Motivating Example and Problem Analysis

5.1.1 Motivating example

In this section, we take supply chain collaboration (purchasing process) as an example to illustrate and motivate the artifact-centric approach to modeling inter-organizational business processes. Figure 5. illustrates a complete purchasing process model in the collaboration extended from our generic purchasing process introduced in Section 3.1. The process involves three roles of participating organizations: Buyer, Supplier, and Logistics. We initiate the discussion of this
example by identifying involved business artifacts and describing how they can be modeled for this inter-organizational process collaboration.

We believe that at the early stage all parties identify and model their required business artifacts of their local process. This step includes defining organization-owned artifacts (called local artifacts) that are internally used/managed by individual party as well as their common agreed artifacts (called shared artifacts) that are used for the coordination between parties in the collaboration. We can understand that the lifecycle of a shared artifact should represent its agreed business stages and possible steps towards the completion of the process. In other words, shared artifacts should be used as a mutual form of interests of all parties; therefore, the coordination occurs at some points where they are processed. In the implementation, these shared artifacts act as messages that are sent and received by these organizations. Based upon the current processing state of the shared artifact, a responsible organization that has received this artifact will invoke a specified task (or service) according to a corresponding business rule defined by that organization. This task will read or update the artifact and other local artifacts defined in the specification.

In Figure 5., we can see that the inter-organizational process consists of Purchase Order (PO), Shipping Order (SO), and Invoice (IV) as shared artifacts. Apart from such three artifacts, Buyer, Supplier, and Logistics have Quote (Q) and Payment (P), Picking List (PL) and Delivery Note (DN), and Shipping List (SL) as their local artifacts, respectively.
Figure 5: An overall view of artifact-centric purchasing processes

Now, let us briefly describe the process in Figure 5 in more details. At the beginning of the process, a buyer creates its quote and PO documents. Once the quote is approved, the PO is confirmed and sent to a selected supplier. When the supplier receives the PO, it creates a PL document for the purpose of acquiring goods for that PO. If the goods run out, then the supplier rejects to supply them and then cancels the PO. Otherwise, the goods are filled, and then the supplier generates an internal DN document and creates a SO document for the designate logistics company. Once the SO is received, the logistics company creates a SL document that is used for picking up the goods from the supplier’s shipping point and also delivers the goods to the buyer. After that, the supplier creates an IV document and sends it to the buyer. Sometime later, the buyer clears the total amount owing in the IV, consequently, the supplier marks the PO as closed. This purchasing process completes when the PO is in the closed state, the SO is in the arrived state, and the IV is in the cleared state.

5.1.2 Problem analysis

Based on the defined sets of business artifacts, tasks, and business rules to associate artifacts with tasks, we can derive a complete inter-organizational
process. Although this approach is practical for intra-organizational business
process modeling, it is not feasible for inter-organizational processes due some
major concerns in the way organizations collaborate. Similar concerns have been in
fact raised in the traditional activity-centric approach [Van Der Aalst and Weske,
2001; Ghattas and Soffer, 2009].

First of all, organizations concern about their privacy. In fact, organizations are
not willing to reveal their private or sensitive data (or internal processes) to
outsiders. They prefer protecting their privacy as at the maximum level as they
can. However, in the modeling phase of the collaboration, it is necessary for the
participating organizations to reveal certain details of their internal processes at an
adequate level of visibility as to establish an overall picture of the collaboration,
which is used as a process agreement or contract among them. In our artifact-
centric inter-organizational process modeling approach, the level of visibility is
determined based on the type of artifacts, i.e., details of local artifacts should be
kept invisible to external parties as much as possible while they also support the
overall process by means of dependency associations with some processing part
(lifecycle) of the shared artifact(s). Consider the DN artifact in Figure 5. for
instance. The DN is privately used by Supplier; however, we can see that it has
dependency associations with the sub-lifecycles of the PO (filled \(\rightarrow\) ready_to_ship \(\rightarrow\)
dispatched) and SO (\(\bigcirc\) \(\rightarrow\) created \(\rightarrow\) scheduled \(\rightarrow\) in_transit), which are shared
artifacts. Similarly, both Quote and PL artifacts are used to support some
processing steps of the PO artifact. To achieve privacy, we assume that if a
processing step of a shared artifact is exclusively controlled by the local artifact(s)
of one party, then that step should not be visible to other parties. For example, the
ready_to_ship state and its related transitions of PO should be kept invisible to
external parties because DN is a local artifact of Supplier. Apart from that, we can
see that the step created \(\rightarrow\) scheduled of SO should be hidden to external parties as
well. In order to support the privacy requirement, an organization needs to have a
mechanism to identify an artifact and/or its parts that need to be revealed to the
other parties. It is worth noting that organizations should not provide too low level
of visibility, i.e., if they do not disclose adequate information about their local
process that supports shared artifact(s), then the successful coordination among
them may not be achieved. This brings in the question of how much extent of local processes can be kept private while not affecting the successful establishment of the collaboration.

Second, organizations prefer to keep the freedom of modifying their internal processes without revealing the changes to other parties. In other word, once they have agreed on the overall process, they should have, possibly, the highest level of flexibility to modify their own local process while remaining autonomous in the collaboration [Van der Aalst and Wake, 2001; Van der Aalst et al., 2007, 2010]. For example, Supplier may modify some processing steps of its PL artifact, and such change should not be disclosed to Buyer and Logistics. Apart from the change of existing local artifact, they should be allowed to add new local artifacts to their process caused by process expansion/improvement. Revisiting Figure 5., Supplier may need to incorporate Inventory List (IL), which is used for inventory management, to their process. The IL artifact needs to interact with existing local artifact PL. It also implies that IL indirectly contributes to the part of shared artifact PO through PL. In order to support the change of local processes, organizations should be able to identify what they want to modify and how such modification can be applied.

Third, stemmed from the modification freedom, the change to any local process should not affect the behavior of the overall collaboration. This implies that a local process always complies with the contract agreed by all parties. An organization does not care about the change that incurs within the other parties; however, the change made to local process should not affect the overall process. Therefore, all parties must ensure that their local changes do not violate the collaboration contract. For example in Figure 5., if Supplier changes their PL artifact by removing ready_to_fill state, then this directly affects transition acquiring → filled of PO; consequently, Logistics is not able determine whether goods of such PO are ready to be picked for delivery. In addition, they should also be aware of the propagation of local changes that will eventually affect the overall process although the changes made in a local artifact do not directly affect any shared artifact. For instance, if there are some changes made on IL artifact, then they directly affect the PL artifact and also propagate to the PO artifact. This raises the issue of how we
can guarantee that local changes made by individual party do not lead to the incorrect behavior of the overall collaboration. In the other words, organizations must ensure that changes in their local processes are in compliance with what they have promised to provide.

The three major concerns discussed above call for an approach to efficient modeling and change validation of artifact-centric inter-organizational business processes. As previously discussed, in this thesis, we study how organizations can apply the concept of views to support and facilitate the modeling of their collaborating business processes in an artifact-centric paradigm. Particularly, we borrow the idea of view approach to inter-organizational workflows which was originally studied in activity-centric business process modeling approaches [Van der Aalst and Weske, 2001; Van der Aalst et al., 2007, 2010], and then explore it in the context of artifact-centric processes.

5.2 Artifact-Centric Process View Framework

In this section, we introduce and discuss our view framework for artifact-centric inter-organizational business processes. In Section 5.2.1, we present the overview of our view framework that aims at addressing the aforementioned requirements for efficient inter-organizational business processes collaboration. Then, we discuss about how inter-organizational business processes can be structured by using artifact-centric modeling approach in Section 5.2.2 followed by the discussion of the $B$-consistency for the purpose of process views and change validation in Section 5.2.3.

5.2.1 Overview

We start this section by introducing our view framework for modeling artifact-centric collaborating business processes - which is influenced by the process-oriented contract approach proposed for service choreographies [Van der Aalst et al., 2010]. In the context of this thesis, we use term inter-organizational business processes to mean the collaboration in the thesis as the collaborating processes are
internally and externally distributed in their nature. Here, our view framework consists of: (1) artifact-centric business process model for inter-organizational business processes (*ACP-i model*) which is extended from the *ACP model*, (2) public view and its construction method, (3) process change mechanism, and (4) process change validation. Figure 5. depicts the overall picture of the framework by taking our motivating purchasing processes (introduced in Section 5.1.1) as an illustrative example.

![Figure 5.: View framework for artifact-centric inter-organizational processes](image)

In summary, our view framework is developed based on the notion of *private views* and *public views*. A *private view* is used to capture the local process of each individual organization, while a *public view* of a particular collaboration process is an abstract representation of the overall process that is necessary for the coordination and protects the private processes of each organization as much as possible. Organizations can achieve an efficient collaboration (regarding the three requirements) based on public-private views approach by the following steps.

—Construct a complete artifact-centric model specification of inter-organizational business process
—Create a public view that is served as a mutual agreement, i.e., contract, of the collaboration.

—Each organization can change and validate their local processes, via the use of private views, without the need for global verification.

First of all, all participating parties specify a complete specification of their inter-organizational business process as to achieve their goal of the collaboration. As previously discussed, the coordination among them can be specified by defining all shared artifacts and their interactions with local artifacts (from each party). We name ACP-i model for the artifact-centric model of the complete inter-organizational process. Once all participating organizations have agreed on the complete view of artifact-centric collaboration process, they construct an agreed public view of such process. This public view reveals only the necessary information of artifacts that is required to be used for the coordination among parties. In the perspective of artifact-centric process modeling, we can consider that a public view should only contain shared artifacts and some necessary parts of local artifacts that interact with shared artifacts. As already discussed in the privacy requirement, the public view should not reveal the part of a shared artifact that is supported by private local artifacts. Similarly, if an entire local artifact can be hidden without incurring any inconsistency to the public view, then it should not be included in the view.

Once the public view is built as a contract, each participating party has the responsibility to ensure that its local processes (i.e., responsible parts of shared artifacts and their local artifacts) can provide what has been specified in the contract to other parties involved in the collaboration. The most important things are: (1) the constructed public view must be sound and consistent with its base process; and, (2) when an organization modifies its own parts of both types of artifacts used in the collaboration, it must guarantee that such local changes do not compromise the correctness of the overall collaboration. We will discuss the construction of public view and changes in local process in more detail in Section 5.3 and Section 5.4, respectively.
5.2.2 ACP-i model for inter-organizational business process

Now, we formally define the artifact-centric process model for inter-organizational business processes (ACP-i model) which is an extended version of the ACP model (defined in Definition 3). An ACP-i model consists of three core components of the ACP model: artifacts, tasks, and business rules, and one additional component: roles, where roles define a set of participating organization roles in the collaboration. An ACP-i model is used to capture the complete specification of a particular inter-organizational business process in the collaboration. As previously discussed, shared artifacts should be used as coordination means in inter-organizational business processes. However, a shared artifact cannot be completely modeled solely by a single organization due to its sharing nature. Therefore, all participating organizations must provide and agree on both the structure (data model) and the behavior (lifecycle) of the shared artifacts in order to define their ACP-i model. Importantly, in order to maintain the integrity of a shared artifact together with all corresponding tasks and business rules that are involved with the artifact, we assume that it must be mutually understood that the name of its state and attribute must be uniquely identified for the same mean across organizations in the collaboration.

Definition 5.: (Organization Role). We denote organization roles \( L = \{l_1, l_2, ..., l_x\} \), where \( l_i \in L(1 \leq i \leq x) \) is a role of organizations that participate in an inter-organizational business process.

Definition 5.: (ACP-i model). Given a set of organization roles \( L \) involved in a collaboration, we define an ACP-i model, denoted as \( \Pi = (Z, V, R, L, \gamma) \), for their inter-organizational business process where,

- \( Z \) is an artifact schema, \( V \) is a set of tasks, and \( R \) is a set of business rules,

- \( L \) is a set of participating organization roles,

- \( \gamma: Z \cup V \cup R \rightarrow 2^L \) is a role mapping function from an artifact class, a business rule, or a task onto organization role(s) as follows:
(a) $\gamma(C)$ returns a set of roles $\{l_1, l_2, \ldots, l_x\}$, where $l_i \in L (1 \leq i \leq x)$ is a role that can access (read/write) to artifact $C \in Z$. Note that shared artifact $C$ implies $|\gamma(C)| > 1$, and if $C$ is a local artifact then $|\gamma(C)| = 1$.

(b) $\gamma(v)$ returns role $l \in L$ of organization who owns task $v \in V$. Note that a task can perform read/write operations on either local artifact or shared artifact or both.

(c) $\gamma(r)$ returns role $l \in L$ of organization who owns business rule $r \in R$.

Table 5. shows an example of business rules defined in the ACP-i model of our inter-organizational purchasing process. We can see that a role of organizations is included to a business rule.

Table 5.: Example of business rules for inter-organizational purchasing process.

<table>
<thead>
<tr>
<th>$r_1$</th>
<th>Buyer approves Quote $q$ to confirm Purchase Order $po$ for a selected Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>$\text{instate}(q, \text{approving}) \land \text{instate}(po, \text{on_hold}) \land \text{defined}(po, \text{OrderID}) \land \text{defined}(po, \text{SupplierID})$</td>
</tr>
<tr>
<td>Task</td>
<td>$\text{approve}(q, po)$</td>
</tr>
<tr>
<td>Role: Buyer</td>
<td></td>
</tr>
<tr>
<td>Post-condition</td>
<td>$\text{instate}(q, \text{approved}) \land \text{instate}(po, \text{confirmed}) \land \text{defined}(po, \text{SubmitDate})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$r_2$</th>
<th>Supplier accepts Purchase Order $po$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>$\text{instate}(po, \text{confirmed}) \land \text{defined}(po, \text{OrderID}) \land \text{defined}(po, \text{SupplierID})$</td>
</tr>
<tr>
<td>Task</td>
<td>$\text{acceptPO}(po)$</td>
</tr>
<tr>
<td>Role: Supplier</td>
<td></td>
</tr>
<tr>
<td>Post-condition</td>
<td>$\text{instate}(po, \text{accepted})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$r_3$</th>
<th>Supplier creates Shipping Order $so$ from Delivery Note $dn$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>$\text{instate}(dn, \text{prepared}) \land \text{defined}(dn, \text{ShipperID}) \land \text{instate}(so, \text{init})$</td>
</tr>
<tr>
<td>Task</td>
<td>$\text{createShipping}(dn, so)$</td>
</tr>
<tr>
<td>Role: Supplier</td>
<td></td>
</tr>
<tr>
<td>Post-condition</td>
<td>$\text{instate}(dn, \text{transferring}) \land \text{instate}(so, \text{created}) \land \text{defined}(so, \text{ShippingID})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$r_4$</th>
<th>Supplier dispatches goods for Purchase Order $po$ that to be shipped by Shipping Order $so$, and simultaneously issues Invoice $iv$ to the Buyer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>$\text{instate}(po, \text{ready_to_ship}) \land \text{instate}(dn, \text{transferring}) \land \text{instate}(so, \text{scheduled}) \land \text{instate}(iv, \text{init})$</td>
</tr>
</tbody>
</table>
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

<table>
<thead>
<tr>
<th>Task</th>
<th>dispatchGoods(po, dn, so)</th>
<th>Role: Supplier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>issueInvoice(po, iv)</td>
<td></td>
</tr>
<tr>
<td>Post-condition</td>
<td>instate(po, delivering) ∧ instate(dn, dispatched) ∧ instate(so, in_transit) ∧ instate(iv, issued) ∧ defined(iv.InvoiceID) ∧ defined(iv.OrderID)</td>
<td></td>
</tr>
</tbody>
</table>

Given an ACP-i model, we can derive a local ACP model for an organization’s local process. Note that in the local process of an organization, the attributes and states of its shared artifact can be obtained from the ACP-i model if they are specified in the business rules of local process.

**Definition 5.** (local ACP model). Given \( \Phi = (Z, V, R, L, \gamma) \) be an ACP-i model, a local ACP model of role \( l \in L \) can be obtained by deriving from \( \Phi \), and it is defined as \( \Phi_l = (Z^l, V^l, R^l) \) where,

- \( Z^l = \{ C \in Z \mid l \in \gamma(C) \} \) is a local artifact schema, such that each shared artifact in \( Z^l \) contains only a state that is defined in the pre- or post-condition of business rule in \( R^l \), i.e.,
  \[
  \forall C \in \{ C \in Z^l \mid l \in \gamma(C) \} \wedge |\gamma(C)| > 1, \forall s \in \mathcal{C}, s \in \text{pre}_s(r, C) \cup \text{post}_s(r, C),
  \]
- \( V^l = \{ v \in V \mid \gamma(v) = l \} \) is a set of local tasks,
- \( R^l = \{ r \in Z \mid \gamma(r) = l \} \) is a set of local business rules.

**Example 5.** Figure 5. shows the Supplier’s local ACP model that derives from the ACP-i model of the purchasing process illustrated in Figure 5. We can see that shared artifacts PO and SO represent only the parts that belong to the Supplier’s local process, e.g., some processing steps of PO (before the confirm state) and SO (after the in_transit state) that belong to Buyer and Logistics, respectively, are not captured in the Supplier’s local ACP model.
Now, consider the behavioral property of a shared artifact defined in local ACP model. It is always true that the local ACP model is not goal-reachable as the lifecycle of shared artifact is partially modelled and can be non-terminated. However, when integrating all the different parts of a shared artifact from each organization, the complete lifecycle must be goal-reachable.

### 5.2.3 B-consistency for process views and process changes

We use the notion of B-consistency (defined in Definition 4.) for two applications in this chapter: process abstraction and process change, as illustrated in Figure 5.

Process abstraction is the method used for constructing a public view from the complete process model. Process change is the method used to allow an organization to modify and changes its own local process. The behavior of a constructed public view and a changed process can be validated against its base process via the B-consistency checking.
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

Figure 5.: Two types of applications for the B-Consistency notion

With the use of B-consistency notion, we define how the two methods (i.e., process abstraction and process changes) can be performed on an ACP-i model while preserving the behavioral consistency property of its base model. The discussions of these two methods with the B-consistency will be detailed in Sections 5.3 and 5.4.

5.3 Public View Construction

In this section, first of all, we introduce the public view definition and its construction as well as discuss behavioral consistency between the constructed view and its base ACP-i model. Second, we present the abstraction methods to construct the public view that preserves the behavioral consistency. Third, we discuss how synchronization dependency between artifacts is affected by an abstraction. Last, we propose a technique for constructing the minimal public view of the collaboration and introduce an algorithm for finding the best possible solution to protect the information (i.e., local processes) of every organization at the highest level of privacy and autonomy.

5.3.1 Constructing public views and view B-consistency

Generally, a public view of a particular collaboration should be constructed by taking into account all parts that coordination between participating organizations takes place in the collaboration, particularly where the processing and exchanging
of shared artifacts occur. From the overall view of the complete process collaboration, organizations should demand their public view to consist of only necessary information and steps that are required to coordinate them towards the completion of the collaboration. Shared artifacts are the main concern as they are used by more than one organization. This requirement raises the question of how to decide which processing part of a shared artifact should be abstracted such that the abstracted part does not affect the coordination. Based on this requirement, we observe that the part of a shared artifact that interacts with local artifact(s) and a processing step of shared artifact that are owned by a single organization should be taken into account for the abstraction. As discussed in Section 0, this is because such part is deemed as local processing of the shared artifact that should not be revealed to other parties (due to out of interest and privacy concern). As such, a constructed public view should have all local artifacts of every organization hidden as much as possible and have all parts of shared artifacts that interact with local artifacts abstracted.

Next, we discuss the ACP abstraction method for the construction of the public view that derives from an underlying ACP-i model. The method is discussed based on the following two points.

—Abstraction of non-synchronized part of lifecycle. It can be easily understood that the abstraction of artifact’s lifecycle can be achieved by abstracting a part of the lifecycle (L-fragment) into either a single state or a single transition. A general technique for state/transition abstraction can be applied directly to a lifecycle fragment of individual artifact. We will discuss this technique in detail in Section 5.3.2.

—Abstraction of synchronized part of lifecycle. Apart from the isolate abstraction, we also consider how two or more synchronized fragments (via sync rules) of different artifact lifecycles can be abstracted. We observe that the abstraction of one synchronized end should require the abstraction of the other end due to preservation of the consistency of their original lifecycles. Both abstracted fragments of two lifecycles must still somehow be either correctly synchronized or none. If an entire lifecycle that synchronizes with a fragment of another lifecycle
is to be abstracted, then the former is totally unsynchronized and should be considered as an embedded fragment of the abstracted lifecycle of the latter. Section 5.3.3 provides a discussion on this issue in more detail.

Based on an ACP-i model, organizations can construct their public view by abstracting their shared artifacts and their local artifacts. We define a public view by applying the abstraction function to map a complete ACP-i model to its public view.

**Definition 5.** (Public view and ACP abstraction). Given ACP-i model $\bar{\Pi} = (Z, V, R, L, \gamma)$, the public view of $\bar{\Pi}$ can be denoted as $pa_{\bar{\Pi}} = (Z^p, V^p, R^p, L^p, \gamma^p)$, where $Z^p, V^p, R^p, L^p, \gamma^p$ are sets of abstract artifacts, abstract tasks, abstract business rules, organization roles, and role mapping function, respectively, and $pa_{\bar{\Pi}}: Z^p \cup V^p \cup R^p \cup L^p \cup \gamma^p \rightarrow Z \cup V \cup R \cup L \cup \gamma$ is an ACP abstraction function mapping from the public view of $\bar{\Pi}$ to $\bar{\Pi}$. $pa_{\bar{\Pi}}$ is a total mapping function such that the following conditions must be satisfied.

—$pa_{\bar{\Pi}}$ and $\bar{\Pi}$ have identical sets of organization roles, i.e., $L^p = L$,

—$pa_{\bar{\Pi}}$ contains each abstract shared artifact of its corresponding shared artifact in $\bar{\Pi}$, i.e.,

$$\forall C \in Z, \exists C_i \in Z^p, |\gamma(C)| > 1 \rightarrow (pa_{\bar{\Pi}}(C_i) \land |\gamma^p(C_i)| > 1),$$

—each abstract shared artifact in $pa_{\bar{\Pi}}$ has an identical role set to its corresponding concrete artifact in $\bar{\Pi}$, i.e.,

$$\forall C_i \in Z^p, \exists C \in Z, pa_{\bar{\Pi}}(C_i) = C \rightarrow \forall l \in \gamma^p(C_i), l \in \gamma(C),$$

—every rule and task defined in $\bar{\Pi}$ must be projected to some abstract rule and abstract task, respectively, defined in $pa_{\bar{\Pi}}$, i.e.,

$$\forall r_i \in R, \exists r \in R^p, pa_{\bar{\Pi}}(r) = r_i \land \forall v_i \in V, \exists v \in V^p, pa_{\bar{\Pi}}(v) = v_i,$$

—if an entire local artifact in $\bar{\Pi}$ is hidden in $pa_{\bar{\Pi}}$ then the following holds:

$$\forall C \in Z, \exists C_i \in Z^p \land |\gamma(C)| = 1 \rightarrow pa_{\bar{\Pi}}(C_i) = C,$$
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

 ROLE MAPPING $\gamma^P$ for abstract artifacts, tasks, and business rules holds:

$$\forall C_i \in Z^P, \gamma^P(C_i) \subseteq L \land \forall v_i \in V^P, \gamma^P(v_i) \in L \land \forall r_i \in R^P, \gamma^P(r_i) \in L.$$ 

Example 5.. Figure 5. illustrates an example of an agreed public view that can be constructed from the purchasing process introduced in Figure 5.. We can see that the public view is achieved by abstracting three shared artifacts $PO$, $SO$, and $IV$ and all local artifacts of each organization. It is possible that, due to aiming to achieve higher level of abstraction, abstract states may be introduced in the public view, e.g., approving, supplying, unpaid are abstract states in Figure 5..

Once a public view is constructed based on its underlying ACP-i model, it is very important to ensure the validity of the public view. As already mentioned, in this thesis we only focus on the behavior perspective of ACP models, so we define the validity of public view as the behavior-consistency between a view and its base model. In our recent work [Yongchareon et al., 2012], we have proposed a behavioral consistency checking approach called $B$-consistency to check whether a specialized process that derives the base process is consistently observable. Here, we use the $B$-consistency notion for checking whether the behavior of a public view is consistent with its underlying model.
5.3.2 Abstracting non-synchronized fragment

We reuse the notion of \textit{L-fragment} (defined in Definition 4.) for capturing the part of an artifact lifecycle that is to be abstracted in a public view. Once an L-fragment is identified in a lifecycle, we apply the abstraction function to map from \textit{L-fragment} to a specified abstract state or abstract transition(s) in an abstract ACP model (i.e., a public view). Based on this, given a complete \textit{ACP-i} model, we use two the abstraction methods discussed in Section 5.3.1 together with L-fragments to construct an abstract \textit{ACP-i} model which can be used to represent the public view of its base model. Next, we discuss the two output types of an abstraction: \textit{abstract transition} and \textit{abstract state}.

In most cases, an abstraction should yield an abstract transition as the transition represents an atomic and uninterruptable step from one state to another state in the lifecycle. However, in some cases, an abstract state is implemented. For instance, consider the \textit{PO} artifact in the concrete purchasing process in Figure 5. and compare it with its public view in Figure 5.. We can see that the approving state in the public view is an abstract state of the lifecycle fragment consisting of states \textit{created} and \textit{on Hold}. There are two possible underlying reasons supporting this case. First, an abstract state is specified in the design of an organization or in the mutual agreement of the collaboration. Second, a fragment cannot be abstracted into a single abstract transition - this is because the fragment is not atomic (i.e., \textit{NAL-fragment} defined in Definition 4.). The result of multiple abstract transitions for L-fragment incurs a difficulty to decide on drawing a projection from a part of the fragment to an abstract transition. Therefore, a possible solution is to use an abstract state to remove the ambiguity of what is abstracted in such transitions. Here, we show different cases of L-fragment abstraction in Figure 5.
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

Example 5.. Figure 5. (a) and Figure 5. (b) show the abstractions for abstract transition of L-fragments in artifact $A_1$ and artifact $A_2$, respectively, while Figure 5. (c) shows a case of abstraction for abstract state of artifact $A_2$. We can see that both L-fragments in $A_1$ and $A_2$ are AL-fragments. However, consider the L-fragment of artifact $A_3$ in Figure 5. (d) which is an NAL-fragment (due to multiple exit states $s_4$ and $s_5$). The abstraction of this fragment does not result in a single abstract transition. As already discussed, multiple abstract transitions are ambiguous. Therefore, to tackle this issue, the abstraction needs to result in an abstract state instead, which is shown in Figure 5. (e).

Next, we define lifecycle abstraction mapping function to map an abstract element in abstract lifecycle onto a state or transition in the base lifecycle, which is shown in Definition 5.. Then, in Definition 5., we define two abstraction functions that are used to construct an abstract lifecycle. These functions are based on the preferred output of the abstraction – that is abstract state or abstract transition.

Definition 5.: (Lifecycle abstraction (la)). Let artifact lifecycle $\mathcal{L}'_{cx} = (S', s'^{init}, \Rightarrow')$ in base ACP model $\Pi$ be an abstract lifecycle of artifact lifecycle $\mathcal{L}_{cx} = (S, s^{init}, \Rightarrow)$ in abstract ACP model $\Pi'$. We define lifecycle abstraction mapping function $la_{\Pi'\rightarrow\Pi}^{\mathcal{L}'}_{\mathcal{L}_{cx} \rightarrow \mathcal{L}'_{cx}} : S' \cup \Rightarrow' \rightarrow S \cup \Rightarrow$, where $la_{\Pi'\rightarrow\Pi}^{\mathcal{L}'}_{\mathcal{L}_{cx} \rightarrow \mathcal{L}'_{cx}}$ is a total function that maps
an abstract transition in $\Rightarrow'$ and an abstract state in $S'$ onto a state and a transition in $L_{c_x}$. We write $la$ without its superscription or its subscription in a clear context.

**Definition 5.** (Abstraction functions for abstract transition ($la_{tran}$) and for abstract state ($la_{state}$)). Let $L$-fragment $\ell_i^{c_x}$ of artifact lifecycle $L_i^{c_x} = (S, s_{init}, \Rightarrow)$ to be abstracted. We can abstract $\ell_i^{c_x}$ into a set of abstract transitions and an abstract state (if applicable) in abstract lifecycle $L'_i^{c_x} = (S', s_{init}, \Rightarrow')$ via the two following abstraction construction functions:

—function $la_{tran}(L_i^{c_x}, \ell_i^{c_x})$ returns $L'_i^{c_x}$ by abstracting $\ell_i^{c_x}$ into a single abstract transition,
—function $la_{state}(L_i^{c_x}, \ell_i^{c_x}, s')$ returns $L'_i^{c_x}$ by abstracting $\ell_i^{c_x}$ into single abstract state $s'$ and corresponding abstract transitions related to $s'$.

Functions $la_{tran}$ and $la_{state}$ can be expressed by lifecycle abstraction mapping $la_{c_x} : L_{c_x} \rightarrow L_{c_x}$ as follows.

(a) Let $\ell_i^{c_x}$ be an AL-fragment with entry state $s_i$ and exit state $s_j$. $\ell_i^{c_x}$ can be abstracted into abstract transition $s_i \Rightarrow s_j$ by applying function $la_{tran}$. We have $L'_i^{c_x} = la_{tran}(L_i^{c_x}, \ell_i^{c_x})$ such that,

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$la(s_i) = s_i \land la(s_j) = s_j$</td>
<td></td>
</tr>
<tr>
<td>$\forall s \in \ell_i^{c_x}, S, la(s \Rightarrow s_j) = s$</td>
<td></td>
</tr>
<tr>
<td>$\forall \Rightarrow \in \ell_i^{c_x}, \Rightarrow la(s_i \Rightarrow s_j) = \Rightarrow$</td>
<td></td>
</tr>
</tbody>
</table>

(b) Let $\ell_i^{c_x}$ be an AL-fragment with entry state $s_i$ and exit state $s_j$. $\ell_i^{c_x}$ can be abstracted into abstract state $s' \in S'$ with a set of abstract transitions $\Rightarrow_x'$ by applying function $la_{state}$. We have $L'_i^{c_x} = la_{state}(L_i^{c_x}, \ell_i^{c_x}, s')$ such that,

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall s \in \ell_i^{c_x}, S, la(s') = s$</td>
<td></td>
</tr>
<tr>
<td>$\forall \Rightarrow \in \ell_i^{c_x}, \Rightarrow \setminus (\ell_i^{c_x}, \Rightarrow_{in} \cup \ell_i^{c_x}, \Rightarrow_{out}), la(s') \Rightarrow$</td>
<td></td>
</tr>
<tr>
<td>$\forall \Rightarrow \in \ell_i^{c_x}, \Rightarrow_{in}, la(s \Rightarrow s') \Rightarrow$</td>
<td></td>
</tr>
<tr>
<td>$\forall \Rightarrow \in \ell_i^{c_x}, \Rightarrow_{out}, la(s \Rightarrow s') \Rightarrow$</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

\[\forall e \in \ell_Cx, \Rightarrow^{\text{out}}, la(s' \Rightarrow s_j) \Rightarrow e.\]

(c) Let \(\ell_Cx\) be an NAL-fragment with a set of entry states \(S^{en}\) and a set of exit states \(S^{ex}\). \(\ell_Cx\) can be abstracted into abstract state \(s' \in S'\) with a set of abstract transitions \(\Rightarrow_z\) by applying function \(la\_state\). We have \(L'_cx = la\_state(L_{cx}', \ell_{cx}', s')\) such that,

\[\forall s \in \ell_{cx}, la(s') = s,\]
\[\forall s' \in \ell_{cx}, \Rightarrow(\ell_{cx}, \Rightarrow^{\text{in}} \cup \ell_{cx}, \Rightarrow^{\text{out}}), la(s') = \Rightarrow_1,\]
\[\forall s_m \in S^{en}, \exists s_n \in \ell_{cx}, S, s_m \Rightarrow s' \in \Rightarrow_z \Rightarrow la(s_m \Rightarrow s') = s_m \Rightarrow s_n,\]
\[\forall s_0 \in S^{ex}, \exists s_p \in \ell_{cx}, S, s' \Rightarrow s_0 \in \Rightarrow_z \Rightarrow la(s' \Rightarrow s_0) = s_p \Rightarrow s_0.\]

**Example 5.** Figure 5. (a) and Figure 5. (b) show the abstract transitions resulted from applying function \(la\_tran\) to AL-fragments, while Figure 5. (c) shows the abstract state that generated from function \(la\_state\) on an AL-fragment. On the other hand, Figure 5. (e) shows the abstract state and related abstract transitions as an output of abstraction by applying function \(la\_state\) on a NAL-fragment.

Next, we define \(B\)-consistent abstraction based on the lifecycle abstraction mapping, and then we show, in **THEOREM 5.**, that applying either function \(la\_tran\) or function \(la\_state\) to generate an abstract lifecycle from its base lifecycle always preserves \(B\)-consistency.

**Definition 5.** (\(B\)-consistent abstraction, \(\equiv_{la}\)). Let artifact lifecycle \(L'_cx = (S', s'^{\text{init}}, \Rightarrow')\) in abstract ACP model \(\Pi'\) be an abstract lifecycle of artifact lifecycle \(L_{cx} = (S, s^{\text{init}}, \Rightarrow)\) in ACP model \(\Pi\) based on lifecycle abstraction mapping function \(la^{\Pi' \rightarrow \Pi} L'_cx \rightarrow L_{cx}\). If \(L'_cx \equiv L_{cx}\), then we say that \(L'_cx\) is a \(B\)-consistent abstraction of \(L_{cx}\), denoted as \(L'_cx \equiv_{la} L_{cx}\).

**THEOREM 5.** (Non-synchronized lifecycle \(B\)-consistent abstraction). Let lifecycle \(L'_cx\) be an abstract lifecycle of \(L_{cx}\) by abstracting L-fragment \(\ell_{cx}\) where \(\ell_{cx} < L_{cx}\). Given \(L'_cx = la\_tran(L_{cx}, \ell_{cx})\), we have \(L'_cx \equiv_{la} L_{cx}\). Correspondingly, \(L'_cx \equiv_{la} L_{cx}\) holds for \(L'_cx = la\_state(L_{cx}, \ell_{cx}, s')\) where \(s' \in L'_cx, S\).
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

The proof of **THEOREM 5.** can be found in the Appendix.

It is worth to note that, in **Definition 5. (c)**, we make use of abstract states to allow the abstraction of **NAL-fragments**; however, there requires rules with appropriate condition that must be specified for the abstract entry and exit transitions of an abstract state. For instance, consider the abstraction in Figure 5. (e). Entry transition \( s_1 \Rightarrow s_x \) abstracts two original entry transitions of its fragment \( (s_1 \Rightarrow s_2 \text{ and } s_1 \Rightarrow s_3) \) and exit transition \( s_x \Rightarrow s_4 \) abstracts two original exit transitions \( (s_2 \Rightarrow s_4 \text{ and } s_3 \Rightarrow s_4) \). Nevertheless, we can see that exit transition \( s_x \Rightarrow s_5 \) should abstract only for one exit transition \( s_3 \Rightarrow s_5 \). Alternatively, if we want not to use an abstract state as the result of abstraction of NAL-fragment due to the above addition mechanism to decide the appropriate trigger conditions for transitions, then we may expand the NAL-fragment until it satisfies the condition of AL-fragment (if possible). This requires the modeler to find a possible AL-fragment from the expanded-boundary of NAL-fragment. If such expanded fragment cannot be found, then they may decide to implement an abstract state for the abstraction. Consider two abstractions (A) and (B) in Figure 5., for instance. On the one hand, the abstraction (A) on **NAL-fragment** consisting of states \( s_2 \text{ and } s_3 \) can result in abstract state \( s_x \) and four abstract transitions. On the other hand, with alternative abstraction (B), the boundary of the fragment is expanded until it satisfies the condition of AL-fragment. We have the expanded fragment covers all states and transitions between states \( s_0 \text{ and } s_7 \). Then, we can abstract the fragment into either abstraction transition \( s_0 \Rightarrow s_7 \) or abstract state \( s_y \).

![Figure 5.: Abstraction on expansion of NAL-fragment](image-url)
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

Now, we present an algorithm to find the minimal AL-fragment of an input L-fragment. The algorithm gets an input fragment and expands its boundary until the expanded fragment satisfies the condition of AL-fragment (in Definition 4.). The output AL-fragment is minimal as in an iteration of the algorithm, it finds the nearest source and target states and adds them into the fragment. If the input already qualifies as AL-fragment then the fragment itself is returned. Otherwise, if the function cannot find a valid AL-fragment, then the null value is returned.

Algorithm (function find_minAL). Finding a minimal AL-fragment from an L-fragment

Input: L-fragment $\ell^{c_x}$ of artifact $C_x$ in ACP-i model $\Phi$.

Output: AL-fragment $\ell'_{c_x}$ if found; otherwise null is returned.

$\ell'^{c_x} = \ell^{c_x}$;

repeat

$S^{en}$ = a set of entry states of $\ell'^{c_x}$;

$S^{ex}$ = a set of exit states of $\ell'^{c_x}$;

if ($|S^{en}| > 1$ or $|S^{ex}| > 1$)

then

if $|S^{en}| > 1$ then

$\ell'^{c_x}$ includes all states in $S^{en}$ and their related exit transitions;

if $|S^{ex}| > 1$ then

$\ell'^{c_x}$ includes all states in $S^{ex}$ and their related entry transitions;

else if ($|S^{en}| = 1$ and $|S^{ex}| = 1$) then return $\ell'^{c_x}$;

until $S^{en} = \emptyset$ and $S^{ex} = \emptyset$

return null;

Based on Theorem 5. along with the help of the find_minAL function for NAL-fragment abstraction, we are able to construct a consistent public view of an individual artifact lifecycle by abstracting its non-synchronized part of the lifecycle. Next, we discuss how a fragment of a lifecycle that synchronizes with a fragment of another lifecycle can be consistently abstracted.

5.3.3 Abstracting synchronized fragments

Here, we define the abstraction relation of the synchronization between two abstract artifacts in the abstract ACP model and the synchronization between two artifacts in its base ACP model. The synchronization abstraction is defined based on lifecycle abstraction (Definition 5.) and sync rules (Definition 4.).
Definition 5.: (Synchronization (Sync) abstraction). Let artifact lifecycles $L'_cX$ and $L'_cY$ in ACP model $\Pi'$ abstract artifact lifecycles $L_cX$ and $L_cY$ in base ACP model $\Pi$ with lifecycle abstraction mappings $la_{L'_cX} \rightarrow L_cX$ and $la_{L'_cY} \rightarrow L_cY$, respectively. We can define sync abstraction mapping function $sa^{\Pi' \rightarrow \Pi} (L'_cX, L'_cY) \rightarrow (L_cX, L_cY) : \varphi(L'_cX, L'_cY) \rightarrow \varphi(L_cX, L_cY)$ that is used to project the abstract sync rule for $L'_cX$ and $L'_cY$ onto its base sync rule for $L_cX$ and $L_cY$.

Note that $sa^{\Pi' \rightarrow \Pi} (L'_cX, L'_cY) \rightarrow (L_cX, L_cY)$ is a total function and we may write $sa$ without its superscription and subscription in a clear context.

Now, we want to perform an abstraction on an L-fragment that synchronizes with other L-fragment(s). Similar to abstraction function for non-synchronized lifecycle, we define sync abstraction function for abstracting AS-region (defined in Definition 4.) that contains synchronized L-fragments.

Definition 5.: (Sync abstraction function (sa_f)). Given ACP model $\Pi$, let AS-region $\omega = (\Gamma, R^\text{sync})$ to be abstracted in abstract ACP model $\Pi'$, where $\Gamma = \{c^i, c^j, ..., c^x\}$ and $c^i (1 \leq i \leq x) \in \Gamma$ is an L-fragment of lifecycle $L_{c_i}$ in $\Pi$. We define sync abstraction function $sa_f (\Pi, \omega)$ to return returns $\Pi'$ with a set of abstract artifact lifecycles $L = \{L'_cX, L'_cY, L'_cZ, ..., L'_cX\}$ and an abstract sync rule $r'$, where $L'_cX (1 \leq i \leq x) \in L$ is a lifecycle of artifact $c_i \in \Pi'.Z$ such that,

— for every $c^i \in \Gamma$, $c^i$ is abstracted into abstract transitions in $L'_c$, i.e.,

$$L'_c_i = la_{\text{tran}}(L_{c_i}, c^i),$$

— for every sync rule $r \in \Pi.R$ that is used to synchronize between any two L-fragments in $\Gamma$, $r$ is abstracted into $r' \in \Pi'.R$, i.e.,

$$\forall c^x, c^y \in \Gamma, \forall r \in \varphi(c^x, c^y), \exists! r' \in \varphi(L'_cX, L'_cY), sa_{(L'_cX, L'_cY)}(r') = r.$$ 

Abstract sync rule $r'$ that is used to synchronize all abstract transitions together can be defined as follow. For every L-fragment $c^i = (S, \Rightarrow, \Rightarrow_{\text{in}}, \Rightarrow_{\text{out}}) \in \Gamma$, we have,
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

\[ \exists s_a, s_t \in \mathcal{L}_{\text{AS}} \setminus S, \ (s_2 = s_t^* \Rightarrow^{	ext{out}} s_1) \rightarrow s_2 \in \text{pre}_S(r', C_l) \land s_t \in \text{post}_S(r', C_l). \]

Example 5. In Figure 5. (a), we can see that AS-region \( \omega_a \) contains two fragments \( l_1 \) and \( l_2 \) with sync rules \( r_1 \) and \( r_2 \). When applying \( \text{sa}_f(\Pi, \omega_a) \), we have abstract lifecycle of \( \mathcal{L}_{A_1} \) and abstract lifecycle of \( \mathcal{L}_{A_2} \) with abstract sync rule \( r' \). In addition, Figure 5. (b) shows a case of sync abstraction for AS-region \( \omega_b \) which contains multiple entry transitions and multiple exit transitions ASL-fragments (both ASL-fragments \( l_3 \) and \( l_4 \)).

![Figure 5: Examples of sync abstraction](image)

In the above example, we demonstrate the sync abstraction of two synchronized fragments. However, it is possible that an AS-region contains more than two ASL-fragments. For wider understanding of sync abstraction, we illustrate a sync abstraction of more than two SL-fragments in Figure 5. (with artifact \( A_3 \) extended to the example in Figure 5. (b)).

Example 5. In Figure 5., we can see that AS-region \( \omega_b \) contains synchronized fragments \( l_3 \) and \( l_4 \) of artifact \( A_1 \) and \( A_2 \), respectively, and \( l_5 \) of artifact \( A_3 \). As all three fragments can be considered as ASL-fragments in \( \omega_b \), we can validly apply function \( \text{sa}_f(\Pi, \omega_b) \) and the abstract lifecycles of artifacts \( \{A'_1, A'_2, A'_3\} \) with abstract sync rules \( \{r', r''\} \) are returned.
Next, we discuss the case of an AS-region containing SL-fragments with a nested (sub) SL-fragment that synchronizes with other lifecycle. Intuitively, the sub SL-fragment and its synchronized lifecycle should be also taken into account when its super fragment has to be abstracted. Therefore, we need to induce the abstraction to its sub fragment together with its counterpart if they both can satisfy the property of AS-region (which is considered as a sub AS-region of the whole). In other words, we can say that the entire AS-region should contain such counterpart in order to have a valid abstraction. We show an example of AS-region consisting of sub SL-fragment in Figure 5..

Example 5. In Figure 5., we can see that L-fragment $l_4$ is a synchronized fragment of L-fragment $l_5$ which is nested under L-fragment $l_1$. The abstraction

Figure 5.: Example of abstraction of more than two ASL-fragments

Figure 5.: Example of abstraction on nested sub-SL-fragment
yields abstract transitions with abstract sync rule $r'_2$ that is $s_1 \xrightarrow{r'_2} s_2$ in the lifecycle of $A'_4$ and $s_1 \xrightarrow{r'_2} s_6$ in the lifecycle of $A'_1$.

Next, we show that sync abstraction function $sa_f$ preserves the $B$-consistency of between two synchronized lifecycles of the input and two outputted abstract lifecycles.

**THEOREM 5. (SYNCHRONIZED FRAGMENTS B-CONSISTENT ABSTRACTION).** Let $\omega = (\bar{\Gamma}, R^{Sync})$ be an AS-region in $ACP$ model $\Pi$ that is to be abstracted in $ACP$ model $\Pi'$ and let $L$ be a set of abstract artifact lifecycles resulted from applying sync abstraction function $sa_f(\Pi, \omega)$. Then the following statement holds.

$$\forall \ell \in L \mid \ell \notin L \land \ell \in \bar{\Gamma}, \exists \ell' \_c_i \_c_j \in L, L' _c_i \_c_j \_c_i \_c_j \rightarrow L' _c_i \_c_j \_c_i \_c_j \rightleftharpoons L'. $$

The proof of THEOREM 5. can be found in the Appendix.

### 5.3.4 B-Consistent public view construction

Now, we use THEOREM 5. and THEOREM 5. to formulate the $B$-consistency for the entire artifact-centric inter-organizational business process.

**THEOREM 5. (B-CONSISTENT PUBLIC VIEW).** Let $pa_{\bar{\Pi}}$ be a public view of $ACP$-i model $\bar{\Pi}$ constructed by applying sync abstraction function $sa_f$ and lifecycle abstraction functions $la_{tran}$ and $la_{state}$. Then $pa_{\bar{\Pi}}$ is a B-consistent abstraction of $\bar{\Pi}$, i.e., $L_{pa_{\bar{\Pi}}} \rightleftharpoons_{la} L_{\bar{\Pi}}$.

THEOREM 5. is derived from THEOREM 5. and THEOREM 5..

Recall that in the public view, all local artifacts should be invisible. Only abstract shared artifacts are revealed for the collaboration. In order to hide those local artifacts of each party, the party must ensure that if a local artifact synchronizes with a fragment of a shared artifact, then this fragment must be hidden as well. This hiding property refers to the abstraction of such fragment.
However, if the entire lifecycle of a local artifact is abstracted, then it can be validly hidden in the public view. We use the definition of *ex-lifecycle* (in *Definition 4.*) to capture this kind of lifecycle. In other words, if the lifecycle of a local artifact is a fully-embedded external lifecycle of the shared artifact, then the local artifact can be hidden. As such, the corresponding fragment(s) in the shared artifact(s) is also abstracted (by using sync abstraction function $sa_f$). However, there is a case if that fragment is not an AL-fragment, then the $sa_f$ function cannot be applied due to the requirement of the input that must be AS-region (consisting of ASL-fragments of shared artifact(s) and local artifact). To cope with this issue, we propose to use lifecycle abstraction function $la_state$ to abstract it into an abstract state. Then a local artifact that synchronizes with that fragment can be abstracted. Although we can abstract an NAL-fragment into an abstract state, it is hard to decide whether that abstraction is valid and consistent in terms of synchronization behavior. A local artifact that synchronizes with any part of such NAL-fragment must be exclusively encapsulated in the abstract state. In addition, the abstract state itself is not considered as atomic if it has multiple entry and multiple exit transitions from/to different states. Therefore, we allow having sync abstraction for an abstract state for the case that the whole lifecycle of a local artifact is synchronized within the NAL-fragment. As previously discussed, representing an NAL-fragment in an abstract transition is deemed inconsistent.

*Example 5.* Revisit our purchasing process example in Figure 5.. We can construct a fragment for the $PO$ artifact that consists of states \{created, on hold\}. The fragment is considered as NAL-fragment (due to two exit states). We can see that it synchronizes with the entire lifecycle of the $Quote$ artifact, therefore, the abstraction of this fragment must yield an abstract state which requires the whole lifecycle of $Quote$ to be abstracted, i.e., fully-embedded – that is the approving state of abstract $PO$ in the public view shown in Figure 5.. Similarly, the entire lifecycle of $PL$ artifact can be abstracted and fully-embedded in the supplying state of the abstract $PO$ in the public view.
5.3.5 Finding minimal B-consistent public view

In this section, we propose an algorithm (for a function named find_minPV) to help organizations to automatically find the minimal, consistent public view from their ACP-i model. To achieve the minimal public view construction, the find_minPV function requires two additional algorithms: find_minASR and find_SR.

Given an L-fragment of one artifact lifecycle that synchronizes with other artifact lifecycle(s), how can we find the minimal AS-region (which contains that L-fragment) and its minimal counterpart(s) in which they can be used as inputs for sync abstraction function sa_f. Here, we propose two algorithms to find the minimal AS-region of an L-fragment. Note that the synchronized counterparts can be many as the L-fragment can synchronize with multiple lifecycles. Two proposed algorithms find_minASR and find_SR are used to expand the boundary of L-fragment along with its synchronized counterpart(s) until all satisfy the condition of AS-region and ASL-fragment (in Definition 4.). If no AS-region can be constructed or an L-fragment has no synchronized counterpart of any other lifecycle, then the null value is returned.

Algorithm (function find_minASR). Finding a minimal AS-region from an L-fragment

Input: L-fragment ℓCR in ACP model Π.
Output: AS-region ω = (Γ, Rsync) if found; otherwise null is returned.

ω ← (ℓCR, ∅);
ω = find_SR(ω);

return ω;

Algorithm (function find_SR). Finding an expanded S-region from an input S-region

Input: S-region ω = (Γ, Rsync) in ACP model Π.
Output: Expanded S-region of ω or null if any SL-fragment in ω cannot satisfy the property of AL-fragment.

for each ℓ ∈ ω.Γ do
    L-fragment ℓ′ = find_minAL(ℓ);
    if (ℓ′ = null) then return null;
    else
        ω.Γ = ω.Γ ∪ {ℓ′} \ {ℓ};
    end if
end for
Here, we use a running example illustrated in Figure 5. to explain the algorithms of functions \( \text{find\_min\_ASR} \) and \( \text{find\_SR} \).
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

We begin with Figure 5. (a) by applying $\text{find\_minASR}(l_1)$ in this example. First it takes L-fragment $l_1$ of artifact $A_2$ as an input and initializes an S-region from $l_1$ and an empty set of sync rule. Then it finds the possible corresponding S-region for $l_1$ (by calling $\text{find\_SR}$ function). Function $\text{find\_SR}$ starts checking whether $l_1$ can satisfy the property of AL-fragment. If so, then it continues searching for all sync rules and their related synchronized states that are used to synchronize $l_1$ with any other lifecycle transitions; otherwise, it immediately returns null. In the iteration of
finding synchronized part of other artifact, if a set of synchronized states is found, then a fragment consisting of such set is constructed (by calling find_lf function defined in Definition 4.) – that is L-fragment \( l_2 \) of artifact \( A_1 \) shown in Figure 5. (b). Next, \( l_2 \) is to be checked whether it is qualified as AL-fragment. If it satisfies, then it is added into the S-region; otherwise, it has to be expanded until it can satisfy AL-fragment. Once qualified, it is added into the S-region. After \( l_2 \) is added to the S-region, we recursively call function find_SR again with the expanded S-region as an input. Figure 5. (c) shows AL-fragment \( l'_2 \) that is expanded from \( l_2 \). We can see that \( l'_2 \) introduces new sync rules \( \{r_5, r_6\} \) that are not included in the S-region. The recursion will continue until all the SL-fragments in the S-region are AL-fragments and include all the possible sync rules that are used in the S-region. Consequently, \( l_1 \) is required to expand itself and becomes new AL-fragment \( l'_1 \), as shown in Figure 5. (d). After that, we can see new sync rules \( \{r_7, r_8\} \) appear in \( l'_1 \). The S-region needs another expansion again to cover those sync rules – that is L-fragment \( l_3 \) of artifact \( A_3 \). Finally, by completing the iteration and recursion with all the conditions of AS-region satisfied, the function returns the minimal AS-region consisting of ASL-fragments \( \{l'_1, l'_2, l_3\} \) as the output from the provided input L-fragment \( l_1 \). If the AS-region cannot be constructed then the function returns false. We can say the resulted AS-region is minimal as the function use the find_minAL function to guarantee the minimal AL-fragment expansion in the AS-region.

Based on Theorem 5. and the use of the find_minSR and find_SR functions, we can construct a consistent public view of synchronized lifecycles by abstracting their synchronized parts of the lifecycles.

Next, we show an algorithm for the find_minPV function. Due to the inconsistency issue on NAL-fragment abstraction, we do not take NAL-fragments into account in this algorithm. After presenting the algorithm, we then show how it can guarantee the B-consistency.

**Algorithm (function find_minPV).** Finding the minimal, consistent public view of ACP-i model

| Input: ACP-i model \( \Phi = (Z,V,R,L,Y) \). |
| Output: the minimal public view of \( \Phi \). |
| public ACP-i model \( \Phi' = \Phi \); |
| for each \( C_i \in \{C \in \Phi'.Z \mid |Y(C)| = 1\} \) do |
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

L-fragment $\ell_i = \text{find}_Lf(C_i, C_i, S)$;

AS-region $\omega = \text{find}_\text{minASR}(\ell_i)$;

if ($\omega \neq \text{null}$)

then

$\mathcal{F}' = \text{sa}_f(\mathcal{F}', \omega)$;
artifacts $Z^i \leftarrow C_i$;

for each $C_j \in \{ C \in \mathcal{F}' : |\mathcal{Y}(C)| = 1 \}$ do

L-fragment $\ell_j = \text{find}_Lf(C_j, C_j, S)$;

if ($\ell_j = \text{null}$) then $Z^i \leftarrow C_j$;

end for

remove all artifacts in $Z^i$ and their related abstract sync rules from $\mathcal{F}'$;

$Z^i = \emptyset$;

end if

end for

return $\mathcal{F}'$;


Now, we explain the algorithm for the $\text{find}_\text{minPV}$ function. First, it initializes a public view as identical to an input $ACP-i$ model. Then, it searches for all local artifacts in the public view. For each local artifact found, it attempts to construct an AS-region that consists of such artifact. If it is able to construct the AS-region, then abstraction function $\text{sa}_f$ is applied on the region and the artifact is added to the set $Z^i$ and all artifacts in $Z^i$ will be later removed from the public view. As $\text{sa}_f$ finds all corresponding ASL-fragments (if constructible) that synchronize with the local artifact into account for the abstraction, the algorithm also searches for other local artifact that qualifies as fully-embedded external lifecycle to be included in $Z^i$. This will eliminate an unnecessary local artifact in the public view; therefore, the number of iterations for finding local artifacts is reduced in the main loop.

From the algorithm we will get the minimal, $B$-consistent public view of the $ACP$-i model. However, it does not guarantee that all local artifacts are abstracted. This depends on whether the lifecycle of such artifacts can be fully-embedded as well as the synchronized part of the shared artifact is an ASL-fragment. In other words, if a fragment is an NAL-fragment, then the abstraction of those local artifacts must be achieved manually – by abstracting each of such fragments into an abstract state instead. This process requires a predefined abstract state for an abstract shared artifact.
Next, we show in THEOREM 5. that our $find_{minPV}$ function produces the minimal, $B$-consistent public view of a given $ACP$-$i$ model.

THEOREM 5.. Given $ACP$-$i$ model $\tilde{\Pi}$, $find_{minPV}(\tilde{\Pi})$ returns a minimal, $B$-consistent public view of $\tilde{\Pi}$.

The proof of THEOREM 5. can be found in the Appendix.

We now conclude Section 5.3. In summary, this section has discussed the public view construction methodology and behavioral consistency and formulates several functions and theorems that can be used to construct the $B$-consistent public view of the collaboration. It also presented our algorithms that help organizations automatically generate the minimal, abstract public view with the assurance of the $B$-consistency.

5.4 Process Changes and Change Validation

In this section, we provide a discussion about a mechanism to allow a party in the collaboration to change their local (private) process without affecting the correctness and consistency of the overall process.

5.4.1 Process changes

Organizations may need to change their local process due to their new (or updated) set of business requirements or regulations that they have to follow. We observe that in artifact-centric processes, changes of local process can be classified into three main points of view based on the three components of $ACP$ model.

—Changes to artifacts. An organization may modify/delete/add an attribute or a state of its local artifact, which can be seen as structural changes to the data model of artifact. In addition, not only the changes to existing artifact, it is possible that the organization can incorporate a set of new artifacts in the local process.
—Changes to tasks. Stemmed from service-oriented architecture, an organization may seek to aggregate existing local tasks into a composite task (i.e., a service). On the other hand, a composite task can be decomposed into smaller tasks. Due to these possible changes of task, the specification of the task is subject to reflecting the changes – that is an input, output, and pre/post-conditions of the task.

—Changes to business rules. Business rules can be changed due to the structural changes of artifacts, the specification changes of tasks, and the changes of process’s logic. It is worth noting that changes may impact to an existing interaction behavior between artifacts.

Next, we define three change operators that can be used to express what and how changes to the above components in the local process can be achieved. As aforementioned, in this thesis we only focus the behavioral aspect of artifact-centric business processes. We restrict our discussion on observing process changes through the change of the behavior of local process. Therefore, for simplicity, we assume that capturing the change of a business rule can reflect the change of artifact(s) and task(s) involved in such rule. This makes sense as the condition of artifacts and tasks are defined in the pre/post-condition and action, respectively, in business rules. In the case of adding new artifact to a local process, we can derive the lifecycle of new artifact and its interaction from the added business rules that are used to induce the state transition of the artifact and the sync rules that are used to synchronize other existing artifact(s) with it, respectively.

Definition 5. (Change operators). Let \( \Pi^l = (Z^l, V^l, R^l) \) be a local ACP model of organization role \( l \) in the collaboration. An organization can change its own local process by applying the following change operators over business rules \( R^l \).

—Add operator \( (R^+) \). We write \( R^+(r) \) to mean that new business rule \( r \) is added into \( R \), i.e., \( R^+(r) = R^l \cup \{r\} \).

—Delete operator \( (R^-) \). We write \( R^-(r) \) to mean that existing business rule \( r \) is deleted from \( R \), i.e., \( R^-(r) = R^l \setminus \{r\} \).
Chapter 5. Facilitating Inter-Organizational Business Process Collaboration

—Replace operator ($R%$). We write $R% (r_1, r_2)$ to mean that existing business rule $r_1$ is replaced by new business rule $r_2$ in $R$, i.e., $R% (r_1, r_2) = R^l \cup \{r_2\} \setminus \{r_1\}$.

It is noted that the replace operator identically performs as the combination of the add operator and the delete operator; however, it provides better traceability of changes by maintaining the relation for the replacement of old business rule and new business rule.

**Definition 5.** (Modified local ACP model, process change function ($pc$)). Let $\Pi^l = (Z^l, V^l, R^l)$ be a local ACP model of role $l$. We can obtain modified local ACP model $\Pi'^l$ from $\Pi^l$ by applying process change function $pc_{\Pi^l}$: $\Pi^l \times X \rightarrow \Pi'^l$ where $X$ is a union set of change operations $R^+$, $R^\times$, and $R^\%$ that perform on some business rules in $R^l$.

From Chapter 4, the process specialization methods: extension and refinement of artifacts can be incorporated to the process change operations based on the similar idea from workflow inheritance presented in [van der Aalst et al., 2002, 2003]. Next, we explain the concept of private view and how it can be used to validate the changes of local process.

### 5.4.2 Private views and process conformance

In the overall picture, we validate the changes of local process by checking whether the $B$-consistency of modified inter-organizational business process (after local process changes applied) and the agreed public view can be preserved. A private view of an organization is used to capture the modified local process for the purpose of locally checking whether the local process is still able to provide what promised in the agreed contract, i.e., public view. We illustrate an overview of process change validation in Figure 5.
Next, we define the *private view* of a particular organization role in the collaboration based on the agreed public view. Based on each version of local process changes, each role has a corresponding private view that captures its (modified) *local ACP model* plus abstract shared artifacts defined in the public view.

**Definition 5. (Private view).** Let \( \pi = (Z^P, V^P, R^P, L^P, \gamma^P) \) be a public view of ACP-i model \( \pi \) and \( \pi^i = (Z^i, V^i, R^i) \) be a local ACP model of role \( l \in L^P \). The *private view* of role \( l \) can be defined by *private view mapping* function \( pv_{\pi^i} : Z^X \cup V^X \cup R^X \cup \gamma^X \rightarrow (Z^P \cup Z^i) \cup V^i \cup (R^P \cup R^i) \cup L^P \cup \gamma^P \) such that the followings hold.

—each abstract shared artifact in \( Z^P \) exists in \( Z^X \), i.e.,

\[
\forall C_i \in \{ C \in Z^P \mid |\gamma^P(C_i)| > 1 \}, \exists C_j \in Z^X, pv_{\pi^i}(C_j) = C_i \land \gamma^X(C_j) = \gamma^P(C_i),
\]

—each local artifact in \( Z^i \) (not in \( Z^P \)) exists in \( Z^X \), i.e.,

\[
\forall C_i \in \{ C \in Z^i \mid |\gamma^P(C_i)| = 0 \}, \exists C_j \in Z^X, pv_{\pi^i}(C_j) = C_i \land \gamma^X(C_j) = l,
\]

—each task in \( V^i \) exists in \( V^X \), i.e.,
∀lv_i \in V^l, \exists v_j \in V^x, pv_{HI}(v_j) = v_i \land y^x(v_j) = l,

— each business rule in \( R^i \) exists in \( R^x \), i.e.,

\[ \forall r_i \in R^i, \exists r_j \in R^x, pv_{HI}(r_j) = r_i, \]

— each abstract business rule in \( R^p \) that is not used for the synchronization between an abstract shared artifact in \( Z^p \) and a local artifact in \( Z^l \) exists in \( R^x \), i.e.,

\[ \forall r_i \in \{ r \in R^p | l \notin y^p(r) \}, \exists r_j \in R^x, pv_{HI}(r_j) = r_i, \]

Note that we may use the term private view in the meaning of the lifecycle of private view in a clear context.

Example 5. Figure 5. depicts the lifecycles of artifacts in the private view of the original local process of Supplier (\( \text{fl}_1^{\text{Supplier}} \)) which is extracted from the complete purchasing process shown in Figure 5.. Such private view can be constructed based on the public view shown in Figure 5.. Apart from the local artifacts \( PL \) and \( DN \), compared to the public view, we can see local process details (in gray-shaded areas) of shared artifacts \( PO \), \( SO \), and \( IV \).

Figure 5.: Supplier’s private view \( \text{fl}_1^{\text{Supplier}} \)
Example 5. Now, consider the case that if Supplier wants to change its local process based on the existing one (cf. Figure 5.). The result of changes is illustrated in private view $\phi^{\text{Supplier}}_2$ in Figure 5.. Since the $IL$ artifact is added into the local process, we can see some synchronization between existing artifact $PL$ and new artifact $IL$, as well as the change of state’s names of $PL$ (from checking to scheduled, from out of stock to unavailable, and from in stock to picking). Existing business rules that correspond to these changes are affected and needed to be adjusted accordingly. Obviously, a new set of business rules is needed to express the lifecycle of the added artifact. This set includes sync rules that are used for the synchronization between $PL$ and $IL$.

![Diagram of process flows](image)

Figure 5.: Supplier’s modified private view $\phi^{\text{Supplier}}_2$

Next, we define process conformance and its conditions that can be used to check whether changes in local process can be implemented while preserving the correctness and consistency of the overall process. Basically, we reuse the definition of $B$-consistency to help us to define process conformance for consistent local process changes as to satisfy the following statements.
—Changes should not lead to an unsound global process, i.e., the modified global process should be able to reach its goal states as its original global process does, and,

—Changes should be guaranteed that the $B$-consistency of the modified global process and its original global process is preserved.

As an agreed public view is constructed for the collaboration which acts like a contract, so we can express consistent process changes by means of not breaking the original public view. In other words, the modified $ACP$-$i$ model must be consistent with the agreed public view. Regarding the behavioral equivalence notion in process algebras [Bloom, 1995], we can say that our approach for consistent process changes preserves the congruence property of the modified local process and its base local process, i.e., they can behave interchangeably without affecting the overall process.

**Definition 5.** (Process conformance, $\models$). Let $\hat{\Pi} = (Z, V, R, L, \gamma)$ be an $ACP$-$i$ model and $pa_\Pi$ be its public view. Let $pv_{\hat{\Pi}}^l$ be a private view of local $ACP$ model $\hat{\Pi}^l$ for role $l \in L$. We say that $\hat{\Pi}^l$ conforms $pa_\Pi$, written as $\hat{\Pi}^l \models pa_\Pi$, iff the lifecycle of $pv_{\hat{\Pi}}^l$ is $B$-consistent with the lifecycle of $pa_\Pi$, i.e.,

$$\hat{\Pi}^l \models pa_\Pi \iff \mathcal{L}_{pv_{\hat{\Pi}}^l} \simeq \mathcal{L}_{pa_\Pi}$$

**Definition 5.** (Consistent process changes, congruent local processes). Given $ACP$-$i$ model $\hat{\Pi}$ and its public view $pa_\Pi$, let $\hat{\Pi}^l$ be a local $ACP$ model for role $l \in L$ and $\hat{\Pi}^l'$ be a modified local $ACP$ model of $\hat{\Pi}^l$ with process change $pc_{\hat{\Pi}^l}$. We say that $pc_{\hat{\Pi}^l}$ is consistent if $\hat{\Pi}^l \models pa_\Pi \Rightarrow \hat{\Pi}^l' \models pa_\Pi$. Correspondingly, we have $\hat{\Pi}^l' \simeq \hat{\Pi}^l$ and $\simeq$ is a congruence for $\hat{\Pi}$.

**Theorem 5.** (Process changes $B$-consistency). Let $pa_\Pi$ be a public view of $ACP$-$i$ model $\hat{\Pi}$, and let $\hat{\Pi}'$ be a modified $ACP$-$i$ model of $\hat{\Pi}$ based on a set of process changes $PC_{\hat{\Pi}^l} = \{pc_{\hat{\Pi}^l_1}, pc_{\hat{\Pi}^l_2}, ..., pc_{\hat{\Pi}^l_i}\}$ where $pc_{\hat{\Pi}^l_i}$ is a process change applied on local $ACP$ model $\hat{\Pi}^l_i$. Then, we have the lifecycle of $\hat{\Pi}'$ $B$-consistent with the lifecycle of $pa_\Pi$ if all process changes in $PC_{\hat{\Pi}^l}$ are consistent, i.e.,
∀pc_{n_i} ∈ PC_{n_i}, pc_{n_i} is consistent → L_{n'} ≃ L_{pa_{n'}}

The proof of THEOREM 5. can be found in the Appendix.

From THEOREM 5., we can see that if all organizations in the collaboration can guarantee that changes in their local process are consistent, then such changes do not violate the \textit{B-consistency} of the global process. It is worthwhile mentioning that checking local processes via private views implies that a global validation of entire process is not required, thus, avoiding the state space exposition problem. The benefit of private views and the local validation mechanism can be seen as a key driver for organizations to meet the three major requirements of efficient collaboration discussed in Section 5.1.2.

5.5 Open Issues

This section lists and discusses some important open issues that should be further investigated.

5.5.1 Changes in local processes and process conformance

In our notion of private views, changes are captured and validated based on the use of process conformance and the \textit{B-consistency} notion so as to guarantee the conformance to the public view and to restrict the behavior of a post-modified local process equivalent to the behavior of its original process. It is more desirable if we can directly (and possibly incrementally) validate change operations on local process without deriving all artifact lifecycles from added/modified business rules. An efficient and automatic checking algorithm should be developed for such validation. Based on existing works, Operating guidelines [Massuthe and Schmidt, 2005; Lohmann et el., 2007], correctness-preserving process configuration [van der Aalst et al., 2012], and compliance rules [Lohmann, 2011] can be adopted to address such requirement and complement our work. Moreover, process changes should be generalized and classified into patterns that address different needs; thus, these patterns can be used to facilitate defining changes specification.
5.5.2 Workflow realization

As our work mainly focuses on view-based methodology that can be applied towards the support of inter-organizational business process modeling and change validation in an artifact-centric paradigm, a realization mechanism for our approach in the current technologies and architectures requires further study.

A realization for our $ACP-i$ models can be possibly achieved by considering two different approaches featured with public view definition and private view validation mechanism. The simple solution is to transform from ACP models to process-centric models (e.g., in [Liu et al., 2010; Li and Wu, 2011]) with the use of model-transformation techniques presented in [Küster et al., 2007; Redding et al, 2008], and then realize them on existing workflow systems plus view features (e.g., in [Jiang et al, 2010]). However, this solution seems not very promising as the transformation poses several drawbacks (as discussed in [Ngamakeur et al., 2012]).

Cohn et al. [2008] presented Seina prototype that allow users to model business artifacts and process as an XML documents. The work on Active XML (AXML) [Abiteboul et al., 2008, 2009; Marinoiu et al., 2010], which is an extension of XML with embedded service calls, can be used for supporting the implementation of artifacts that are accessed within or among organizations in the collaboration. Influenced by these work, we aim to develop the pure artifact-centric process system for realizing our artifact-centric view framework. Currently, we are developing a prototype based on the realization framework in [Ngamakeur et al., 2012] with the support of shared/local XML-based artifacts.

5.6 Summary and Discussion

This chapter presented a view framework for modeling and change validation of inter-organizational business processes. We formally defined the artifact-centric inter-organizational business process model, namely $ACP-i$ model, with the notions of public and private views to address the major requirements for efficient collaboration: compliance, flexibility, and autonomy.
In our model, we distinguished two different types of artifacts, local artifacts and shared artifacts, to enable the views of private and public processes. Lifecycle abstraction methods to abstract artifacts and their synchronization in the public view were introduced along with the notions of \textit{B-consistency} and \textit{AS-region} for the behavioral consistency checking between the concrete processes and their derived (public/private) views. Based on these two notions, we develop algorithms to automatically find the minimal, consistent public view for a given inter-organizational business process.

In a private view, we allow organizations to extend their local processes by adding artifact into the processes. The extension is guaranteed to conform to the global process based on the \textit{B-consistency} and \textit{AS-region} notions as well. We illustrate that any change occurred (via change operations) in a local process can be validated via the use of process conformance such that the change will not affect the correctness and behavioral consistency of a global process. We used a conventional state composition technique to check the soundness of processes. However, stemming from the concern of organization’s privacy and autonomy, we proposed a validation technique that performs local composition (on private views) instead of total composition on the whole collaboration process; therefore, the problem of state exposition caused by performing global verification is avoided.
Chapter 6. Deriving Process-Oriented User Interfaces

As the media bridging systems and users, User Interfaces (UIs) show users particular data of certain artifacts and also enable users to input/edit the data of artifacts and invoke related functions. Driven by underlying business processes, a user may go through a sequence of UIs which help the users fulfill a certain procedure of the business process. From the declarative manner in describing artifact-centric business processes based on defining business rules, we can see that data required to perform tasks is explicit while control flows between tasks are implicit. This brings in the challenge to develop a mechanism to discover the flows of UIs and the data information of such UIs from an artifact-centric business process model. As such, we discuss on how UIs can be developed to support user-interaction as well as to serve as means for visualizing artifact-centric business processes. We propose a model-driven UIs derivation to achieve automatic
generation of UIs by performing model transformation from the artifact-centric business process model (ACP-i model) to our User Interface Flow model (UIF model) with a role-based view support. With the role-based feature, the UIF model can be used to visualize a process that is owned by a particular role of organization, or to serve as a model template to generate actual user interfaces that interact with users of the process.

We begin this chapter by addressing key problems and introducing the framework for automatic user interface derivation in Section 6.1. Then, Section 6.2 describes our UIF model which is the core component of the framework. Section 6.3 explains how the UIF model can be derived from the underlying artifact-centric business process model. Section 6.4 discusses a system architecture that can be implemented to support the framework. Lastly, the discussion and summary of this chapter is presented in Section 6.5.

6.1 Problems and UI Derivation Framework

6.1.1 Key problems

The first issue of the UIs’ generation in artifact-centric paradigm is to identify the navigation flow among UIs with the information that each UI must contain. We can view a single UI as a webpage that contains one or several forms with corresponding input fields required from users. At a particular stage of a business process, a form, its inputs, and the next step after the completion of the form, should be defined corresponding to the particular requirement of the process. For example, consider our inter-organizational purchasing process illustrated in Figure 5.. A (web) form should be appeared to Buyer when the PO artifact is in its on_hold state. This form should allow Buyer to confirm or cancel the PO. However, the PO can be confirmed only after the Quote is in its approved state. We can see that the processing state and data dependency of artifacts must be taken into account for the correct generation of UIs.

The second issue is how to decide a succeeding UI once a form in a current UI is submitted. There may be multiple following UIs for users to follow depending on
the current step and possible next step(s) allowed for them. Especially for collaborative business processes, the flow of UIs can be distributed across organizational boundary. Revisiting Figure 5. for example, once the PO is confirmed, then the Buyer should be waiting until the Supplier and Logistics finish their delivery process. After the Buyer receives an invoice sent from the Supplier (i.e., the Invoice is in its sent state and the Payment of Buyer is in its init state), then a corresponding UI (for an accounting department) must be activated so that the Buyer can make a payment to the Supplier.

The above two issues need to be addressed to support user interfaces for different roles of users that are involved in the inter-organizational business process.

6.1.2 A UIs derivation framework

Here, we consider two aspects of UIs: behavior and information. The behavioral aspect presents the flows between UIs, while the informational aspect presents the required data that users must complete for each UI so as to proceed to its following UI. In addition, UIs are defined for users and users in the organization have different roles. The authority to perform an activity in the process is restricted by particular roles of users. Different roles may have their own views of UIs. For this reason, we consider the role-based characteristic of UIs. Based on these requirements and the use of Model-Driven Architecture (MDA) approach, we propose the automatic UI derivation framework for artifact-centric business processes. The framework consists of three layers: ACP-i model, UIF-base model, and UIF-role model as illustrated in Figure 6.
The **UIF-base model** layer contains conceptual UIs and the dependencies among them. UIs are constructed based on behavior and information of business processes in the **ACP-i model**. Each individual UI in the model contains a set of states and attributes of related artifacts defined and used within it. The flow from one UI to another UI may be restricted by the behavior of correlated artifacts used in both UIs. The model is conceptual therefore it does not provide any physical layout or structure of the user-interfaces. This layer logically presents user-interfaces and their relations derived from the **ACP-i model**. The **UIF-role model** layer represents the role-based user-interface model. This model can be seen as a user view of UIF-base model defining which user-interfaces are enabled for particular roles which are defined in the **ACP-i model**. These UIF models can be used to guide UI designers to design and customize physical user-interfaces. The arrows in Figure 6. show the direction of transformations or derivations from a lower layer to an upper layer.

**Figure 6.: Automatic UI Derivation Framework for ACP-i model**

The **UIF-base model** layer contains conceptual UIs and the dependencies among them. UIs are constructed based on behavior and information of business processes in the **ACP-i model**. Each individual UI in the model contains a set of states and attributes of related artifacts defined and used within it. The flow from one UI to another UI may be restricted by the behavior of correlated artifacts used in both UIs. The model is conceptual therefore it does not provide any physical layout or structure of the user-interfaces. This layer logically presents user-interfaces and their relations derived from the **ACP-i model**. The **UIF-role model** layer represents the role-based user-interface model. This model can be seen as a user view of UIF-base model defining which user-interfaces are enabled for particular roles which are defined in the **ACP-i model**. These UIF models can be used to guide UI designers to design and customize physical user-interfaces. The arrows in Figure 6. show the direction of transformations or derivations from a lower layer to an upper layer.
6.2 User Interface Flow (UIF) Model

In this section, we formally describe the terminology and constructs in User Interface Flow Model (UIF model), and propose an approach to automatically derive UIF models from underlying artifact-centric business process models. The model comprises (1) a set of web pages and (2) relations between these pages. A page may contain a single or multiple input forms. Each form contains input fields that user must fill in data to make a form completed. The model represents a conceptual, but operational, level of user interfaces; thus it will not describe physical components and presentation structure of UIs. However, the concrete UIs can be generated by using Extensible Stylesheet Language Transformation (XSLT) [W3C, 2010] to transform from the UIF model file with predefined XSL templates. Here, we consider two abstract aspects of the UIF models: behavioural aspect (navigational control flow relations between UIs) and informational aspect (related/required data for each UI).

Figure 6.: A UIF Model and UIC with its artifacts

Figure 6. shows the components and structure of the UIF model. The round-rectangle depicts a User Interface Container (UIC), which represents a single web page of UIs. An Interface represents a form comprising a set of required attributes of corresponding artifact that is used in the form. A single UIC may contain either empty interface (for the final or initial UIC), or a single or multiple interfaces (for normal UIC). The Navigational Control Flow (NCF) is used to indicate that once
the interface with all required data has been submitted, then the action, e.g., tasks, corresponding to such interface is performed and the following UIC then becomes active. The UIF starts at the initial UIC and terminates when it reaches the final UIC.

6.2.1 UIF model syntax

Here we define UIF model and Interface that is contained in the UIF model.

Definition 6.: (Interface). An interface represents a form in a web page. It contains a required set of attributes of artifact, as well as a role of users and a corresponding task that will be invoked if users complete the form. Given ACP-i model \( \Phi = (Z, V, R, L, \gamma) \), we denote \( b \) for an interface that derives from \( \Phi \), and it is defined as tuple \( b = (O, \partial, \Delta, v, l) \) where,

\( \partial \subseteq Z \) is a finite set of artifact classes used in the interface,

\( \partial \subseteq \bigcup_i O_i.A (O_i \in O) \) is a required attribute set, which can be inputted or edited by users,

\( \Delta \subseteq \bigcup_i O_i.S (O_i \in O) \) defines a set of current states of each artifact of class in \( O \) when they are in the interface, i.e.,

\[ \forall s \in \Delta \exists O_i \in O, \exists r \in R, s \in O_i.S \rightarrow s \in post_s(r, O_i), \]

\( v \in V \) is a corresponding task which can be performed by users after attributes in \( \partial \) are all completed,

\( l \in L \) is a role of users (of organization role \( l \)). Only users with role \( l \) are permitted to see/access the interface.

We denote \( S_{O_i}^j \) to mean the \( S_j \) state of artifact of class \( O_i \). Note that an interface may contain nothing, called empty interface, if \( O, \partial, \Delta = \emptyset \). It is only used in the initial and the final UICs.

Definition 6.: (UIF Model, UIC, NCF). A UIF model, denoted as \( \Theta \), represents UI components and their relations, and it is tuple \( (\Sigma, \Omega, B, F) \) where,
Chapter 6. Deriving Process-Oriented User Interfaces

\[-\Sigma = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_x\}, \varepsilon_i \in \Sigma \ (1 \leq i \leq x) \text{ is a UIC},\]

\[\text{type: } \Sigma \rightarrow \{\text{initial, normal, final}\} \text{ is a mapping function that maps a UIC on a type of UIC},\]

\[-B = \{b_1, b_2, \ldots, b_y\} \cup \{b^{nil}\}, b_i \in B \ (1 \leq i \leq y) \text{ is an interface, where } b^{nil} \text{ denotes an empty interface},\]

\[\Omega \subseteq \Sigma \times B \text{ defines the relation between UICs and interfaces},\]

\[-F \subseteq \Omega \times \Sigma \text{ is a finite set of Navigational Control Flow (NCF) relations. A flow } f = ((\varepsilon_s, b_x), \varepsilon_t) \in F \text{ corresponds to a NCF relation between the source UIC } \varepsilon_s \text{ and the target UIC } \varepsilon_t, \text{ such that when } \varepsilon_s \text{ is active and every attribute in } \partial \text{ of interface } b_x \text{ is completed then } \varepsilon_t \text{ is enabled (activated) and } \varepsilon_s \text{ is disabled (deactivated)},\]

\[-F^* \text{ is reflexive transitive closure of } F, \text{ we write } ((\varepsilon_s, b_x)F^* \varepsilon_t \text{ or } \varepsilon_sF^* \varepsilon_t \text{ if there exists a sequence of flows from UIC } \varepsilon_s \text{ to UIC } \varepsilon_t.\]

### 6.2.2 Well-formed UIF model

Here, we discuss the well-formedness property of UIF models. Similar to behavioral soundness property of ACP models, UIF models should be safely-structured and behave correctly as desired based on an underlying business process it derives from.

We define a set of conditions for a *well-formed* UIF model.

**Definition 6.: (Well-formed UIF model).** Given UIF model \( \theta = (\Sigma, \Omega, B, F) \), \( \theta \) is said to be well-formed iff the following statements hold.

1. for every interface except empty interface, it must be assigned to one and only one UIC, i.e.,
   \[\forall b \in B \setminus \{b^{nil}\}, \exists \varepsilon_x \in \Sigma, \forall \varepsilon_y \in \Sigma \setminus \{\varepsilon_x\}, (\varepsilon_x, b) \in \Omega \land (\varepsilon_y, b) \notin \Omega\]

2. each normal UIC must have at least one, and not empty, interface assigned, i.e.,
   \[\forall \varepsilon \in \{\Sigma \mid \text{type}(\Sigma) = \text{normal}\}, \exists b \in B \setminus \{b^{nil}\}, (\varepsilon, b) \in \Omega\]
(3) the initial UIC must have an empty interface assigned and must have some outgoing flows to normal UIC, i.e.,

$$\forall \varepsilon_x \in \{\Sigma \mid \text{type}(\Sigma) = \text{initial}\}, \exists \varepsilon_y \in \{\Sigma \mid \text{type}(\Sigma) = \text{normal}\}, (\varepsilon_x, b_{\text{init}}) \in \Omega \land ((\varepsilon_x, b_{\text{init}}), \varepsilon_y) \in F,$$

(4) there must be no flows coming into the initial UIC, i.e.,

$$\forall \varepsilon_x \in \{\Sigma \mid \text{type}(\Sigma) = \text{initial}\}, \forall \varepsilon_y \in \Sigma, \forall b \in B, ((\varepsilon_y, b), \varepsilon_x) \notin F,$$

(5) the final UIC must have an empty interface assigned and must have some incoming flows from normal UIC, i.e.,

$$\forall \varepsilon_x \in \{\Sigma \mid \text{type}(\Sigma) = \text{final}\}, \exists \varepsilon_y \in \{\Sigma \mid \text{type}(\Sigma) = \text{normal}\},$$

$$\exists b_m \in B, \exists \varepsilon_y \in B, ((\varepsilon_y, b_{\text{init}}), \varepsilon_x) \in \Omega \land ((\varepsilon_y, b_m), \varepsilon_x) \in F,$$

(6) there must be no flows going out from the final UIC, i.e.,

$$\forall \varepsilon_x \in \{\Sigma \mid \text{type}(\Sigma) = \text{final}\}, \forall \varepsilon_y \in \Sigma, ((\varepsilon_x, b_{\text{init}}), \varepsilon_y) \notin F,$$

(7) each normal UIC must be in some sequences between initial and final UICs, i.e.,

$$\forall \varepsilon_x \in \{\Sigma \mid \text{type}(\Sigma) = \text{normal}\}, \exists \varepsilon_x \in \{\Sigma \mid \text{type}(\Sigma) = \text{initial}\},$$

$$\exists \varepsilon_y \in \{\Sigma \mid \text{type}(\Sigma) = \text{final}\}, \varepsilon_x F^* \varepsilon_y \land \varepsilon_y F^* \varepsilon_x.$$

### 6.3 Deriving UIF Model for ACP Model

For the two aspects of UIF models, more precisely, the behavioral aspect is represented by its UI Components and their NCF relations, while the informational aspect is represented by internal information of artifacts required for each interface. Once we have an ACP-i model defined, then the next step is to derive UIF models from the ACP-i model. We divide the derivation into two main steps: (1) generating the interfaces and their NCF relations for constructing the behavior of the model, and (2) mapping the required artifacts and their attributes.
for constructing the information for each interface. These steps are described in Section 6.3.1 and 6.3.2, respectively. We name UIF-base model for the models that directly derived from an underlying ACP-i model.

### 6.3.1 Deriving the behavior of UIF model

The behavioral aspect of UIF-model can be obtained from the behavior of ACP-i model. First we generate the lifecycle of ACP-i model by using ACP lifecycle composition function $compose_ACP$ (in Definition 3.), which is developed based on the three inferences rules defined in Definition 3. Then, we perform a behavioral mapping from the ACP lifecycle to the UIF-model. The entire process from generating ACP lifecycle to the mapping from the lifecycle to the UIF model can be performed by using function $mapB/UIF$ described in Algorithm . The result of the function is the behavioral aspect of the UIF model.

The procedure of the behavior mapping is divided into two parts: (1) states to UICs mapping and transitions to interfaces mapping. (2) NCF relation generation. There can be multiple interfaces in a single UIC if such state has multiple exit transitions. The algorithm discards a state with automatic transition as it does not require any input from users, so the navigational control flows are rerouted to its succeeding UIC.

![Algorithm](function mapB/UIF). Generating the behavioral aspect of UIF-base model $\theta$ from ACP-i model $\Pi$.

**Input:** ACP-i model $\Pi = (Z, V, R, L, Y)$

**Output:** UIF-base model $\theta = (C, \Omega B, F)$

- $\Sigma, \Omega B, F = \emptyset$;
- $\Sigma \leftarrow \{s_{init} | type(s_{init}) \leftarrow initial\}$;
- ACP lifecycle $L_\Pi = (S_c, l_{init}, \Rightarrow_c)$;
- $L_\Pi = compose_ACP(\Pi)$;

**for each** state $s_i \in S_c$ **do** // $s_i$ is the combined state of all lifecycles in $L_\Pi$

**for each** transition $\Rightarrow = (s_i, v, s_j) \in \Rightarrow_c$ **do**

- let interface $b = (O, \delta, \Delta, v, l)$;
- $b. v \leftarrow \Rightarrow. v$;
- $b.l \leftarrow \{y(r) | \exists r \in R, b. v = r. v\}$;
- $b. \Delta \leftarrow \bigcup_k s_k$ for each state in $s_i(s_1, s_2, ... s_k)$; //assign lifecycle’s states to interface’s states
- $B \leftarrow \{b\}$;
Chapter 6. Deriving Process-Oriented User Interfaces

\[ \Omega \leftarrow \{(a_i, b)\} \quad \text{//assign interface } b \text{ to UIC } a_i \text{ and add to relation set } \Omega \]

if \( \forall s \in S, (s_f, v, g, s) \notin \epsilon \) then type(\( \epsilon \)) = final else type(\( \epsilon \)) = normal ;

if \( s_i = s_i^{\text{init}} \) then \( F \leftarrow \{(a_{\text{init}}, b^{\text{init}}, a_i)\} \); //a flow from initial UIC to the first normal UIC

end for

\[ \Sigma \leftarrow \{a_i\} \quad \text{//assign for} \]

end for

for each UIC \( a_x \in \{ \Sigma \mid \text{type}(a_x) \in \{\text{normal}, \text{final}\} \} \) do

for each interface \( b_a \in \{B \mid \exists(a_x, b_a) \in \Omega\} \) do

if \( \exists(s_n, b_a, v, g, s_n) \in \epsilon \) then \( \exists(s_n, b_a, v, g, s_p) \in \epsilon \) then \( \exists b_y \in \Sigma, (a_x, b_a) \in \Omega \)

then \( \leftarrow \{(a_x, b_a), b_y\} \); //assign a flow from \( b_a \) in UIC \( a_x \) to UIC \( a_y \)

end if

end for

end for

THEOREM 6.. Given ACP-i model \( \Phi \), mapB_UIF(\( \Phi \)) returns well-formed UIF model \( \theta \) for \( \Phi \).

The proof of THEOREM 6. can be found in the Appendix.

Next, we show a running example when applying the mapB_UIF function to an ACP model (in Example 6., Example 6., and Example 6.). For the simplification of illustration, we introduce a simple purchasing process presented in Figure 6., which is another version of our fully-detailed purchasing process introduced in Section 3.1.

Example 6. Figure 6. shows the lifecycles of all artifacts that are used in the simplified purchasing process with two roles involved: Customer (Buyer) and Retailer (Supplier). Notice for some transitions, we attach a role to the task to indicate the responsible organization that is going to perform the task on the transition.
Example 6. Figure 6. shows the ACP lifecycle that is composed from all artifact lifecycles shown in Figure 6. We can see that each composite state in the composed lifecycle comprises every of each state of artifact that is defined in the process.
Figure 6.: ACP lifecycle of the simplified purchasing process

Example 6. Figure 6. depicts an example of a diagram representing behavioral aspect of UIF-base model for our purchasing processes. We also indicate which role of users is permitted to operate on which interface by using the shaded rectangles to represent interfaces used by Supplier and white ones represent Buyer’s interfaces. We can see that each UIC contains various relationships of interfaces and performer’s roles: (1) one single interface to one role ($\varepsilon_1$), (2) multiple interfaces to one role ($\varepsilon_2$), (3) multiple interfaces to multiple roles ($\varepsilon_3$, $\varepsilon_4$, $\varepsilon_5$). Multiple interfaces for the same role can be grouped together as shown in $\varepsilon_3$. 

Chapter 6. Deriving Process-Oriented User Interfaces
Here, we can see that the generated behavior aspect for an UIF-base model precisely captures the operation sequences of its underlying business process model it derives from. Note that Algorithm does not provide a mapping of artifact data and interfaces. The information mapping will be provided in Section 6.3.2.

### 6.3.2 Mapping the information of artifacts and business rules to UIF model

Once we completed the generation of the behavioral aspect of UIF-base model, then we need to generate its informational aspect by assigning artifacts based on the conditions of related business rules onto interfaces. In this step, we need to find all the corresponding artifacts required for each interface. We can classify the information needs for a particular interface into: (1) a set of artifacts to be read or updated, and (2) a set of required values to be assigned to attributes of such artifacts. We can simply find both sets by extracting them from every condition of business rules that corresponds to a task to be executed of such interface. We present function map$_{U1F}$ for the information mapping and its procedure is illustrated in Algorithm .
Algorithm (function map1 UIF). Mapping information from ACP-i model Π to UIF-base model θ

Input: ACP-i model Π = (Z, V, R, L, γ) and UIF model θ = (Σ, Ω, B, F)
Output: Attribute set θ and artifacts O for each interface b ∈ B in the UIF model θ

for each interface b_i ∈ B do
    for each business rule r_j ∈ R do
        let state set S = ∪_{o_e ∈ p r e (r_j, o_k)};
        if r_j.v = b_i.v ∧ (∀s ∈ S, s ∈ b_i.Δ) then
            let scalar comparison predicate set SC = {C.a | C.a is compared with a variables x or other artifact’s attribute in r_j.λ, e.g., C.a = x}
            let defined predicate set DC = {C.a | defined(C, a) in r_j.λ}
            let artifact set O_t = Ø;
            for each term τ in DC ∪ SC do
                O_t ← τ.C ;// add artifact C in term τ to the artifact set
                b_i.∂ ← τ.C.x ; // add attributes x in term τ to the attribute set
            end for
            for each artifact C ∈ {C} in state (C, s) in r_j.λ do
                O_t ← C;  // add artifact C required for state condition to the artifact set
            end for
            O ← O_t ;  // assign set of required artifacts to interface b_i
        end if
    end for
end for

The algorithm first retrieves a set of business rules that corresponds to each interface. This can be done by matching the task action in each rule with the task of interface in accordance of the state conditions that such interface holds (as there might be many occurrences of the same task execution in multiple business rules).

We require that the state of an interface conforms to every state condition in every corresponding business rule for such interface. Then the artifacts and required attribute set are extracted from the condition of each rule. This step is done iteratively until every business rule satisfying the condition in line 4 is parsed. Note that the required attribute set and artifacts for each interface are minimal and sufficient. They can be extended if users would like to incorporate other related artifacts by adding them into the interface; however, these additional artifacts need to be validated as to ensure that the behaviour consistency between UIF and ACP-i model is preserved.

From the above, we can say that our proposed information mapping explicitly overcomes the drawbacks of current approaches in which activities, data and their
relation are treated separately. Note that as the information mapping algorithm does not involve in changing the behavior of an input UIF model, so the well-formedness property of the model is unaffected.

6.3.3 Deriving user interfaces for a particular role

The UIF-base model discussed above is generated for all users, and users in the organization may have different roles. Due to the distribution of tasks in inter-organizational business processes and authority reasons, it is necessary to build customized UIs for users of a particular role (within and across organizations). As the UIF-base model does not explicitly show clear flows of UIs for a particular role, we customize the base model to the *UIF-role model* which can be constructed by extracting interfaces and their relations from the UIF-base model for a specific role. It is noted that the derived UIF-role model (from UIF-base model) corresponds to role-based UIs of an organization that derives from its local ACP model. Given that the UIF-role model is generated by further deriving its base model, both behavior and information consistency must be preserved to guarantee the correctness of the generated model.

Before generating UIF-role models, we need to define the sequence of the interfaces.

*Definition 6.* (Interface sequence). The interface sequence is the ordered list of interfaces $\langle b_1, b_2, \ldots, b_n \rangle$. Given UIF-base model $\theta = (\Sigma, \Omega, B, F)$, a set of interface sequences from interface $b_x \in B$ to interface $b_y \in B$, denoted as $SEQ_{b_x}^{b_y} = \{\langle b_x, \ldots, b_i, \ldots, b_y \rangle, \ldots, \langle b_x, \ldots, b_j, \ldots, b_y \rangle\}$, such that,

$$\exists b_x, b_y \in B, \exists \epsilon_x, \epsilon_t \in \Sigma, \exists (\epsilon_x, b_x), (\epsilon_t, b_y) \in \Omega, \epsilon_t F^* \epsilon_t.$$  

In addition, let $SEQ_{b_{\text{init}}}^{b_{\text{final}}}$ denote a set of every possible sequence of interfaces from the initial interface(s) (following from *initial UIC*) to the final interface(s) of the UIF-base model. We also define some necessary functions to be used for generating UIF-role models:
Chapter 6. Deriving Process-Oriented User Interfaces

—function \( \omega(SEQ_{b_x}^y, l) \) returns a set of interface sequences for role \( l \) by removing interfaces that not belong to role \( l \) out of the set. Function \( \omega \) preserves the order of interfaces, i.e., the order of interfaces in each sequence in \( \omega(SEQ_{b_x}^y, l) \) is consistent with the order of interfaces in each sequence in \( SEQ_{b_x}^y \).

—function \( \text{first}(SEQ_{b_x}^y) \) returns a set of \textit{first-ordered} interfaces from each sequence in \( SEQ_{b_x}^y \).

—function \( \text{last}(SEQ_{b_x}^y) \) returns a set of \textit{last-ordered} interfaces from each sequence in \( SEQ_{b_x}^y \).

**Definition 6.** (UIF-role model). Given ACP-i model \( \Pi = (Z, V, R, L, \gamma) \) and well-formed UIF-base model \( \theta = (\Sigma, \Omega, B, F) \), we can derive UIF-role model for role \( l \in L \) from \( \theta \), and it is denoted as \( \theta^l = (\Sigma^l, \mathcal{U}^l, B^l, F^l) \), where \( B^l \subseteq B \) such that the following statements must be satisfied.

1. \( \theta^l \) is well-formed,
2. each interface of \( \theta^l \) must belong to role \( l \), i.e.,
   \[ \forall b \in B^l, b.l = l \]
3. each sequence of interfaces in \( \theta^l \) preserves the order consistency with the corresponding sequence in \( \theta \), i.e.,
   \[ \forall b_x \in B_l \cap B, \forall b_y \in B_l \cap B, SEQ_{b_x}^y \equiv \omega(SEQ_{b_x}^y, l) \]
4. each first-ordered interface of each sequence in \( SEQ_{b_{final}}^{b_{initial}} \) must be assigned to the normal UIC that follows the initial UIC, i.e.,
   \[ \forall b \in \text{first}(SEQ_{b_{initial}}^{b_{final}}) \subseteq B^l, \exists b_y \in \{ \Sigma^l | \text{type}(\Sigma^l) = \text{initial} \}, \exists \epsilon_y \in \{ \Sigma^l | \text{type}(\Sigma^l) = \text{normal} \}, \exists (\epsilon_y, b) \in \mathcal{U}^l, ((\epsilon_y, b^{n^l}), \epsilon_y) \in F^l \]
(5) each last-ordered interface of each sequence in $SEQ^{b_{final}}_{b_initial}$ must be assigned to some normal UIC(s) and followed by the final UIC, i.e.,

$$\forall b \in \text{last}(SEQ^{b_{final}}_{b_initial}) \subseteq B^l, \exists \epsilon_x \in \{\Sigma^l| \text{type}(\Sigma^l) = \text{final}\}, \exists \epsilon_y \in \{\Sigma^l| \text{type}(\Sigma^l) = \text{normal}\}, \exists (\epsilon_y, b) \in \Omega^l, \exists (\epsilon_x, b^{a_{\text{ui}}}i) \in \Omega^l, ((\epsilon_y, b), (\epsilon_x, \epsilon_x)) \in F^l.$$

The rules defined in Definition 6. above show behavioral restrictions for the derivation of the UIF-role model from its base UIF model. They are used to guide the extraction procedure and to guarantee that the extraction always produces a well-formed UIF-role model, as we shown in Theorem 6..

**Theorem 6.** Given ACP-I model $\bar{\Pi}$, UIF-role model $\theta^l$ for role $l \in \bar{\Pi}.L$ is well-formed.

The proof of Theorem 6. can be found in the Appendix.

Each interface in an UIF-role model contains the same information (artifacts and required attributes) as it is in the base model. As an individual UIF-role model represents for only a single role, the interface or UIC dependencies between its role and other roles are not considered in the model. However, such dependencies can be determined by tracing its base model.

**Example 6.** Figure 6. shows an example of UIF-role models for (a) Supplier and (b) Buyer extracted from its UIF-base model based on Definition 6. with the use of Algorithm  and Algorithm . In Figure 6. (a), we can see that Supplier may start with any one of two interfaces: createShipment or holdItem. If we interpret the base model, the createShipment interface can start immediately at the beginning, or after Buyer completes either createOrder or addItem interface. Similarly, the holdItem interface must wait until the Buyer has followed the sequence of createOrder and addItem interfaces. Figure 6. (b) illustrates the information of the interface according to the artifact attributes and states of the UIF-role model for the Buyer. The white input fields indicate that a customer has to fill in the required data, while the gray fields contain values from its previous interface. All required inputs have to be filled to complete the interface.
6.4 ACP Execution System Architecture with UIF Support

The use of workflow management system in organizations has been considered as a promising automation approach to enable them to design, execute, monitor and control their business processes. It is conceived that current workflow technologies support organization’s own developed web applications for users to efficiently interact with the processes they involve. With the Model-driven approach, user interfaces (UIs) of these applications can be generated based on functional and data requirements obtained from underlying process models. Here, we briefly present the system architecture for the implementation for realizing ACP model and automatic UI derivation framework, as illustrated in Figure 6..
Chapter 6. Deriving Process-Oriented User Interfaces

Our ACP execution system prototype has been developed based on the ACP realization framework presented in [Ngamakeur et al., 2012]. The system requires an ACP model specification, which is specified in an XML format, as an input to execute. Our business rule engine developed on the top of the Drools engine [JBoss community, 2012] is a core component of the system providing functionalities to deal with business rules (e.g., loading and interpreting business rules, event-handlers and listeners), and invoke tasks (or web-services). There are some other components such as process deployer, and artifact and web-service controllers, etc., that are parts of the back-end process interface of the system.

In order to support the automatic generation of user interfaces, the UIF generator has been designed and integrated with XSL Transformation (XSLT) [W3C, 2010] processor into the main system. It is used for generating the meta-code of UIF mode called UIFML from an input ACP model and from predefined XSL designed templates. Similar to ACP model specification, a UIFML is also specified in an XML format. The UIF generator uses Algorithm and Algorithm to produce the UIFML document, which represents the structure of pages and the flows between such pages. Based on the generated UIFML, UI modelers design a set of concrete web design templates in XSL format (plus Cascading Style Sheets (CSS) [W3C, 2011]) corresponding to each page (or a collection of pages). Note that the XSL templates can be designed without a concern of business logic.
When a business process is executing, the system calls XSLT processor to read provided XSL templates and UIFML document generated for the process, transforms a current page corresponding to the current step of the business process into HTML page, and then displays the page to process users. Once users completed and submitted a form in the current showing page, a task specified in such form is invoked, and then brings the users to the next page specified in the UIFML. Users will be guided through the navigation of pages until the process finishes. It is noted that, currently, the system is in an ongoing development and cannot provide a role-based support for generating user interfaces that are distributed among different organizations. However, at least we can demonstrate the usefulness of our approach at some desired level that promotes the benefit of using artifact-centric approach for the generation of user interfaces in the spirit of MDA paradigm.

6.4.1 ACP specification format

For the ACP specification, the definition of business rules is defined based on the extension of RuleML [RuleML Initiative, 2011] standard. The structure of our ACP business rule definition is illustrated in Figure 6. with the example of business rules for the purchasing process in an XML format as shown in Figure 6.
We can see from the format of ACP business rules, the pre-condition and post-conditions are defined based on two types of atoms (on artifacts): attribute's value and state's value. We do not explicitly define the action part of a business rule. This is why we use the post-condition of a business rule to identify the task to be invoked (if the pre-condition of such rule holds). The post-condition also defines the after (or target) state of each artifact that a task will update. Each target state must be changed after the task is successfully performed. To limit the scope of the system, we assume that there is no cases of failure of task execution; thus, the post-condition always satisfies once the task finishes its execution.

6.4.2 UIF specification format (UIFML)

For the UIF specification, we structure UICs (pages), their interfaces (forms), and input fields in the XML format called as UIFML, as exemplified in Figure 6.. The navigational flow between two pages (i.e., a link from a form of one page to another page) is specified by the ID of the target page in the “targetpage” attribute of the form element.
As discussed, with the UIFML document serving as a meta-template of user interfaces structure and their flows, we can generate a series of concrete HTML pages from the UIFML by using XSLT (and possibly together with CSS templates). With our approach to generating (web-based) user interfaces from an artifact-centric business process model, we can view the structure and activity flows of the process in a very intuitive way as well as support the development of concrete user interfaces in the MDA-based style.

6.5 Summary and Discussion

This chapter discussed a MDA-based automatic UI derivation framework for business processes based on the artifact-centric approach. In the framework, the \textit{ACP-i model} and the \textit{UIF model} are used with a mechanism that automatically derives the UIF model from the \textit{ACP-i model}.

The UIF-base and UIF-role models are defined to support the user interface generation for enabling the visualization of the process and information flow to process users. Apart from that, UIF models intuitively represent what information is required during the process and how user-interface designers can use this generated conceptual model to build concrete user interfaces. Particularly, the UIF-role model provides role-based UI requirements for UI designers to build templates of concrete user interfaces for a particular role. With this framework, it enables
further improvement for supporting wider user interface requirements, e.g.,
supporting optional data elements, role-based configuration and customization.
Chapter 7.

Conclusion and Future Work

In this final chapter, we conclude the thesis by summarizing the main contributions in Section 7.1 and providing a discussion on open problems and an outlook on possible future research in Section 7.2.

7.1 Summary of Contributions

In this thesis, we identified three key challenging issues for the support of artifact-centric business process modeling: (1) support for business process reuse, (2) support for efficient inter-organizational business process collaboration, and (3) support for process visualization and user interfaces development. Based on the Artifact-Centric Process (ACP) model introduced and discussed in Chapter 3, we propose a novel approach, which is presented in each chapter, to tackle each of the three issues.


7.1.1 Support for business process reuse

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, 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. Synchronization dependencies between artifacts become a major concern as they can lead the behavior of a specialized process inconsistent with (and unobservable from) the behavior of its base process. Especially, it raises a non-trivial issue when an artifact and its dependency can be added, modified, or removed in the specialized process.

In Chapter 4, we proposed the process specialization framework for specializing Artifact-Centric Process (ACP) model, which comprises several dependent artifact classes. Three different specialization methods were defined: artifact refinement, artifact extension, and artifact reduction. We studied the dynamic behavior of artifact class and the interaction (via synchronization dependency) behavior of artifacts in a process, and formulated a set of rules that is used to preserve the behavioral consistency, called B-consistency, between a specialized process and its base process. Particularly, we presented our fragmental behavioral analysis technique with supporting theorems to check the interaction behavioral consistency between synchronized fragments of artifacts in the specialized process and in the base process. With the notion of specialization, process modelers can reuse existing artifact-centric process models at both artifact level and process level. All different specialized process models of a particular generic process model can be consistently observed at different level of specialization hierarchy, thus, allowing both artifact and process instances of a specialized process to be aggregated and reported at the more generic level.

7.1.2 Support for efficient inter-organizational business process collaboration

In artifact-centric approach, the flexibility is naturally achieved as it is deemed as one of the benefits of artifact-centric models. However, a comprehensive study on
supporting organizations to achieve all the three collaboration requirements (compliance, flexibility, and autonomy) is still missing. Based on existing literature and practices, we observed that the existing view-based concept for inter-organizational business process management can provide a promising and efficient way of modeling and change validation to address such requirements, e.g., public-private view approaches [Van der Aalst and Weske, 2001], process-oriented contract for service choreographies [van der Aalst et al., 2010], artifact-centric hub [Hull et al., 2009], artifact-centric choreographies [Lohmann and Wolf [2010]; nevertheless, the view-based approach has not been yet adequately explored in the context of artifact-centric inter-organizational business processes.

In Chapter 5, we proposed the view framework for artifact-centric inter-organizational business processes. The framework comprises artifact-centric model for inter-organizational business processes called ACP-i model, the notions of public views and private views, and the view consistency rules. An ACP-i model is used to capture the specification of a complete inter-organizational process. A public view is used to serve as an agreed contract of the process (defined in ACP-i model) at the more abstract level. A private view is used to capture an organization’s local process and to permit internal changes of the local process. Based on our B-consistency notion, we proposed mechanisms to construct a behavior-consistent public view and an algorithm that helps organizations automatically find the most abstract public view based on their local processes, and ensure that changes on local process preserve the correctness and behavioral consistency of global process. Particularly, our public/private view approach permits the process verification to be achieved locally, thus, avoiding the state exposition problem that may occur in the global verification.

7.1.3 Support for process visualization and user interfaces development

Unlike a conventional activity-centric approach, in the artifact-centric approach, expressing business process logic by using declarative-style business rules makes users difficult to see and understand the structure and flow of the process, especially a business process can be distributed in cross-organizational environment. In the user-centric aspects, this brings in a key challenge to have a
natural approach for representing such declarative process models to non-technical stakeholders and users of the processes [Hull, 2008]. By having a closer look at artifacts and user-interfaces, we observed that on the one hand a user-interface is used for users to view/input business artifact data and invoke a related function that will affect to the process; on the other hand, the artifact data and business rules decide which user-interface should be brought to the users to fulfil a certain procedure of the process.

In artifact-centric setting, there are two related existing works from Deutsch et al., [2007] and Sukaviriya et al. [2007]. The former studied data-centric web application while the latter studied a MDA-based generation of UIs with the consideration of artifact data. Küster et al., [2007] proposed an approach to generating business process models from object lifecycles, which can be used to complement our UIF model generation. However, there is still no study on how to automatically generate UIs from artifact-centric process models.

In Chapter 6, we proposed the User Interface Flow (UIF) model and algorithms to automatically derive a UIF model from an underlying ACP-i model with a view-based support (for different roles of local/inter organizational users). The UIF model is a conceptual model used to represent the logical structure of user interfaces and their navigation flows, which can be used for the purpose of visualizing business processes as well as facilitating the user-centric model-driven development and implementation of concrete user interfaces for business processes. On the one hand, it provides an intuitive process structure, likewise an activity/control-flow structure, for artifact-centric models, thus, helping process users and non-technical stakeholders easily understand the flow of the process. On the other hand, it supports different roles of users, and therefore, allowing different roles to see and navigate through only their accessible user interfaces based on the specification of the process. In addition, the model facilitates the MDA-based development of actual (concrete) user interfaces. We serialized a UIF model in the XML format so that it can be used to generate actual HTML webpages via the use of XSLT together with CSS design templates.
Chapter 7. Conclusion and Future Work

7.2 Open Problems and Future Work

Although in this thesis we provided a comprehensive discussion on the approaches for supporting artifact-centric business process modeling, there are some issues remain open for further investigation.

—*Modeling languages and tools.* In our work, we did not focus on the development of modeling language nor restrict our framework to any language. We used a simple modeling syntax for capturing the specification of a business process according to the artifact-centric business process framework presented in [Cohn and Hull, 2009; Hull, 2008]. In the essence of artifact-centric approaches, rule-based languages are deemed most suitable in its spirit to achieve highly-flexible process specifications. We introduced our ACP specification by using simple condition-action-styled business rules to define the control logic of business processes. This kind of rules can be extended to fully-fledged ECA rules in more declarative style with temporal constraints (e.g., CTL [Gerede and Su, 2007], TiLe [Zhao et al., 2009], and LTL-FO [Damaggio et al., 2011]). As our work only considers the behavioral aspect of business processes, the interpretation of business rules to generate (LTS) lifecycles of artifacts is required (e.g., for our rule syntax, the lifecycle can be constructed from Definition 3). Especially, the verification of business rules (in a particular language) is required to explore for each context of the three different support’s issues. In addition, the modeling tools need to be further developed to facilitate and help process modelers define, construct, and verify artifact-centric process models. In our work, we manually captured models in XML documents for capturing the specification of artifacts, tasks, and business rules. There should be a more graphical and intuitive way to support all the modeling and all other related tasks, particularly, for defining the specialization of ACP models with automated $B$-consistency checking.

—*Abstraction of semantic web service specification.* As already discussed in Section 3.2, the definition of tasks (or services) in artifact-centric approaches can be naturally described in a semantic manner (based on OWL-S [2003]). For the abstraction of artifacts, we assumed that the pre- and post-conditions of tasks to be abstracted conform to the pre- and post-conditions of business rules that
associate such tasks with the artifacts. Therefore, there is no need to check whether such abstraction is valid. However, there may be the case that the conditions of tasks do not strongly conform to the conditions of business rules; hence, the conditions of abstract task and corresponding abstract rule(s) can be inconsistent. This requires the abstraction to include an additional validation mechanism in order to handle such issue.

—Workflow realization. A realization for our ACP-i models can be possibly achieved by considering two different approaches featured with public view definition and private view validation mechanism. The simple solution is to transform from ACP models to process-centric models (e.g., in [Liu et al., 2010; Li and Wu, 2011]) with the use of model-transformation techniques presented in [Küster et al., 2007; Redding et al, 2008], and then realize them on existing workflow systems plus view features (e.g., in [Jiang et al, 2010]). However, this solution does not seem very promising as the transformation poses several drawbacks (as discussed in [Ngamakeur et al., 2012]). Cohn et al. [2008] presented Seina prototype that allow users to model business artifacts and process as an XML documents. The work on Active XML (AXML) [Abiteboul et al., 2008, 2009; Marinoiu et al., 2010], which is an extension of XML with embedded service calls, can be used for supporting the implementation of artifacts that are accessed within or among organizations. Influenced by these work, we aim to develop the pure artifact-centric process system for realizing our artifact-centric view framework. Currently, we are developing a prototype based on the realization framework in [Ngamakeur et al., 2012] with the support of shared/local XML-based artifacts.

In summary, there are several remaining issues to be discussed for supporting business process modeling in the artifact-centric paradigm. Our work presented in the thesis can be seen as a significant step leading to further advanced investigation of artifact-centric approach in different directions towards more efficient modeling and management of business processes. As several aspects of future works can be extended based on our study, we hope our work to be beneficial to the broader area of business process management and software engineering.
APPENDIX

In this appendix, we provide proofs for the lemmas and theorems presented in the thesis.

PROOFS OF THEOREM 3. and THEOREM 3. Both theorems can be proved by using Definition 3. and Definition 3. As only one (non-sync) business rule can be used for a single transition in a lifecycle and if the rule is defined in one lifecycle then that rule must not be used in another lifecycle, therefore, the safety property (in Definition 3.) is always satisfied by the composition. For THEOREM 3., given two lifecycles to be composed, only two inference rules 3.1 and 3.2 (in Definition 3.) are used to generate the composed transition in the composed lifecycle of such two lifecycles as there is no synchronization dependency from the two lifecycles. Therefore, there is no possibility that the goal-reachability property of the composed lifecycle can be violated. For THEOREM 3., which is in contrast to THEOREM 3., inference rules 3.3 (in Definition 3.) is applied to generate the composed lifecycle due to the existence of sync rules. The sync rule can lead a dead-lock situation when two lifecycles are composed as the transition (which is triggered by that rule) of one lifecycle depends on an unreachable state of one another lifecycle. Therefore, the goal-reachability cannot be guaranteed for the composed lifecycle.

PROOF OF LEMMA 4. We can prove the lemma by induction over the conditions of an AL-fragment (in Definition 4.), the three inference rules for lifecycle composition (in Definition 3.), and the soundness's condition (in Definition 3.). We prove the necessity of each of the four conditions as follows.

—For the first condition. We have, if $\ell \in \Gamma$ is not an AL-fragment, then the resulted fragment from the composition of $\ell \in \Gamma$ 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., if $\ell \in \Gamma$ 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 as the AL-fragment is always sound and no synchronization is stated in this condition.

The second, third, and forth conditions of LEMMA 4. are used to restrict two SL-fragments (to be composed for S-region) to include every transition and
corresponding sync rule that are used for only the synchronization between L-
fragments in \( \Gamma \). We prove that these 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., 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 \( \ell^c_i \) with any L-fragment that is not in \( \Gamma \), 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
some state that does not belong to any L-fragment in \( \Gamma \); 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., 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 for 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..

Proof of Theorem 4.. The theorem can be proved by: checking for each
statement of the theorem, a refined L-fragment (in Definition 4.) does not violate
the \( B\)-consistency's condition (in Definition 4.).

—Consider the first statement of the theorem. An AL-fragment that refines a base
lifecycle always preserves the \( B\)-consistency's condition as the L-fragment is
atomic with single-entry state and single-exit state. The entire L-fragment can be
completely reduced (or abstracted) in a single transition if such L-fragment is atomic. We can see that the condition of AL-fragment (in Definition 4.) naturally conforms to the condition of the $B$-consistency.

—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 in 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 the condition of the $B$-consistency.

This completes the proof of THEOREM 4. □

PROOF OF THEOREM 4.. The theorem can be proved by induction over the three inference rules for lifecycle composition (in Definition 3.), the soundness's condition (in Definition 3.), and the B-consistency's condition (in Definition 4.). As refined L-fragments do no introduce any synchronization dependencies to specialized artifacts, then the proof follows from the proof of THEOREM 3. - that is the composed lifecycle is always sound. As the conditions of THEOREM 4. restrict a specialized artifact preserves the $B$-consistency when applying refined L-fragments, then the composed lifecycle of all specialized artifacts is $B$-consistent. □

PROOF OF LEMMA 4.. We can prove it by induction over the ex-lifecycle condition (in Definition 4.), the $B$-consistency condition (in Definition 4.), the condition for atomic composition of SL-fragments (in LEMMA 4.), and the condition of $B$-consistent refined L-fragment (in THEOREM 4.). Revisiting the four conditions in LEMMA 4., 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., the AL-fragment condition conforms to the first condition of Lemma 4. and the ex-lifecycle condition (in Definition 4.) conforms to the second, third, and fourth conditions of the LEMMA 4.. Followed from THEOREM 4., the composed lifecycle can be considered as a refined,
composite AL-fragment (composed of $L'_c^y$ and $\ell$) in $\ell$; therefore, the composed lifecycle is $B$-consistent with $\ell$.

**Proof of Lemma 4.** Similar to the proof of Lemma 4., we can prove it by induction over Definition 4., the $B$-consistency condition (in Definition 4.), the condition for atomic composition of SL-fragments (in Lemma 4.), and the condition of $B$-consistent refined L-fragment (in Theorem 4.). This proof can be achieved based on the proof of Lemma 4.. If the composition of two synchronized fragments in the refined S-region is atomic, then based on Theorem 4. (by considering refined S-region as a refined L-fragment), the $B$-consistency is preserved.

**Proof of Lemma 4.** The proof can be derived from Lemma 4. and Lemma 4. as the refinement of existing artifact satisfies the condition of Lemma 4. and the extension of artifact satisfies the condition of Lemma 4..

**Proof of Lemma 4.** The proof can be derived from Lemma 4. and Definition 4. by taking into account the transitivity property of sync rules (in definition Definition 4.) and the lifecycle composition (in Definition 3.).

**Proof of Lemma 4.** We can prove it by induction over Definition 4., Definition 4., Definition 4., the $B$-consistency condition (in Definition 4.), and the condition for atomic composition of SL-fragments (in Lemma 4.). For every reducible L-fragments in $L'^\circ$, it is reduced into a transition. Based on Definition 4., Definition 4., and Lemma 4., 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.** We can prove the theorem as follows.

—For the if condition, we have to prove that if the two statements are satisfied, then $\Pi'$ is $B$-consistent with $\Pi$. For the first condition, we prove that why each specialized artifact needs to be $B$-consistent with its base artifact. Based on Definition 4., this statement follows from Theorem 4.. For the second condition, we prove that why each sync specialization needs to be $S$-consistent. Based on Definition 4., we this statement follows from Lemma 4., Lemma 4., Lemma 4., and
LEMMA 4. 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 have to prove that if $\Pi'$ is a behavior-consistent specialization of $\Pi$, then the two statements satisfy. This can be proved based on the definition of ACP Specialization (in Definition 4.) and the definition of lifecycle specialization $B$-consistency (in Definition 4.), and $S$-consistency (Definition 4.). 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.. Then, consider the three $S$-consistency conditions based on artifact all of these specialization methods.

—For artifact extension. A new added artifact is not needed to be $B$-consistent. Either LEMMA 4. (sync extension) or LEMMA 4. (sync refinement of existing and extended artifacts), or the combination of them is required.

—For artifact refinement. If there is no synchronization taken into account, the $B$-consistency for artifact refinement follows from THEOREM 4.. With synchronization, THEOREM 4. and LEMMA 4. (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.. and LEMMA 4. (sync reduction).

This completes the proof of the only if direction.

Therefore, the proof of THEOREM 4. is complete. 

PROOF OF THEOREM 4.. The theorem holds only the first statement of THEOREM 4. which is insufficient. 

PROOF OF THEOREM 5.. We can prove the theorem by applying $B$-consistency checking (in Definition 4.) to compare between the input lifecycle and the output (abstract) lifecycle based on the three use cases of those two abstraction functions.
Consider the first two cases (a) and (b) in Definition 5. We know that abstracting an \textit{AL-fragment} always produces a single abstract transition or abstract state (with single entry transition and single exit transition), therefore, preserving the atomicity of the fragment to be abstracted. Thus, the output lifecycle is consistent with its base. For the third case (c) in Definition 5., we generate a single abstract state with the restriction on its abstract entry transitions and exit transitions. Similar to case (b), an abstract state represents the internal behavior of an abstracted fragment and it is itself atomic.

\textbf{Proof of Theorem 5.} We can prove the theorem by using the composite S-region (in Definition 4.) and the AS-region (Definition 4.) for the \textit{B-consistency} checking between two abstract lifecycles and their base lifecycles, and between the composition between the former and the composition between the latter. From the sync abstraction definition (in Definition 5.), sync abstraction \textit{sa.f} always returns a single abstract transition in each abstract lifecycle and a single sync rule between two abstract transitions, so the composition of such abstract transitions always yields an atomic composite fragment. The composition of ASL-fragments in their base lifecycle is atomic and can entirely be represented by the composition of abstract transitions.

\textbf{Proof of Theorem 5.} The proof is straightforward as the theorem is derived from Theorem 5. and Theorem 5..

\textbf{Proof of Theorem 5.} We can prove the theorem as follows. First, we prove that \textit{find_minPV(\bar{\Pi})} returns a correct public view of \bar{\Pi}. This can be done by induction over all mapping conditions for the \textit{ACP abstraction} (in Definition 5.). Note that removing local artifact from the public view where its entire lifecycle can be abstracted also conforms to the definition of public view. Second, we prove that given \textit{ACP:}\textit{i model} \bar{\Pi}, \textit{find_minPV(\bar{\Pi})} is guaranteed to return minimal public view of \bar{\Pi}. As we use function \textit{find_minASR} to search for the minimal AS-region that can be used as an input of the abstraction function \textit{sa.f}, then this statement is naturally satisfied. Last, we prove the \textit{B-consistency} of the generated public view. Based on Theorem 5. and the sync abstraction function (in Definition 5.), the public view preserves \textit{B-consistency} as function \textit{sa.f} always yields a \textit{B-consistent} abstract ACP model of the input ACP model.

184
PROOF OF THEOREM 5. The proof of the theorem is straightforward as it can be derived from Definition 5. and Definition 5. □

PROOF OF THEOREM 6. We can prove the theorem by checking the algorithm of the mapB_UIF function (in Algorithm) against each of the conditions for well-formed UIF model (in Definition 6). All conditions in Definition 6. must be satisfied from the algorithm.

—Conditions (1) and (2) are satisfied as an interface and a corresponding UIC that it belongs to are derived from the composite state of the ACP lifecycle.

—Conditions (3) and (4) are held from the algorithm as it searches for the entry and exit transitions of an initial composite state (no any incoming transition) and maps the transitions to the outgoing flow from an initial UIC to some target UICs.

—Conditions (5) and (6) are satisfied as the final composite state has no any outgoing transition and the algorithm maps the final state and all incoming transitions to a corresponding preceding interface and the flows from the interface to the final UIC.

—Condition (7) is held as the first-level and second-level loops iteratively finding and mapping all states and transitions defined in an input ACP-i model to the corresponding interfaces and UICs, and then assigning a navigational flow between an interface and a target UIC which is decided from the transition of one state to another state. The iteration begins at the initial composite state until it reaches the final composite state of the ACP lifecycle.

This completes the proof of THEOREM 6. □

PROOF OF THEOREM 6. We can prove the theorem by induction over the condition of well-formed UIF model (Definition 6.) and the definition of UIF-role model (Definition 6.), and THEOREM 6.. Condition (1) of UIF-role model is naturally satisfied based on THEOREM 6.. As Condition (2) restricts the UIF-role model to derive from the UIF-base model for a particular role, it does not affect to any well-formedness condition. Condition (3) conforms to Condition (7) in Definition 6.
Condition (4) conforms to Conditions (3) and (4) in Definition 6. Lastly, Condition (5) conforms to Conditions (5) and (6) in Definition 6. Since all conditions of UIF-role model conform to the conditions of well-formedness, the theorem holds.

\[\square\]
REFERENCES

Artifact-centric service interoperation (ACSI), 2011, http://acsi-project.eu/


Fons, J., Pelechano, V., Albert, M., Pastor, O.: Development of Web Applications from Web Enhanced Conceptual Schemas. In Workshop on Conceptual Modeling and the Web, ER'03, LNCS 2813


JBoss community, Drools 5.4.0 final, 2012, http://www.jboss.org/drools/


Keller, G., Nu’ttgens, M., Scheer, A.W.: Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK), Veroﬀentlichungen des Instituts fu¨r Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbru¨cken, 1992

191


Liu, C, Li, Q, and, Zhao, X: Challenges and opportunities in collaborative business process management, 2008, Information System Frontiers, May 21


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


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


Transactions on Software Engineering and Methodology, 2009, vol. 19, no. 1, article 2

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


RuleML Initiative, RuleML 1.1, 2011, http://ruleml.org/1.0/


Torres, V., Pelechano, V., Pastor, O.: Building semantic web services based on a model driven web engineering method. In Proceedings of the Advances in


W3C, Web Service Definition Language (WSDL) 1.1, 2001, http://www.w3.org/TR/wSDL
W3C, Cascading Style Sheets (CSS), 2011, http://www.w3.org/TR/CSS/

199