

TRANSFORMATION AND SYNTHESIS OF ARTIFACT-CENTRIC BUSINESS PROCESSES

A THESIS SUBMITTED TO AUCKLAND UNIVERSITY OF TECHNOLOGY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Supervisors

Assoc Prof Jian Yu

Assoc Prof Quan Bai

Dr Sira Yongchareon

May 2020

By

Naga Jyothi Kunchala

School of Engineering, Computer and Mathematical Sciences

Abstract

In today's dynamic business environment, business process modeling (BPM) has become a fundamental tool in many organizations to gain operational benefits and stay competitive with their rivals. Thus, there is always an increasing demand for modeling paradigms to support flexible and reusable process design, facilitate the accommodation of rapidly changing business requirements, and provide insights on process progress to improve process performance and minimize operating costs.

The activity-centric and artifact-centric approaches are the two major modeling paradigms in BPM. While the traditional activity-centric paradigm emphasizes representing activities and their control flows, the recent artifact-centric paradigm provides equal support to control flow and data by incorporating "artifacts" which are key business entities and their lifecycles. In recent years, the artifact-centric paradigm has been extensively studied and there is evidence that this paradigm provides better flexibility and reusability support compared with the traditional activity-centric paradigm.

With the prevalence of the artifact-centric paradigm, the transformation of traditional activity-centric process models into artifact-centric process models has emerged as an important research challenge. Most of the existing approaches for this purpose have a restricted view on the artifacts and their interaction (or synchronization) dependencies, and only support a semi-automatic transformation. Furthermore, these approaches focus on transforming standalone activity-centric process models that specify activities performed in one single organization, while ignoring the inter-organizational business

process (IOBP) models that represent activities and data (artifacts) distributed among various organizations. In addition, the transformation of artifact-centric process models into activity-centric process models has also received considerable attention, as the activity-centric process models provide better process view compared with the artifact-centric process models that contain unorganized sets of business rules. The existing approaches to achieve this objective are also limited in terms of using artifact lifecycles rather than the artifact-centric process models to construct the activity-centric process models and only supporting a semi-automatic transformation.

Therefore, automated approaches are required to improve the scope and efficiency of the proposed transformations. In this regard, this thesis aims to address the following research questions: (1) *How to efficiently transform the activity-centric process models into the artifact-centric process models?*; (2) *How to merge the collaborating processes of activity-centric IOBP (inter-organizational business process) models?*; and (3) *How to construct the activity-centric process models from the artifact-centric process models.*

First, an efficient tree-based approach is proposed to transform activity-centric process models into artifact-centric process models. The proposed transformation mainly aims to synthesize or generate synchronized lifecycles of artifacts from the activity-centric process models. The proposed approach provides a set of algorithms that initially extracts the *hierarchical tree* representation of the activity-centric process model, then generates a lifecycle for every artifact, and also synchronizes the generated lifecycles. The proposed approach is demonstrated using a case study and also implemented and evaluated using a process model collection from the BPM Academic Initiative (BPMAI).

Second, a process interaction-based approach is proposed to merge the collaborating processes of activity-centric inter-organizational business process (IOBP) models that have artifact annotations in order to synthesize the lifecycles of artifacts from the resulting integrated process models. The proposed approach comprises a set of algorithms

that merge two or more collaborating processes based on their interaction patterns. Specifically, the approach first identifies the type of interaction, such as *synchronous* or *asynchronous*, between the collaborating processes and then merge them according to the rules formulated for each type of process interaction. The proposed approach is demonstrated using a case study and also implemented and evaluated using a model collection from the BPMAI.

Last, a reverse transformation approach is proposed for constructing activity-centric process models from artifact-centric process models, and for checking the consistency between the constructed model and the base model. A trace-based method is used to analyze the model consistency, where the execution traces of the base model is analyzed over the execution traces of the constructed model. The proposed approach is demonstrated using a case study and also implemented and evaluated using two motivating process models.

Contents

Abstract	2
Attestation of Authorship	11
Publications	12
Acknowledgements	13
Dedication	14
1 Introduction	15
1.1 Activity-Centric and Artifact-Centric Modeling	18
1.1.1 Activity-Centric Business Process Modeling	18
1.1.2 Artifact-Centric Business Process Modeling	19
1.2 Research Questions	20
1.3 Research Objectives	22
1.3.1 Synthesizing Artifact Lifecycles from Activity-Centric Process Models	22
1.3.2 Merging Collaborating Processes of Inter-Organizational Business Process (IOBP) Models	24
1.3.3 Constructing Activity-Centric Process Models from Artifact-Centric Process Models	26
1.4 Thesis Contributions	27
1.4.1 An Approach to Synthesize Artifact Lifecycles from Activity-Centric Process Models	27
1.4.2 An Approach to Merge Collaborating Processes of Activity-Centric IOBP Models	28
1.4.3 An Approach to Construct Activity-Centric Process Models from Artifact-Centric Process Models	29
1.5 Thesis Outline	29
2 Literature Review	31
2.1 Activity-centric business process modeling	31
2.1.1 Process Modeling	32
2.1.2 Process Realization	35

2.1.3	Inter-Organizational Business Process Modeling	39
2.2	Artifact-centric business process modeling	43
2.2.1	Process Design Methodology	46
2.2.2	Process Discovery and Construction	48
2.2.3	Process Specification and Verification	50
2.2.4	Inter-Organizational Business Process Modeling	55
2.2.5	Process Realization	57
2.3	Process Model Transformation	61
2.3.1	Process Tree Generation	61
2.3.2	Artifact Lifecycle Synthesis	63
2.4	Process Model Merging	66
2.5	Process Model Construction	70
2.6	Summary	73
3	Synthesizing Artifact Lifecycles from Activity-Centric Process Models	74
3.1	Motivating Example	75
3.2	Problem Statement and Definitions	77
3.3	The Synthesis Approach	80
3.4	Algorithms	82
3.4.1	Building a Process Tree	82
3.4.2	Generating Artifact Lifecycles	86
3.4.3	Refining and Synchronizing Artifact Lifecycles	90
3.5	Evaluation	95
3.5.1	Case Study	95
3.5.2	Implementation	98
3.5.3	Experimental Discussion	98
3.5.4	Performance Analysis	101
3.6	Discussion and Related Work	103
3.7	Summary	104
4	Merging the Collaborating Processes of Activity-Centric Inter- Organizational Business Process Models	105
4.1	Motivating Example	106
4.2	Problem Statement and Definitions	108
4.3	The Merge Approach	111
4.3.1	Process Interaction Patterns	111
4.3.2	Merge Overview	114
4.3.3	Types of Merge	115
4.4	Algorithms	122
4.4.1	Parallel Merge: Merging Non-Synchronous Nodes	124
4.4.2	Interactive Merge: Merging Synchronous and Asynchronous Nodes	128
4.5	Evaluation	135
4.5.1	Case Study	135

4.5.2	Implementation	140
4.5.3	Experimental Discussion	141
4.5.4	Performance Analysis	144
4.6	Discussion and Related Work	145
4.7	Summary	146
5	Constructing Activity-Centric Process Models from Artifact-Centric Process Models	147
5.1	Motivating Example	148
5.2	Problem Statement and Definitions	151
5.3	Approach Overview	153
5.4	Algorithms	154
5.4.1	Model Construction	154
5.4.2	Extract Model Traces	161
5.4.3	Trace-based Analysis	164
5.5	Evaluation	165
5.5.1	Case Study	165
5.5.2	Implementation	172
5.5.3	Experimental Discussion and Analysis	173
5.6	Discussion and Related Work	174
5.7	Summary	175
6	Conclusion and Future Directions	177
6.1	Thesis Contributions	177
6.1.1	The Synthesis Approach	178
6.1.2	The Merge Approach	178
6.1.3	The Construction Approach	179
6.2	Limitations and Possible Improvements	180
6.3	Future Research Directions	181
6.3.1	Internet-of-Things (IoT)	182
6.3.2	Block Chain	183
	References	186
	Appendices	201

List of Tables

3.1	Dataset	100
4.1	Summary of symbols used	114
4.2	Data Set	142
5.1	Artifact-centric Process Model (ACP Model)	149
5.2	ACP Log	151
5.3	ACP Model of Purchasing Process	168
5.4	Execution Log of Purchasing Process Model	170

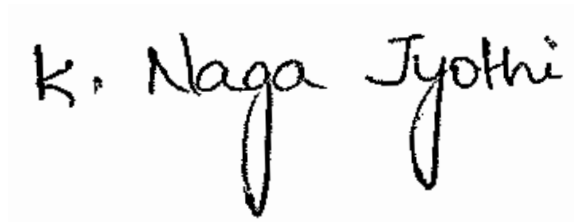
List of Figures

2.1	A Business Process Diagram represented in BPMN (White, 2004) . . .	37
2.2	Process-view approach (Jiang, Shao, Gao, Qiu & Li, 2010)	42
2.3	Lifecycle of Guest Check (Nigam & Caswell, 2003)	44
2.4	Four Dimensional Framework (Bhattacharya, Hull & Su, 2009)	45
2.5	Design Methodology from (Bhattacharya et al., 2009)	47
2.6	Artifact-centric View Framework (Yongchareon, Yu, Zhao et al., 2015)	57
2.7	Artifact-centric Process Realization Framework (Ngamakeur, Yongchareon & Liu, 2012)	60
3.1	Customer Order Process	76
3.2	Synthesis Approach	81
3.3	Customer Order Process (updated AAPM)	83
3.4	Process Tree of AAPM	85
3.5	Lifecycle of Product artifact	88
3.6	Generated Lifecycle of Order artifact	92
3.7	Refined Lifecycle of Order artifact	93
3.8	Statechart representation of Product artifact lifecycle	93
3.9	Synchronized artifact lifecycles of customer order process	94
3.10	Histogram of Sorted BPMN Model Collection	96
3.11	Excerpt of the Recruitment Process Model	97
3.12	Visualization of Process Tree of Customer Order Process	99
3.13	Visualization of Synchronized Artifact Lifecycle of Customer Order Process	99
3.14	Execution times of Synthesis algorithms	102
4.1	Buyer-Seller Inter-organizational Business Process (IOBP)	107
4.2	Process Interaction Patterns	112
4.3	Types of Merge	116
4.4	Parallel Merge	117
4.5	Synchronous Non-Split Merge	118
4.6	Synchronous Non-Split Merge	119
4.7	Synchronous Split Merge	120
4.8	Asynchronous Non-Split Merge	121
4.9	Asynchronous Split Merge	122
4.10	Parallel Merge	126

4.11	Parallel Merge	127
4.12	Synchronous Merge	131
4.13	Synchronous Merge	132
4.14	Asynchronous Split Merge	134
4.15	Histogram of Sorted BPMN Choreography Process models	136
4.16	Excerpt of AirTravel Process Model	137
4.17	Buyer-Seller IOBP and its Integrated Process Model	140
4.18	Integrated Process Model of Buyer-Seller IOBP	141
4.19	Execution times of Merge algorithms	144
5.1	Transformation Approach	154
5.2	The constructed Activity-Centric Process Model	157
5.3	Execution Traces of Activity-Centric Process Model	163
5.4	Process Traces and Analysis	165
5.5	Purchasing process model (Yongchareon et al., 2015)	167
5.6	An Abstracted Activity-Centric Model of Purchasing Process	170
5.7	The constructed Activity-Centric Model of Purchasing Process	171
5.8	XML Format of ACP Model	173
A.1	Complete Recruitment process model	203
A.2	Process tree fragment of Recruitment process model	204
A.3	Lifecycle of Position artifact	204
A.4	XML Specification of BPMN Process Model	205
A.5	AirTravel Process Model	206
A.6	The constructed Activity-Centric Model of Purchasing Process	207

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.

A handwritten signature in black ink on a light gray background. The signature reads "K. Naga Jyothi" in a cursive, flowing script. The "K" is small and followed by a period. "Naga" and "Jyothi" are written in a larger, more expressive hand.

Signature of candidate

Publications

- Kunchala, J., Yu, J., Yongchareon, S., Wang, G. (2020, February). Trace-Based Approach for Consistent Construction of Activity-Centric Process Models from Data-Centric Process Models. In Australasian Database Conference (pp. 42-54). Springer.
- Kunchala, J., Yu, J., Yongchareon, S., Liu, C. (2019). An approach to merge collaborating processes of an inter-organizational business process for artifact lifecycle synthesis. *Computing Journal* (IF 2.063, CORE A), 1-26.
- Kunchala, J., Yu, J., Yongchareon, S., Han, Y. (2017, January). Towards merging collaborating processes for artifact lifecycle synthesis. In *Proceedings of the Australasian Computer Science Week Multiconference* (pp. 1-8).
- Kunchala, J., Yu, J., Sheng, Q. Z., Han, Y., Yongchareon, S. (2015, September). Synthesis of artifact lifecycles from activity-centric process models. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference* (pp. 29-37). IEEE.
- Kunchala, J., Yu, J., Yongchareon, S. (2014, October). A survey on approaches to modeling artifact-centric business processes. In *International Conference on Web Information Systems Engineering* (pp. 117-132). Springer.

Acknowledgements

With immense pleasure and respect, I would like to thank everyone who contributed to the successful completion of my doctoral study. I would have never accomplished this thesis without the love and support of many people.

First and the foremost, I am deeply grateful to my primary supervisor Dr Jian Yu. I would like to express sincere gratitude to Dr Jian Yu for being with me and helping me in every step of my PhD journey. From Dr Jian Yu, I received insightful suggestions, hearty support, and invaluable encouragement during the course of this study. The research methods and wisdom shared during the meetings were the major contributions toward the successful completion of this thesis. I would also like to thank for providing the teaching opportunity here at the School of Engineering, Computer, and Mathematical Sciences (SECMS), which helped me improve my communication and teaching skills, also to socialize and interact with students and other staff members. I would also like to thank Dr Jian Yu for helping me to find a PhD Fees Scholarship and one-off stipend, which helped me to concentrate on research without a financial distress.

Next, I would like to thank my second supervisor Dr Quan Bai for his valuable guidance and motivation. Dr Quan Bai has always provided insightful suggestions starting from the time of my successful admission to the completion.

I would also like to express my sincere gratitude to my additional supervisor Dr Sira Yongchareon. With immense knowledge in the Artifact-centric Business Process Modeling (ACBPM) research area, Dr Sira has guided me through and offered valuable suggestions in almost every stage of my PhD study.

I am also thankful to numerous staff at AUT, including Saide and Karishma. I would thank for their friendship, advice, moral and social support, and for guiding and sharing their research knowledge and experience. The SCCRL research lab has been my second home during the last few years and I found fabulous people especially Monjur and Abid whose incredible hard work, determination, and perseverance helped me to keep motivated. The SERL research lab has also been an amazing research group to familiarize with and share ideas.

Finally, I am grateful and indebted to my family especially to my parents Mrs Papamma Thanniru and Mr Nageshwara Rao Thanniru, and my beloved husband Mr Hari Babu Kunchala and children Sharath Chandra Kunchala and Nikhil Chandra Kunchala for their extreme care, love and support during this long journey.

Dedication

With great love and respect, I dedicate this thesis to my beloved husband *Mr Hari Babu Kunchala* and children *Sharath Chandra Kunchala* and *Nikhil Chandra Kunchala*.

Chapter 1

Introduction

Business processes have become a major driving force enabling organizations to achieve defined objectives and to stay competitive in the ever-changing business environment. Business process modeling (BPM) is one of the foundational characteristics of business process management (Van der Aalst, 2013; Jonnavithula, Antunes, Cranefield & Pino, 2015), aiding organizations in designing and managing their business processes. BPM can increase productivity and decrease costs by streamlining business operations. Process models are a means to achieve these objectives by communicating business information to a wide range of people – from analysts who sketch the initial drafts of the processes and system developers responsible for implementing these processes, to the customers who use the final product (R. Liu, Bhattacharya & Wu, 2007).

In today's competitive marketplace, BPM is a prominent tool helping organizations refine their business operations to gain as well as provide value to their customers. BPM intends to separate process logic from application logic in order to improve process efficiency by enabling the automation of business processes (Orlowska, 1997). Recently, BPM has greatly accelerated through two technological evolutions, including Service-Oriented Architecture (SOA) and Internet of Things (IoT). SOA provides a set of design principles and methodologies (MacKenzie et al., 2006) to aid organizations in

constructing loosely-coupled and inter-operable business processes using web services. SOA also helps organizations to update their business processes in a more rapid and cost-effective manner; and to seamlessly obtain and integrate new business partners and customers. IoT has the potential to connect physical and digital objects, when they are equipped with smart devices, such as sensors and actuators (Goyal & Jain, 2011; Friedow, Völker & Hewelt, 2018), to serve a specific purpose. Many organizations have already implemented IoT technology for monitoring their business processes – including manufacturing, retailing and transportation, in order to gain some of the benefits of IoT, such as reducing costs and improving process visibility and efficiency (Pundir, Jagannath & Ganapathy, 2019).

The integration of such technologies can bring new challenges to organizations, mainly in the modeling and management of their business processes. Thus, modeling paradigms are required to enable flexible and reusable process designs to address emerging business requirements. Currently, the *activity-centric* and *artifact-centric* approaches are the two major paradigms in BPM that can address these requirements. Both paradigms aim to represent two important aspects of a business process including: *control flow* and *data*. The *control flow* represents the partial order between activities, events, and gateways, whereas the *data* (or data aspects) represent the key business objects (or entities), their states, attributes including their interrelations (Meyer, Smirnov & Weske, 2011). A data object represents the information (such as physical/electronic documents) that is created, manipulated and used by a business process. The activity-centric approach is a traditional approach that places more emphasis on the control flow aspect and treats data as simple black boxes that act as input and output to these activities. On the other hand, the artifact-centric approach provides equal support to both the control flow and data aspects. *Business artifacts* (or artifacts) and their interacting *lifecycles* are the building blocks of artifact-centric process models (Nigam & Caswell, 2003). An *artifact* is a business-related object that is created, evolved and archived as

it passes through a business process (Cohn & Hull, 2009). The *lifecycle* captures all possible stages that an artifact navigates from its creation to archiving.

The activity-centric and artifact-centric approaches support modeling both the standalone processes that are used within an organization and those that span several organizations, such as inter-organizational business processes (IOBPs). The traditional activity-centric approach has been extensively used to support the dynamic cooperation of IOBPs in a Service-Oriented setting (2006; 2010), and is also used for the modeling, execution and monitoring of IoT-aware business processes (Petrasch & Hentschke, 2015; Schöning, Ackermann, Jablonski & Ermer, 2018). In recent years, the artifact-centric approach has also been identified as one promising paradigm that lends itself well to Service-Oriented collaboration due to its modular nature (R. Liu, Wu & Kumaran, 2010; Yongchareon et al., 2015). This approach has also been extended to monitor the compliant execution of IOT-enabled IOBPs. (Meroni, Di Ciccio, Mendling et al., 2017; Meroni, Baresi, Montali & Plebani, 2018).

In recent years, transformation between the activity-centric and artifact-centric paradigms has been proposed as an important research aspect to understand the relationship between the two paradigms (S. Kumaran, Liu & Wu, 2008; R. Liu et al., 2010). However, most of the existing research has focused on unidirectional transformation neglecting that both paradigms are compatible to each other such that one can be transformed into the other and vice versa (Meyer & Weske, 2013). In addition, the existing approaches for the activity-centric process transformation into the artifact-centric process neglect the interaction dependencies between the artifacts (Ryndina, Küster & Gall, 2006; Cabanillas, Resinas, Ruiz-Cortés & Awad, 2011). Thus, this thesis aims to bridge these gaps between the activity- and artifact-centric paradigms by proposing approaches that not only fill the above specified gaps but also facilitate the transformation between these two mainstream modeling paradigms. The remaining chapter is organized into five sections. Section 1.1 reviews the two major paradigms for business process modeling.

Section 1.2 outlines the research questions. Section 1.3 summarizes the objectives of the proposed research. Section 1.4 summarizes the research contributions of this thesis. Section 1.5 provides an outline of this thesis.

1.1 Activity-Centric and Artifact-Centric Modeling

In this section, the benefits and drawbacks of each modeling paradigm are discussed as well as the domains where the activity-centric and the artifact-centric paradigms can be useful to gain major advantage.

1.1.1 Activity-Centric Business Process Modeling

The traditional activity-centric approach is more beneficial in the classical business domain, where the process flow is based on a predefined order of activities, such as *accounting*, *insurance handling* and *municipal procedures* (Meyer & Weske, 2013). An activity-centric approach is imperative in nature, which means it is characterised by a straightforward execution of predefined execution paths, which may result in considerable process efficiency for static and standardised business processes (Goedertier, Vanthienen & Caron, 2015). In addition, most of the activity-centric modeling languages, such as UML and BPMN ¹, provide clear visual representations of business processes.

Although this approach is efficient in modeling activity-driven processes, it becomes difficult to understand these processes when they grow in size and complexity (R. Liu et al., 2010). This drawback is mainly due to their tightly-coupled process structures, which also make it difficult to extend and reuse these processes naturally (S. Kumaran et al., 2008). In addition, due to the the lack of a holistic view of the control flow and

¹Business Process Model and Notation, <http://www.bpmn.org/>

data aspects, business people using this approach often focus on the execution of activities rather than the data emerging from them, thus hindering operational innovations (Bhattacharya, Gerede, Hull, Liu & Su, 2007). The absence of data in activity-centric process models can also bring challenges in designing effective user-friendly human interfaces (Yongchareon et al., 2018) that are responsible for driving overall process execution in process management systems. Therefore, the activity-centric approach is considered less efficient for modeling processes such as those in the *healthcare* domain (Künzle & Reichert, 2011), for which both the control flow and data are crucial.

1.1.2 Artifact-Centric Business Process Modeling

In recent years, the artifact-centric approach (Nigam & Caswell, 2003) has gained increased attention for elevating data to the same level as the control flow. Over a decade, this approach has been studied by several academic and industrial researchers whose efforts lead to the emergence of various *design methodologies* (Bhattacharya et al., 2009; Kovář, Beránek & Feuerlicht, 2017), *frameworks* (Bhattacharya et al., 2009; Cohn & Hull, 2009; Kucukoguz & Su, 2010; Limonad, Boaz, Hull, Vaculin & Heath, 2012; Ngamakeur et al., 2012; Solomakhin, Montali, Tessaris & De Masellis, 2013; Yongchareon et al., 2015), *meta-models* (G. Liu et al., 2009; Abiteboul, Bourhis, Galland & Marinoiu, 2009; Kucukoguz & Su, 2010; Hull et al., 2010; Lohmann & Wolf, 2010), *modeling tools* (Strosnider, Nandi, Kumaran, Ghosh & Arsnajani, 2008; Cohn, Dhoolia, Heath, Pinel & Vergo, 2008; Heath et al., 2013) and *user-centric approaches* (Sukaviriya, Mani & Sinha, 2009; Yongchareon, Liu, Zhao & Xu, 2010; Yongchareon et al., 2018) in support to this new modeling paradigm.

The application of the artifact-centric approach in various domains, including *insurance* and *global finance* (S. Kumaran et al., 2008; Chao et al., 2009; Cohn & Hull, 2009) has revealed that this approach provides rich and natural communication of

business operations and processes among diverse stakeholders of an organization, and also lends itself well to process componentization and extension (Cohn & Hull, 2009; S. Kumaran et al., 2008) compared with the traditional activity-centric approach. The existing literature has also demonstrated that the artifact-centric approach offers higher flexibility and reusability support than the traditional activity-centric approach (R. Liu et al., 2010; Yongchareon et al., 2015). The other key advantages of this approach are in reducing process complexity and business-level articulation, providing an ability to analyze and reconcile changes from multiple business perspectives (eg. process and organization) and also enabling the rapid creation of IT solutions (Bhattacharya, Caswell, Kumaran, Nigam & Wu, 2007; R. Liu et al., 2010). As discussed above, the existing literature of these two modeling paradigms acknowledges that the activity-centric paradigm is useful in a static business environment where the process efficiency is of major concern, as most of the existing process specification and execution standards support this traditional paradigm (Meyer & Weske, 2013; de Leoni, Maggi & van der Aalst, 2015; Caron & Vanthienen, 2016). While in a dynamic environment, the artifact-centric approach can be adopted to build flexible and reusable process structures that can quickly adapt to changing business requirements and also bring cost benefits.

1.2 Research Questions

In recent years, the artifact-centric approach has gained significant interest due to its aforementioned benefits. Process flexibility is the remarkable benefit of this paradigm with loosely-coupled process structures that allow changes at both design-time and run-time. As discussed in the previous section, various aspects of both the activity-centric and artifact-centric paradigms have been extensively studied, showing a promising trend in their state of the art. However, further research is suggested to bridge the gap between these two modeling paradigms to communicate their importance, value and

usefulness to a wider business and research community (Hull, 2008; S. Kumaran et al., 2008; Meyer & Weske, 2013; Yongchareon et al., 2015; Hull, 2017; Mendling et al., 2018). Therefore, this thesis focuses on addressing three key research questions to provide increased support for organizations that intend a transformation between their activity-centric and artifact-centric process models.

RQ 1. How to efficiently transform the traditional activity-centric process models into the artifact-centric process models?

The artifact-centric approach describes a business process in terms of artifacts that represent key business entities and their interacting lifecycles. With the increased significance of this approach, the research community has focused on the transformation of traditional activity-centric process models into artifact-centric process models. The proposed transformation mainly aims to generate or synthesize the synchronized artifact lifecycles from the activity-centric process models. The existing approaches (S. Kumaran et al., 2008; Eshuis & Van Gorp, 2016) for reaching this objective are limited in terms of having a restricted view on the artifacts and their synchronization dependencies or only supporting semi-automatic transformation (Ryndina et al., 2006). Therefore, an automated approach must be proposed to address these limitations and facilitate the defined transformation.

Artifact-centric process models are particularly useful to ensure the correct execution of business processes that span several organizations (Küster, Ryndina & Gall, 2007), such as IOBPs that share artifacts to achieve a common business goal. The existing transformation approaches can be utilized to synthesize the artifact-centric process model of an IOBP, if the collaborating processes of IOBP are merged into a single integrated process. Thus, this requirement leads to the following research question.

RQ 2. How to merge the collaborating processes of activity-centric inter-organizational business process models?

The transformation of activity-centric IOBPs is not proposed, mainly due to the

challenges in capturing artifacts and preserving their state and interaction dependencies. Therefore, an approach must be proposed to address the above research question.

The value of an artifact-centric approach is realized in the flexible representations that allow business people to easily update and manage their business processes (Nigam & Caswell, 2003). However, the declarative rule-based modeling of this approach that offers higher flexibility often results in process models that are less comprehensible than the activity-centric process models, due to their large and unstructured sets of business rules (Haisjackl & Zugal, 2014; Caron & Vanthienen, 2016). Thus, this requirement leads to the following research question.

RQ 3. How to construct the activity-centric process models from the artifact-centric process models?

There are a few approaches that use object (artifact) lifecycles to generate activity-centric process models (Küster et al., 2007; Redding, Dumas, Hofstede & Iordachescu, 2008; Meyer & Weske, 2013) rather than the artifact-centric process models that contain business rules. Therefore, an approach that uses artifact-centric process models to construct activity-centric process models must be proposed.

1.3 Research Objectives

This section elaborates on the objectives of proposed research, including the motivation to study each of the research questions.

1.3.1 Synthesizing Artifact Lifecycles from Activity-Centric Process Models

The artifact-centric approach provides a compelling way to model and manage business processes. This approach is now incorporated into organizations such as IBM (Chao et

al., 2009) and Unicorn (Kovář et al., 2017), enhancing the way that these businesses model and deploy their business operations and processes. The holistic view that this approach takes avoids the notorious discrepancy between the process modeling and data modeling of the traditional activity-centric approach which considers these two aspects separately (Bagheri Hariri, Calvanese, De Giacomo, Deutsch & Montali, 2013).

In recent years, research on the transformation of the traditional activity-centric process models into artifact-centric process models has received considerable attention. The proposed transformation aims to synthesize (or generate) the synchronized lifecycles of artifacts from activity-centric process models. A synchronized artifact lifecycle is a commonly used visual representation of an artifact-centric process model that represents process behaviour in a set of states that an artifact assumes throughout a business process and its dependencies with the states of other artifacts. Some of the benefits that stem from such a transformation include reduced process complexity and reusable process design (R. Liu et al., 2010); and improved flexibility and efficient user interface development (Yongchareon et al., 2018). Therefore, this thesis focuses on proposing an approach to facilitate this transformation.

In the past several years, some approaches have been proposed to achieve this objective (Ryndina et al., 2006; S. Kumaran et al., 2008; Cabanillas et al., 2011; Eshuis & Van Gorp, 2012; Meyer & Weske, 2013). However, the existing approaches have some limitations in either not supporting automatic transformation or having a restricted view on the artifacts and their synchronization (or interaction) dependencies. For example, the approaches presented in (S. Kumaran et al., 2008; Eshuis & Van Gorp, 2012) restrict the flow of artifacts among activities, and the approaches presented in (Ryndina et al., 2006; Cabanillas et al., 2011) do not represent the synchronization dependencies of artifacts (or objects) in the generated lifecycles, which degrades the understanding of process behaviour. Similarly, the approach presented in (Meyer & Weske, 2013) does not support automatic model transformation and this approach is not

thoroughly evaluated.

Clearly, for any organization that intends to transform its traditional activity-centric process models into artifact-centric process models, an automated approach that does not pose such restrictions is always desirable. Therefore, this thesis aims to present an automated approach that synthesizes the lifecycles of artifacts from activity-centric process models that contain artifact data flows. These lifecycles are also synchronized to show interaction dependencies between artifacts. The synthesized artifact lifecycles are useful to ensure the correct execution of business processes by tracking their progress towards the defined business objectives. The existing works use a case study and a set of process models to evaluate their approaches. This study also follows similar evaluation criteria, where the feasibility of the proposed approach will be demonstrated using a case study and the applicability and efficiency is evaluated by utilizing a process model collection from the BPM Academic Initiative (Kunze, Berger, Weske, Lohmann & Moser, 2012).

1.3.2 Merging Collaborating Processes of Inter-Organizational Business Process (IOBP) Models

Due to growing business requirements, organizations tend to integrate their business processes to better serve their customers' needs and achieve competitive benefits. In this context, an Inter-Organizational Business Process (IOBP) model enables two or more organizations to represent their business activities to achieve a common business objective. IOBP provides a collaborative environment, where participating organizations can coordinate through mutual contracts agreed upon to fulfill their objectives while maintaining their privacy and autonomy.

Both, the activity-centric and artifact-centric approaches provide support to the modeling and management of these IOBPs. The activity-centric IOBP can be represented

as a set of collaborating processes, where they exchange some information (or data) to perform a business activity. The artifact-centric IOBP is represented as a set of lifecycles of artifacts and their synchronization dependencies. Recent studies have explored how the artifact-centric modeling can facilitate collaboration between organizations (Hull, Narendra & Nigam, 2009; Yongchareon et al., 2015). When organizations involved in such collaborative environments intend to shift their IOBPs from the traditional activity-centric approach to the artifact-centric approach, an automated approach is needed to facilitate such a transformation. However, unlike standalone activity-centric process models, the transformation of activity-centric IOBPs is challenging due to the data (artifacts and states) shared between the collaborating processes to conduct a business activity.

In recent years, process merging has become increasingly important for organizations to redesign their business processes in order to gain operational improvements and cost savings. There exist many research contributions for merging business process models (S. Sun, Kumar & Yen, 2006; Gottschalk, van der Aalst & Jansen-Vullers, 2008; La Rosa, Dumas, Uba & Dijkman, 2013; Schunselaar, Leopold, Verbeek, van der Aalst & Reijers, 2014; Zemni, Mammar & Hadj-Alouane, 2016; Huang, Li, Liang, Xue & Wang, 2018). However, these existing works focus on merging standalone process models by considering their common process elements, irrespective of the data aspect. Additionally, only a few of them support the automatic merging of process models. For example, the merge approaches presented in (S. Sun et al., 2006; La Rosa et al., 2013) can only support the semi-automatic merging of standalone process models, and those presented in (Gottschalk et al., 2008; Schunselaar et al., 2014; Zemni et al., 2016; Huang et al., 2018) use similarity matching between the process models, where the matching nodes are first mapped into the merged model and then the remaining nodes are added based on their execution dependencies with the previously merged fragments.

Based on the notion of process merging, a novel approach is proposed that merges

the collaborating processes of IOBP models containing artifacts. The proposed merge approach combines processes based on the *synchronous* and *asynchronous* process interaction patterns that define the type of communication between the collaborating processes. This approach is also validated by following evaluation criteria from the existing literature, where the feasibility and applicability of the proposed approach is evaluated using a case study and a set of IOBP models from varying business domains. The structural and behavioural aspects of the resulting merged (or integrated) process models are also proved using theorems.

1.3.3 Constructing Activity-Centric Process Models from Artifact-Centric Process Models

Nowadays, organizations increasingly rely on web service technologies to implement, manage and automate their business processes. The automation starts from process models that represent a series of activities to achieve a specific business goal. Automation efficiency is reduced if these models are specified with many alternative execution paths, which increase complexity and degrade the process efficiency. The activity-centric approach is imperative in nature, which requires a process model to explicitly specify every alternative execution sequence during design-time. On the other hand, the artifact-centric approach follows a declarative style to describe process behaviour, where the execution sequence of activities is governed by business rules. Although the artifact-centric approach provides higher design-time flexibility, most of the existing process models are designed using the activity-centric approach. This is mainly due to the existence of numerous modeling notations, tools and web technologies that provide higher intuitiveness and run-time efficiency for activity-centric process models. Therefore, a reverse transformation approach is proposed, where artifact-centric process models are used to construct activity-centric process models.

A few approaches exist for the defined transformation (Küster et al., 2007; Meyer & Weske, 2013; Prescher, Di Ciccio & Mendling, 2014; De Giacomo, Dumas, Maggi & Montali, 2015). However, these approaches have different objectives and limitations. For example, the approaches presented in (Küster et al., 2007; Meyer & Weske, 2013) aim to generate activity-centric process models from unsynchronized object lifecycles. The approach presented in (Prescher et al., 2014) produces duplicate tasks that lead to an increased number of execution alternatives for the resulting process models. Similarly, the approach presented in (De Giacomo et al., 2015) does not consider the parallel states of objects. Therefore, an approach is proposed here that supports the automatic transformation and addresses the above mentioned limitations. Following the evaluation criteria from the existing literature, the feasibility of the proposed approach is demonstrated using a case study and the applicability is evaluated using prototype implementation.

1.4 Thesis Contributions

This thesis aims to study the activity-centric and artifact-centric approaches to business process modeling and address the three research questions outlined in Section 1.2. In this section, the research contributions of this thesis are briefly summarized.

1.4.1 An Approach to Synthesize Artifact Lifecycles from Activity-Centric Process Models

This thesis proposes an automated approach to transform activity-centric process models into artifact-centric process models. The proposed approach aims to synthesize the artifact lifecycles and their synchronization dependencies from the activity-centric process models that contain artifact data flows. To achieve this objective, the approach

first generates a tree representation of the activity-centric process model and then constructs the lifecycle for each artifact inherent in the process model. Moreover, the synchronization dependencies between the artifacts are also identified and modelled in the synthesized artifact lifecycles.

The proposed synthesis approach consists of algorithms, which are implemented to support the automatic transformation of activity-centric process models. A process model collection from the repository of BPM Academic Initiative has been utilized to evaluate the feasibility and applicability of the proposed approach. Compared to the existing approaches, this approach does not restrict the flow of artifacts, considers synchronization dependencies between the artifacts and is useful to automatically synthesize artifact-centric counterparts such as the synchronized artifact lifecycles from activity-centric process models.

1.4.2 An Approach to Merge Collaborating Processes of Activity-Centric IOBP Models

This thesis proposes an automated approach to merge the collaborating processes of activity-centric IOBP models that contain artifact data flows. The proposed approach is based on the synchronous and asynchronous interactions between the collaborating processes. Specifically, rules are proposed that define ways to combine both the interacting and non-interacting nodes of collaborating processes. Algorithms are also presented based on these rules to support automatic merging. The algorithms are implemented and evaluated to show their feasibility and applicability using a subset of IOBPs from different business domains.

The proposed merge approach compliments the synthesis approach by extending support to IOBP models. Compared to the existing process merging approaches, the proposed approach does not demand the collaborating processes of IOBP models with

common process fragments and also considers their data aspects.

1.4.3 An Approach to Construct Activity-Centric Process Models from Artifact-Centric Process Models

An automated approach is proposed to construct activity-centric process models from artifact-centric process models. The proposed approach consists of algorithms to support the construction of activity-centric process models and to determine their consistency with the base models. A trace-based method is a part of these algorithms, which extracts the execution traces of constructed models and analyzes these traces over the execution traces of artifact-centric process models to check the consistency between the base and constructed process models. The feasibility and applicability of the proposed approach is demonstrated using a process model from the supply chain domain.

The proposed algorithms are also implemented and evaluated using two motivating business scenarios. Compared to the existing approaches, the proposed approach supports automatic construction and it does not result in duplicate tasks, thus avoiding the redundant execution alternatives in the constructed process models.

It is worth mentioning that the scope of the above discussed transformation approaches is limited to structured (and semi-structured) process models that contain commonly used modeling constructs including artifacts and states. Thus, one may need to extend these approaches to transform process models that contain other modeling elements such as nested loops or subroutines.

1.5 Thesis Outline

This thesis is organized into six chapters. In this section, the key contents of each of the remaining chapters are briefly summarized.

Chapter 2 presents the necessary background to the activity-centric modeling paradigm and reviews the overwhelming evolution of the artifact-centric approach over the last decade. This chapter also discusses related works regarding the three research questions highlighted and discussed in the previous section.

Chapter 3 presents an automated approach to transform activity-centric process models into artifact-centric process models. In this chapter, the problem statement is first formulated and then each key notion used in resolving the proposed problem is formally defined. Then, the synthesis approach is presented and the three important phases of this approach is discussed in detail. A case study is also presented along with the implementation and evaluation details of the proposed approach. The performance analysis of the proposed approach is also demonstrated.

Chapter 4 presents an automated approach to merge the interacting processes of an activity-centric IOBP model that contains artifact data flows. In this chapter, different types of merge notions and their corresponding rules are proposed. The algorithms that are defined based on the proposed rules are presented and their implementation and evaluation details are also discussed. The performance analysis of the proposed approach is also demonstrated.

Chapter 5 presents an approach for the construction of activity-centric process models from artifact-centric process models. The proposed approach provides algorithms to construct the activity-centric process model and to determine the consistency between the constructed model and the base model. A comprehensive business case is used to demonstrate the feasibility of the proposed approach, which is also implemented and evaluated using two motivating business scenarios.

Chapter 6 summarizes the objectives and contributions of this thesis and concludes the thesis by presenting a discussion on the limitations and future directions of the proposed research.

Chapter 2

Literature Review

In this chapter, a critical review of the literature is presented to provide the context in which the research is undertaken. This chapter also classifies and discusses existing works related to the three research questions outlined in the previous chapter. Section 2.1 presents the necessary background to an activity-centric business process modeling approach. Section 2.2 reviews the state of the art of artifact-centric business process modeling approach. Section 2.3 discusses the existing works related to the activity-centric process model transformation (RQ1). Section 2.4 discusses the existing approaches to the merging of business processes (RQ2). Section 2.5 discusses the existing approaches to activity-centric process model construction (RQ3). Section 2.6 outlines the summary of this chapter.

2.1 Activity-centric business process modeling

The activity-centric approach emphasizes the representation of structured activities carried out to achieve business objectives, where the data involved in conducting these activities is of secondary concern. This traditional approach is considered more appropriate for traditional domains, such as *accounting*, *insurance* and *municipal*

procedures, where the process flow is based on the activities that need to be executed in a specific order (Meyer & Weske, 2013).

A large volume of literature exists in support to the modeling, management and execution of activity-centric business processes. In the following, we briefly elaborate on some of the key contributions regarding this traditional modeling paradigm.

2.1.1 Process Modeling

Several activity-centric modeling notations exist that vary in methods of capturing a business process and their level of expressive power. The most commonly used graphical notations include UML Activity Diagrams (UML AD) (Dumas & Ter Hofstede, 2001), Business Process Model and Notation (BPMN) (White, 2004), Event-driven Process Chain (EPC) (Kindler, 2004; Keller, Scheer & Nüttgens, 1992), and Yet Another Workflow Language (YAWL) (Van Der Aalst & Ter Hofstede, 2005). Each of these modeling notations provides its own set of constructs and semantics to represent business processes.

UML Activity Diagrams (UML AD) (Dumas & Ter Hofstede, 2001) is one of the commonly used modeling notations for representing business processes. A business process model expressed in UML AD can specify activities and their associations, including the business objects that flow among these activities. The modeling constructs of this notation also support parallelism and the event-driven behaviour of complex business processes (Eshuis, 2006).

Event-driven Process Chain (EPC) (Keller et al., 1992) is another modeling language used to represent temporal and logical dependencies between the activities of a business process. EPC provides three types of modeling elements, including *functions* (to capture the business activities), *events* (to specify pre- and post-conditions of functions) and *connectors* (that link functions and events) to construct an activity-centric process model

(Kindler, 2004).

Yet Another Workflow Language (YAWL) (Van Der Aalst & Ter Hofstede, 2005) is a modeling language based on Petri Nets (Murata, 1989) which is proposed to facilitate the modeling of complex workflows. YAWL is extended from workflow nets (a class of Petri Nets) described in (Van der Aalst, 1998; Van Der Aalst, Van Hee & van Hee, 2004). This extension provides modeling support to multiple instances, composite tasks, OR-joins, removal of tokens and directly connected transitions in a complex workflow (Van Der Aalst & Ter Hofstede, 2005).

Business Process Model and Notation (BPMN) (White, 2004) is another modeling notation that gained popularity for specifying business processes in a highly expressive manner (Chinosi & Trombetta, 2012). The primary objective of BPMN is to provide a graphical notation, called Business Process Diagram (BPD), that is easily understandable to business users.

In recent years, BPMN has grown as an OMG Standard (B. P. M. N. OMG, 2006) by presenting intuitive and highly expressive modeling constructs for representing both standalone and inter-organizational business processes (IOBPs). Therefore, this thesis will utilize this notation for representing activity-centric process models. In order to familiarize the reader with BPMN, a brief discussion of the modeling constructs of this notation follows. These modeling elements are mainly categorized into Flow Objects, Data, Connecting Objects, Swimlanes and Artifacts (Zafar et al., 2018; Dijkman, Dumas & Ouyang, 2008).

Flow Objects (Events, Activities and Gateways). These are the key modeling constructs that specify the behaviour of a business process. An *event* (start, intermediate, end) in this category represents an occurrence of some action that may have an impact on the business process. The *start* event acts as a trigger to initiate the process. The *intermediate* event represents the occurrence of an event in between the starting and ending of a business process. The *end* event represents the completion of a process.

Activities constitute units of work performed in a business process and *gateways* are for controlling the flow of these activities.

Data (Data Objects, Data Inputs, Data Outputs and Data Stores). These elements represent the data entities used and produced by the activities in the BPMN process model. While the *data objects* refer to entities, their mapping as input and output to related activities are specified using *data inputs* and *data outputs*. The *data stores* are mechanisms to store *data objects* in order to support the execution of activities.

Connecting Objects (Sequence Flow, Message Flow and Association). These elements are the means to represent flow objects that are connected to form a process flow. While a *sequence flow* represents an order of flow elements, the *message flow* shows the interaction between different process components. An *association* is for representing information flows in a BPMN process model. Usually, associations are for linking data objects to BPMN activities.

Swimlanes (Pools and Lanes). These elements are for grouping primary modeling constructs. *Pools* are used to set the boundaries of a business process, where a pool represents at most one business process. *Lanes* are for representing sub-partitions within a pool and are used to organize and categorize activities. Most commonly, a lane represents an organizational role in a BPMN process model.

Artifacts (Data Object, Group and Annotation). These elements are used to provide additional information about a business process in the form of *data objects*. *Group* provides a means to categorize several process elements together according to certain criteria. *Annotation* is for adding more information to comprehend a process component or the process itself.

As discussed above, each modeling notation has its own constructs and capabilities to represent business processes. The formal languages, such as Petri Nets (Murata, 1989) and Pi calculus (Milner, 1999), also support the modeling and verification of activity-centric business processes.

2.1.2 Process Realization

The Web Service Business Process Execution Language (WS-BPEL or BPEL) (Andrews et al., 2003) and Web Services Choreography Description Language (WS-CDL) (W3C, 2005) are the two commonly known languages that provide a code-based way to specify activity-centric processes executable in the process management systems (Marrella et al., 2015).

Business Process Execution Language (BPEL)

The Business Process Execution Language for Web Services (previously known as BPEL4WS) (Andrews et al., 2003) is a popular, commonly accepted and specialized language that emerged as a de-facto industry standard for defining executable business processes. BPEL extends the web service interaction model (Andrews et al., 2003) and became an OASIS standard (Arkin et al., 2005) for web service composition and orchestration. It is an XML-based language designed to execute a series of web-based transactions and/or characterize interfaces needed to complete web-based transactions in a service-oriented setting. BPEL combines two workflow languages: WSFL (Leymann et al., 2001) and XLANG (Peltz, 2003), and supports a web service technology stack that contains SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination and WS-Transaction (Andrews et al., 2003; Ouyang et al., 2007).

BPEL provides an environment where enterprises can easily and efficiently develop their business processes and quickly respond to changing needs. BPEL is platform independent and supports features such as *scalability* and *flexibility*. To define business processes, BPEL provides a variety of XML constructs, including: *partners* to define the actors in a business transaction; *containers* to define the messages that need to be transmitted; *operations* to define the type of web services required; and *port types* to define the web service connections required for the operations (Andrews et al.,

2003; Ouyang et al., 2007). These constructs can also be used to specify the order of operations, their looping and synchronous and asynchronous requirements.

In addition to process orchestration, BPEL has also been extended to support *Choreography* (Decker, Kopp, Leymann & Weske, 2007) as a choreography, language named BPEL4Chor. Orchestration refers to the process, where one central web service (process) controls all the involved web services and coordinates the execution of different operations on them. Here the central web service only knows about the composition, other web services are not required to realize that they are involved in that composition. On the other hand, process choreography does not rely on a central coordinator. Thus, every web service in this setting knows exactly when to execute its operations and when to interact with other web services. Process-centric languages such as UML and BPMN, can be naturally mapped to BPEL (Ouyang, Dumas, Ter Hofstede & Van der Aalst, 2006; Ouyang, van der Aalst, Dumas & Ter Hofstede, 2006; Ouyang, Dumas, Breutel & ter Hofstede, 2006).

Web Services Choreography Description Language (WS-CDL)

The WS-CDL is an XML-based specification language for describing cross-organizational collaborations in a choreography setting, where multiple parties exchange messages using web services to accomplish a common business objective. WS-CDL is designed to define the observable behaviours of multiple parties involved in the collaboration from a global perspective (Kavantzas, Burdett, Ritzinger & Lafon, 2005). WS-CDL has been designed in a manner that it can be used in conjunction with the WS-BPEL (web services business process execution language). Mendling et al. (2008) proposed to derive the BPEL process definitions for every collaborating party from a WS-CDL choreography model.

Figure 2.1 presents an example of process choreography, where a business process diagram (BPD) is represented using the modeling constructs of BPMN. Here, the BPD

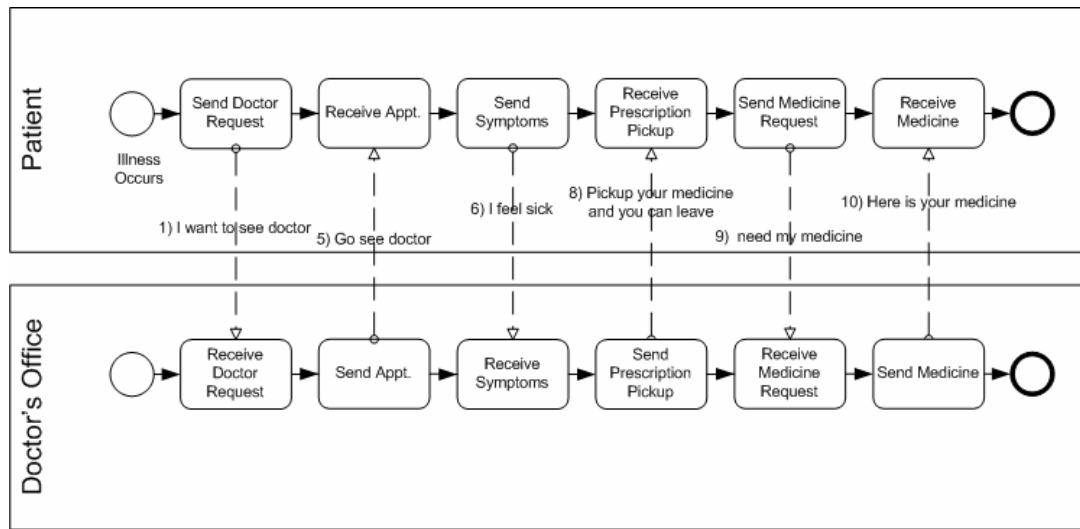


Figure 2.1: A Business Process Diagram represented in BPMN (White, 2004)

is used to capture the interaction between two participants or entities. As shown in the figure, the two participants (Doctor and Patient) are separated by pools and their interaction is represented through message flows that are exchanged to carry out some activity.

There are some existing works on modeling and executing activity-centric process interactions under SOA. Decker et al. proposed choreography modeling by extending Petri Nets (as interaction Petri Nets) (Decker & Weske, 2007) and BPMN (as interaction BPMN (iBPMN)) (Decker & Barros, 2007) for representing process interactions. It is stated that, these extensions can improve the way designers create and understand interaction models and also help in reducing compatibility issues. Later, Decker and Weske presented a comprehensive toolset (Decker & Weske, 2011) that includes a modeling environment for iBPMN interaction models and interaction Petri Nets, and supports the analysis of these models.

Barker et al. (2009) also introduced a choreography language, named Multiagent Protocols (MAP), based on process calculus (Milner, Parrow & Walker, 1992) for the specification, verification and enactment of web service interactions. Wieczorek et al.

(2009) presented a choreography modeling language, called Message Choreography Modeling (MCM), to address the service interaction requirements in the ERP software development context.

There also exists a few declarative languages to specify activity-centric processes and their interactions. Let's Dance (Zaha, Barros, Dumas & ter Hofstede, 2006) is a declarative language for modeling behavioural dependencies between web services that capture message interactions from both local and global viewpoints. This language is proposed based on Computation Tree Logic (CTL) (Clarke & Emerson, 1981) to address the crucial requirements (abstraction, comprehensibility and suitability) of service interaction modeling.

Van der Aalst and Pesic (2006) proposed a declarative service flow language, called DecSerFlow, for the specification and enactment of service flows and their monitoring based on constraints (or rules) defined in temporal logic (Clarke, 1999). The authors state that this language can be extended to monitor the conformance of service flows. Later, Montali et al. (2010) adopted DecSerFlow to propose mappings onto Linear Temporal Logic (LTL) (Jard & Jeron, 1989) and Abductive Logic Programming (Kakas, Kowalski & Toni, 1992) to enrich the expressiveness of this declarative language, and also to enable the verification of service choreographies.

DECLARE (Pesic, Schonenberg & Van der Aalst, 2007) is a constraint-based system that provides declarative semantics to model, execute and verify activity-centric processes based on temporal logic which is mainly used in specifying the constraints that a process model should follow during its execution. DECLARE can be used to develop and execute DecSerFlow models (Van Der Aalst & Pesic, 2006; Montali et al., 2010) or other models specified in LTL based language.

2.1.3 Inter-Organizational Business Process Modeling

A plethora of approaches exist for the modeling and management of activity-centric Inter-Organizational Business Processes (IOBPs). Most of these existing approaches extend the key concepts of *public views* and *private views* proposed in (van der Aalst & Weske, 2001) to support this requirement.

Van der Aalst and Weske (2001) were the first to propose a view-based approach (Public-to-Private (P2P)) to support inter-organizational workflow modeling based on the idea of inheritance. The proposed approach enables each participant in the collaboration to have a controlled visibility over their *public workflow* (common part of original workflow that all the participants have agreed upon) and *private workflow* (part of the workflow that belongs to a specific participant) and allows the refinement of their private parts without affecting the agreed execution of original workflow. Van der Aalst and Basten (2002) later proposed inheritance-preserving transformation rules to guarantee that the dynamic changes made by a participant do not affect the privacy and autonomy of other participants involved in the workflow collaboration.

Liu and Shen (2003) proposed an approach to derive a process view model from conventional activity-centric process models to provide different participants with the required process information in the form of an abstracted process for the effective management of workflow. The authors also presented an algorithm to support the automatic generation of a process view model and a formalism to define that model. This model is further extended in (D. R. Liu & Shen, 2004) to address coordination issues in managing workflows within inter-organizational collaboration.

Chiu et al. (2004) proposed an interoperation model to facilitate the management of E-service contracts and their interoperability based on the notion of *workflow views*. As an interaction mechanism, workflow views (Chiu, Karlapalem, Li & Kafeza, 2002) provide increased support for the cross-organizational workflow interaction and enable

controlled visibility of subworkflows by external parties.

Schulz and Orlowska (2004) proposed to facilitate the execution of cross-organizational workflows. The authors designed an architecture for cross-organizational workflows to enable the execution of private workflows without compromising the privacy and contingency of other private workflows and a scalable visibility of structural information to trading partners.

Chebbi et al. (2006) presented a view-based approach to support the dynamic co-operation of inter-organizational workflows in a service-oriented setting. The proposed approach provides varying levels of visibility over workflows and their resources to enable participating organizations to retain the privacy and security of their internal workflows. The approach also enables participants having freedom to change their workflows without reflecting that change on their roles in the workflow cooperation.

Lin (2007) proposed an approach to model loosely-coupled inter-organizational workflow from the local process views of participating organizations. A design compatibility analysis mechanism and its implementation tool are also presented for detecting incompatibilities among different organizations from their local process views.

Eshuis and Grefen (2008) presented an approach to automatically construct customized process views from a structured cross-organizational business process. To protect the privacy and ownership of a participant, a customized process view is constructed by hiding the private information of the underlying process and providing only the details requested by the other participant.

Zhao and Liu (2006) proposed a matrix-based framework, which enables a participating organization to derive tracking structures over relative workflows (X. Zhao, Liu & Yang, 2005) and the involved relevant workflows of its partner organizations. The framework also supports workflow monitoring through generated tracking structures. The authors later proposed a role-based process view model (X. Zhao, Liu, Sadiq & Kowalkiewicz, 2008) to support view derivation and composition between different

participants. A process view can be derived from another view or composed from multiple views. Rules have been defined to guarantee the structural consistency and validity of resulting process views. Following works of these authors (X. Zhao, Liu, Sadiq, Kowalkiewicz & Yongchareon, 2009, 2011) focused on presenting a framework that supports process abstraction and concretisation and enables the process view implementation in the web service environment.

Jiang et al. (2010) proposed a process view approach to manage cross-organizational workflows that combines process views and Timed Colored Petri Net (TCPN) (Jensen & Kristensen, 2009). This approach is presented in Figure 2.2, which includes a formal definition, the mapping from TCPN workflow models to process-view workflow models in which the control flow and data flow aspects are considered together, and the collaborative execution mechanisms of cross-organizational workflow instances. In addition, a hybrid Peer-to-Peer (P2P) based decentralized workflow management system (WMS) combined with the process-view approach is also proposed and built on top of the open JXTA platform to provide a flexible and scalable architecture for the management of cross-organizational workflows.

Ye et al. (2009) presented a framework to construct atomicity-preserving process views and their analysis. The framework is based on process algebra and used to analyze a set of interacting public process views that use atomicity spheres (Schuldt, Alonso, Beeri & Schek, 2002). Similarly, Bouchbout and Alimazighi (2011) proposed a framework based on the principles of SOA for modeling inter-organizational business processes which preserves the autonomy, privacy and heterogeneity of participating entities.

Eshuis et al. (2011) introduced the notion of transactional process views to facilitate trustworthy and fine-grained collaboration between organizations. These views are constructed from an internal business process annotated with transactional properties, including nested and chained transactions, and can be used to obtain robust and reliable

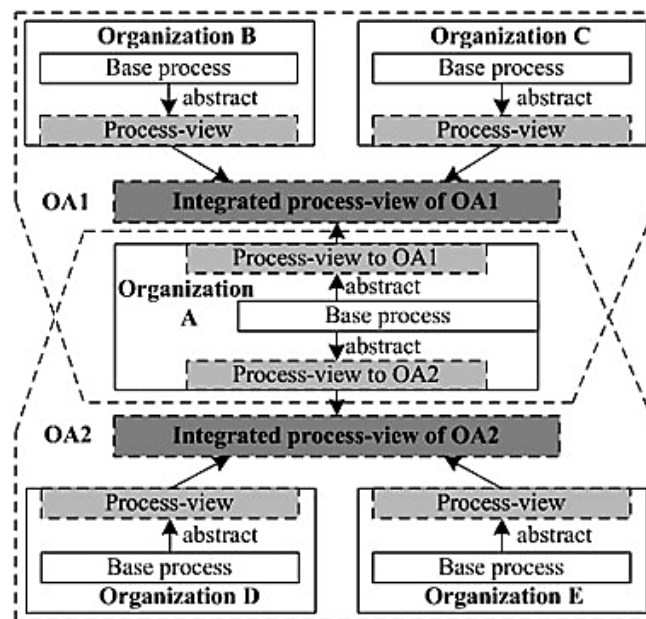


Figure 2.2: Process-view approach (Jiang et al., 2010)

process behaviours from the public external level.

Norta et al. (2014) presented a reference architecture for managing dynamic inter-organizational business process collaboration. The proposed architecture supports the evaluation and design of standard and concrete architectures for the business process collaboration. A set of functional and non-functional requirements for the proposed architecture have been defined from careful observation of collaboration features.

Reypens et al. (2016) presented a multi-level cyclical process framework to support multiple stakeholders that participate in a collaboration network to create business value. This framework enables maximizing value for both individual stakeholders in the network and the network as a whole by presenting ways to address the challenges of understanding and cooperating in a dynamic collaborative environment.

Recently, Barcelona et al. (2018) proposed a Collaborative Business Generation (CBG) framework that follows a bottom-up approach to generate collaborative business process model from the individual process models of different organizations that may

have created with different modeling languages. The framework includes a meta-model, a method, a set of model transformations and a tool to support the creation of collaborative process models by maintaining privacy and autonomy.

2.2 Artifact-centric business process modeling

Nigam and Caswell (2003) proposed the notion of modeling business processes from artifacts (or business artifacts) and their interacting lifecycles. In general, an *artifact* can be described as a means to record information required to conduct the operations of a business, while in business terminology it is better described as a key business-relevant entity responsible for driving overall operations of a business (Kunchala, Yu & Yongchareon, 2014). The *lifecycle* represents the behaviour of artifact in a several possible stages that an artifact navigates in response to business activities.

The core objective of artifact-centric (operational) modeling is to provide an *intuitive* and *flexible* representation to business people that is used to analyze, manage and control their business operations. According to Nigam and Caswell (2003), an *artifact* and its *lifecycle* can be defined as follows:

- “An artifact is a concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business”.
- “An artifact lifecycle captures the end-to-end processing of a specific artifact, from creation to completion and archiving”.

To further demonstrate this modeling paradigm, an example of *guest check* artifact and its lifecycle is presented in Figure 2.3. The *guest check* is a key entity in the restaurant business around which the entire business is operated. As shown in the figure, the lifecycle of a *guest check* artifact captures the processing steps of this artifact in a set of tasks (or activities) and repositories. In the lifecycle, a task updates the content of *guest check* and a repository stores this artifact until it is requested by another

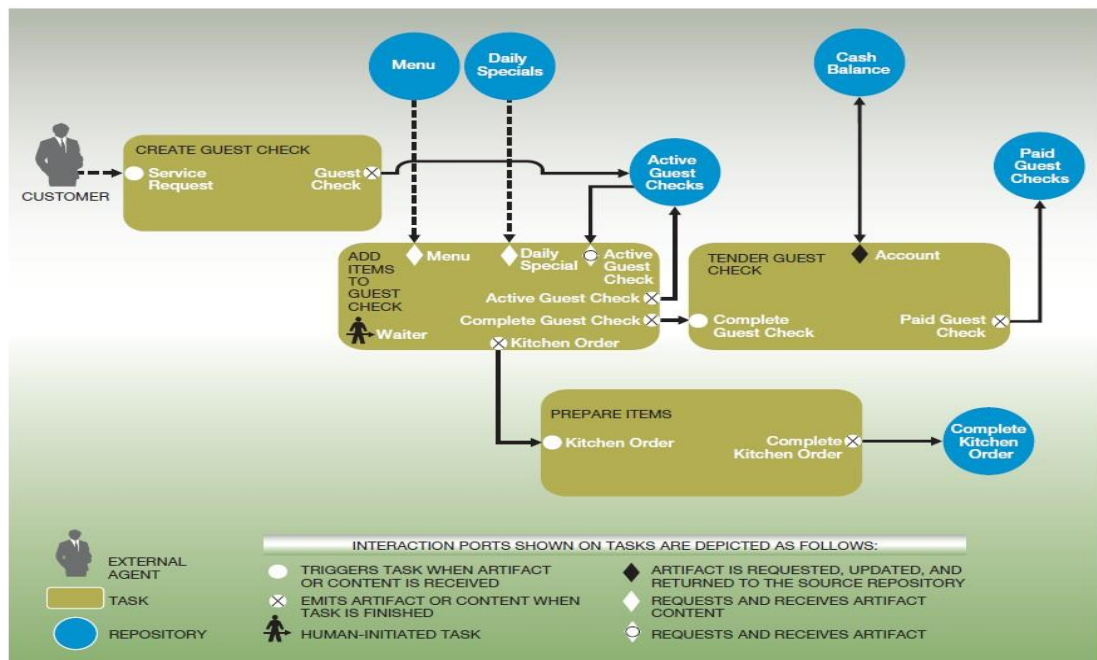


Figure 2.3: Lifecycle of Guest Check (Nigam & Caswell, 2003)

task. A task can also create new artifacts while processing an artifact, which results in synchronization dependencies between corresponding artifact lifecycles.

It can be seen in Figure 2.3 how an artifact serves as a building block to model the artifact-centric operational processes by aggregating the *information* (data) and *process* (control flow) aspects in a more comprehensive manner (Kunchala et al., 2014). According to Nigam and Caswell (2003), every artifact consists of a unique identity and self-describing content that can be represented as nested name-value pair. Technically, artifacts correspond to complex objects in the object-oriented (database) world, which have structure and relations with associated formal algebra. The ER (Entity-Relationship) data model can also be used to represent artifacts and their nested attribute relations (Bhattacharya et al., 2009). The Extensive Markup Language (XML) also provides syntax in the form of Data Type Definition (DTD) for specifying the attributes of artifacts.

Bhattacharya et al. (2009) proposed a four-dimensional framework, named *BALSA*,

for modeling artifact-centric business processes. The four inter-related but separable dimensions of this framework are illustrated in Figure 2.4, which includes *Business Artifacts*, *Macro-Level Lifecycles*, *Services* and *Associations*.

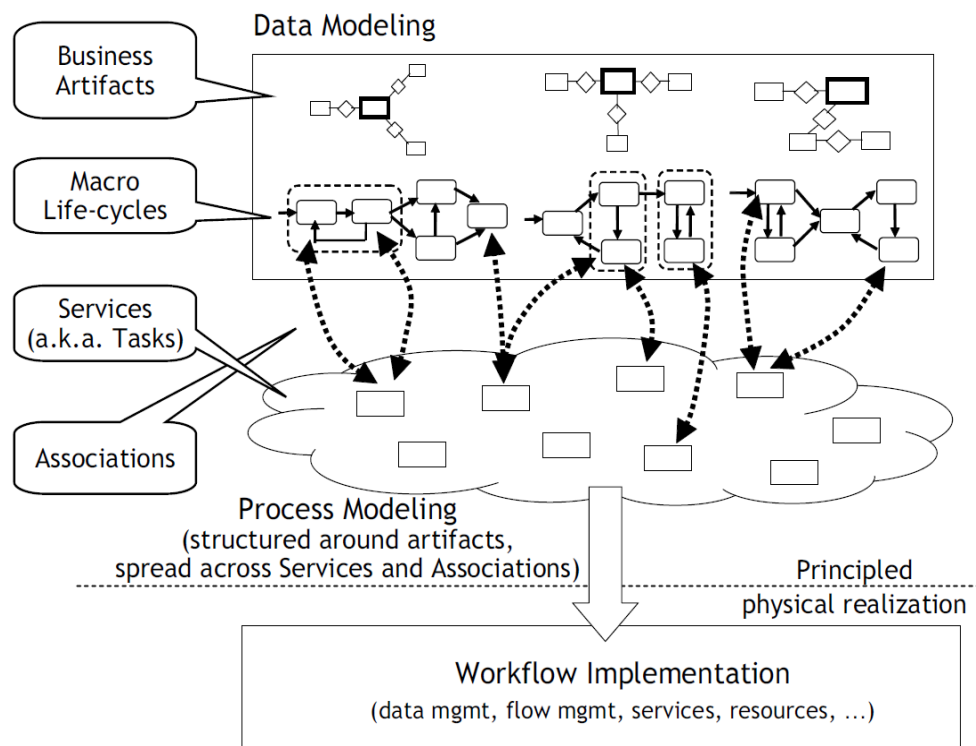


Figure 2.4: Four Dimensional Framework (Bhattacharya et al., 2009)

Business Artifacts (Information Model). Intuitively, business artifacts (or artifacts) combine data and control flow into a holistic unit to drive an entire business process execution. In general, the artifact not only contains data related to a business object but also its lifecycle and other contextual information. These artifacts may have different life expectancies, where some are short-lived and others are long-lived.

Macro-Level lifecycles. The (macro-level) lifecycles describe the possible business-relevant stages in the evolution of artifacts, from their creation to completion and archiving. The lifecycles are usually represented using a variant of finite-state machines, where a state represents a stage in the life of an artifact.

Services. A service is a business task (or action) performed on an artifact to progress

towards the objectives of a business. Invocation of a service on an artifact may change the state and/or content of that artifact. Services are specified in a declarative manner using *pre-* and *post-conditions* (Kunchala et al., 2014). A service here can correspond to a service in the SOA-based distributed environment.

Associations. The associations describe how the services must be operated on artifacts by specifying constraints. The constraints here correspond to conditions that restrict the invocation of services on the respective artifacts. The constraints can be specified in a procedural manner by using flowcharts (Nigam & Caswell, 2003) or in a declarative style based on ECA (Event-Condition-Action) rules (Bhattacharya, Gerede et al., 2007; Deutsch, Hull, Patrizi & Vianu, 2009; Fritz, Hull & Su, 2009).

Business artifacts, as a data-centric paradigm for workflow and business process specification (Hull et al., 2009) have been successfully employed in various client engagements at IBM (Bhattacharya, Gerede et al., 2007; Chao et al., 2009) for business analysis and business-driven development. In the past decade, various aspects of this paradigm have been extensively studied and extended to reveal the importance of this approach in practice. In the following section, these research aspects are categorized and existing literature is reviewed.

2.2.1 Process Design Methodology

The artifact-centric approach enables rich flexibility in the creation and evolution of business processes through the separation of data and control flow aspects (Bhattacharya, Gerede et al., 2007). There are some data-centric approaches, including: *Adaptive Documents* (ADocs) (S. Kumaran, Nandi, Heath, Bhaskaran & Das, 2003); *Case handling* (Van der Aalst, Weske & Grünbauer, 2005); *Document-oriented workflows* (Wang & Kumar, 2005); *Adaptive Business Objects* (ABOs) (Nandi & Kumaran, 2005); *FlexConnect* (Redding, Dumas, ter Hofstede & Iordachescu, 2010); and *PHILharmonicFlows*

(Künzle & Reichert, 2011). These approaches share some level of similarity with the artifact-centric approach, where they also look at a process from the data perspective with a central focus on evolving business entities.

Bhattacharya et al. (2009) proposed a design methodology based on the three-level framework presented for processes that emphasize artifacts (or data) and their (macro-level) lifecycles. As illustrated in Figure 2.5, this methodology consists of four major steps. Step 1 is to discover the key business entities (artifacts) and important stages in their lifecycles to develop a high-level business process specification. Step 2 is to construct Business Operation Models (BOM) from the logical specification of discovered artifacts, services and their associations that are specified using ECA rules. The BOM as a logical specification of business operations provides meaningful insights to stakeholders and supports technical analysis and verification. Step 3 is to develop the conceptual flow diagram that coordinates the artifacts and services in a procedural manner to meet the operational requirements specified in the BOM. Step 4 is for obtaining an operational workflow system from the BOM specification.

STEP 1: *Business Artifact Discovery*

- (a) Identify critical artifacts for the business process
- (b) Discover key stages of artifact's life cycles from the scenario-based requirements

STEP 2: *Design of Business Operations Model (BOM)*

- (a) Logical design of artifact schemas
- (b) Specify services for artifacts needed for moving artifacts through the life cycles
- (c) Develop ECA rules that enable artifacts progress in their life cycles

STEP 3: *Design of Conceptual Flow Diagram*

STEP 4: *Workflow Realization*

Figure 2.5: Design Methodology from (Bhattacharya et al., 2009)

Recently, Kovář et al. (2017) proposed a methodology, named *uuProcess*, that uses business artifacts as a fundamental building block in modeling enterprise applications to support the development of information systems by the Unicorn Group of companies

(www.unicorn.com). The core components of this methodology are the artifacts and their lifecycles, including activities. In this methodology, an artifact is used to store descriptive information about tangible and intangible objects and its lifecycle is used to manage activities that invoke these objects. This methodology was used successfully to implement Unicorn's internal operations and to manage their customer projects. The resulting uuprocess models can be mapped to executable applications supported by the Unicorn Universe Platform (Kökörčený & Kovář, 2015), a digital construction kit for the development of SOA-based applications from reusable process components.

2.2.2 Process Discovery and Construction

According to the design methodology presented in Figure 2.5, the artifacts and their lifecycles must be discovered first to construct an artifact-centric process model. Discovering the artifacts and their lifecycles require deep understanding of critical data and how that data is being manipulated throughout the process. Following here are some existing approaches for discovering and constructing artifacts-centric processes.

Fritz et al. (2009) proposed studying the automatic construction of goal-directed declarative artifact-centric workflows with the concern of general setting, design-time analysis, and the synthesis of workflow schemas from goal specifications. The proposed approach consists of an algorithm to automatically construct an artifact-centric workflow model from *services* that satisfy the specified conditions (or goals).

Maamar et al. (2010) proposed a method to discover and model artifacts from business requirements. The method applies a bottom-up analysis to discover artifacts from three perspectives: *data*, *operation* and *connection*, which describe the discovery of artifacts and their dependencies, operations and their dependencies, and finally to consolidate the artifacts and operations to build a system.

Nooijen et.al (2012) proposed a technique to discover the lifecycles of artifacts.

The proposed technique consists of algorithms that can automatically extract artifact schemas and corresponding lifecycle logs from a structured relational data source that describes how each artifact evolves during process execution. This work has been extended in (X. Lu, Nagelkerke, van de Wiel & Fahland, 2015), where the authors focused on semi-automatically discovering the artifact-centric process models that describe artifacts and their lifecycles, including the interactions between the artifacts.

Popova et al. (2013) aimed to address the problem of artifact lifecycle discovery. The proposed method first generates artifact-centric logs by determining the events that belong to the instances of artifacts, then discovers the lifecycle of every artifact from the generated logs that are represented using Petri Nets. Then the discovered lifecycles are translated into an artifact-centric meta-model, such as Guard-Stage-Milestone (GSM) notation (Hull et al., 2010). The proposed method has been extended and implemented as a generic open-source process mining framework, called ProM (www.processmining.org), to automatically discover artifact lifecycles and their synchronization conditions in the following works (Popova & Dumas, 2013; Popova, Fahland & Dumas, 2015).

Li et al. (2017) presented an algorithm to support the automatic discovery of object-centric behavioural constraint (OCBC) models from object-centric event logs. Similar to an artifact-centric process, an OCBC declarative model describes the interaction between data objects (or artifacts) and their complex dependencies.

Van Eck et al. (2017) studied the automatic discovery of composite state machines representing artifact-centric processes from event data. The approach also supports the analysis of behavioural interactions between artifacts. The proposed approach has been fully implemented as a ProM plug-in and the CSM Miner, an interactive tool that focuses on discovering and exploring state-based processes from multiple perspectives.

2.2.3 Process Specification and Verification

Artifact-centric process specification and verification is one of the most largely explored areas of research. In the past decade, many efforts have been made to support the design and analysis of artifact-centric business processes. In this regard, the research on process specification aimed at formalizing artifact-centric processes, while process verification aimed at checking some temporal properties on these processes.

Liu et al. (2007) formulated nine operational patterns to describe the behaviour of artifacts. These patterns form a basis for constructing artifact-centric operational models. The authors also proposed the transformation of these operational models into colored Petri Nets to verify the correctness of these models through reachability analysis.

Gerede et al. (2007) aimed to analyze artifact-centric operational models by identifying some properties in these models, such as arrival, persistence, uniqueness and redundancy. Mainly, the authors propose a formalism for every construct of artifact-centric operational models that enable the static analysis of these properties. Gerede and Su (2007) proposed a temporal logic based language, named Artifact Behaviour Specification Language (ABSL), for specifying lifecycle properties of artifacts. The authors also presented a verification technique that analyzes behavioural properties, including reachability and general temporal constraints specified in ABSL.

Bhattacharya et al. (2007) presented a formal model for specifying artifact-centric business models using semantic web services in the spirit of OWL-S (Martin et al., 2004) and declarative semantics based on business rules. The emphasis is placed on analyzing the semantics of specified artifact-centric models in relation to reachability of goal states, existence of dead ends and redundancy.

Zhao et al. (2009) also introduced a formal model for artifact-centric processes and proposed a declarative language, called TiLE (Time Line Expression), based on Past

LTL for specifying and enforcing business constraints during the execution of these processes. The authors also present the complexity results on the satisfiability of TiLE constraints.

Abiteboul et al. (2009) proposed an AXML (Active XML) artifact model based on the extension of XML (Abiteboul, Benjelloun & Milo, 2008) to specify data-intensive (artifact-centric) processes that evolve over time in a distributed setting. The AXML artifact model captures various aspects of artifacts, such as their states, evolutions specified using declarative constraints over lifecycles, interactions via web services and history recorded as part of the artifact. The authors later proposed to verify the AXML Systems based on Tree-LTL, a tree-pattern-based temporal logic (Abiteboul, Segoufin & Vianu, 2009b, 2009a). An AXML system represents a set of interacting (AXML) documents with embedded web service calls.

Lohmann and Nyolt (2011) proposed extensions for the BPMN standard notation to support the modeling of artifact-centric business processes. These extensions include artifacts, lifecycles, location information, access control, goal states, and policies. Artifacts and their lifecycles form the key constructs of this model. Location information specifies how the location of artifacts is changed. Access control specifies remote accessibility of artifacts. The extensions, such as *goal states* and *policies*, are first proposed in (Lohmann & Wolf, 2010) to exclude undesired behaviour of processes, where policies mainly restrict the execution order of activities that use different artifacts and goal states to specify desired final states for artifacts involved in the process.

Damaggio et al. (2011) addressed the problem of automatic verification of artifact-centric business processes and artifact systems (Deutsch et al., 2009) with an objective to enhance artifact-centric modeling. The authors achieved their goal through the static verification of all the runs of an artifact system (a collection of artifacts and services) in satisfying the desired correctness properties expressed in a LTL-FO (First Order extension of Linear-time Temporal Logic (LTL)).

De Giacomo et al. (2012) presented a verification formalism for conjunctive artifact-centric services that pose balanced attention to both the data component (a full-fledged relational database) and process component (tasks that act on database) following the artifact-centric approach. These services also guarantee decidability through a suitable use of conjunctive queries in specifying task pre-conditions and post-conditions. The formalism verifies the temporal properties expressed in a first-order variant of μ -calculus (Emerson, 1997; Luckham, Park & Paterson, 1970) under a reasonably weak acyclicity (Fagin, Kolaitis, Miller & Popa, 2005) restriction on the effects of tasks. In this work, the authors extended the formalism presented in their previous works (Cangialosi, De Giacomo, De Masellis & Rosati, 2010; Hariri, Calvanese, De Giacomo, De Masellis & Felli, 2011). The notion of conjunctive artifact-centric services was first introduced in (Cangialosi et al., 2010), where the conjunctive queries and homomorphism are used for the verification; and the work (Hariri et al., 2011) first studied the weak acyclic processes over relational artifacts by using negation and full first-order queries in defining the pre-conditions of actions.

There are some other declarative approaches to specify and verify artifacts and their evolving lifecycles. Kucukoguz and Su (2010) proposed ArtiNets, a formal variant of the artifact-centric workflow model, to specify artifact lifecycles and their constraints. Inspired by a Declarative Service Flow Language (DecSerFlow) (Van Der Aalst & Pesic, 2006) that supports declarative constraints on service flows using temporal logic, the ArtiNet workflow model also allows a declarative specification of lifecycle constraints that govern workflow enactment towards the defined objective. The ArtiNet framework resembles Petri Nets (Murata, 1989), with two differences: where artifacts form the key constructs of ArtiNet model instead of tokens; and the firing rule that consumes only one artifact though the corresponding transition contain multiple input or output artifacts.

Hull et al. (2010) introduced a Guard-Stage-Milestone (GSM) meta-model for

specifying the artifact lifecycles in a more declarative way. The authors also proposed a formal specification and operational semantics for GSM in order to support the interaction between multiple artifact instances (Hull et al., 2011). As the name implies, GSM uses three key constructs in modeling artifact-centric processes, that include *stages*, *guards* and *milestones*. Here, *Stages* represent activities to be performed by the artifact instances; *Milestones* specify goals that the corresponding artifacts must achieve; and the *Guards* refer to conditions or triggering events that control stages and milestones. Event-Condition-Action (ECA) rules are used in the behavioural specification of GSM models in order to support parallelism within artifact instances and modular structure through hierarchical clustering of activities, thus enabling a basis for formal verification and reasoning (Damaggio, Hull & Vaculín, 2013).

Belardinelli et al. (2012a) presented an abstraction technique to verify the artifact-centric multi agent systems, i.e., systems of agents interacting through artifact systems. The verification is through model checking, where the decidability of the deployed artifact system is checked against temporal epistemic properties expressed in a FO-CTLK. In this work, the authors extend their verification technique from their study (Belardinelli, Lomuscio & Patrizi, 2011), where the basic temporal properties expressed in FO-CTL for artifact systems were verified. The authors later proposed semantics to GSM-based artifact-centric systems to verify the knowledge of participants in a multi-agent setting (Belardinelli, Lomuscio & Patrizi, 2012b).

Gonzalez et al. (2012) presented an approach to verify GSM-based artifact systems. The proposed methodology is based on developing a symbolic representation of the GSM models for verifying the behaviour of artifacts. A model-checking tool GSM Checker (GSMC) has also been implemented to support automatic verification. The authors extended their formalism (Gonzalez, Griesmayer & Lomuscio, 2013) to define and verify GSM-based artifacts systems from a multi-agent perspective, based on the declarative semantics proposed for an artifact-centric multi-agent system (AC-MAS).

Solomakhin et al. (2013) proposed an approach to automatically verify the decidability property over GSM models by translating them into DCDSs (Data-Centric Dynamic Systems) (Bagheri Hariri et al., 2013), a formal framework for data or artifact-centric processes. Bagheri Hariri et al. (2013) proposed a formalism for verifying the decidability property over (relational) DCDSs based on the First Order variant of μ calculus. The authors also proposed (Hariri, Calvanese, Montali, Santoso & Solomakhin, 2013) an OWL 2 QL (Motik et al., 2009), the ontology-based semantics for the specification and verification of temporal properties over the semantic layer of GSM models. The authors introduced Semantically-enhanced Artifact Systems (SAS) as a variant of GSM, which are verified according to the evolution in the underlying GSM model.

Deutsch et al. (2014; 2016; 2018) proposed the verification of Hierarchical Artifact Systems (HAS) that extend the artifact systems of their previous work (Deutsch et al., 2009) with the core elements of GSM model. The authors introduced a Hierarchical LTL-FO (HLTL-FO), a variant of First Order Linear Time Logic (LTL-FO) to verify the core fragments of these systems, namely tuple artifact systems (TAS), which are composed of a database, a set of artifact attributes and services that specify *pre-* and *post-conditions* on these systems. An automatic verification tool for these systems is also proposed and implemented in their following works (Y. Li, Deutsch & Vianu, 2017b, 2017a).

Guosheng Kang et al. (2019) proposed verifying the behavioural soundness of the artifact-centric process model that includes synchronizations. The verification approach maps each artifact lifecycle model to Petri Net and then integrates them into a workflow net based on a set of synchronization rules. The reachability graph of this workflow net is used to derive all the implicitly specified service execution sequences. The behavioural soundness is then verified by investigating the proper completion of every service execution sequence.

2.2.4 Inter-Organizational Business Process Modeling

Several recent works have emphasized extending the artifact-centric paradigm to the modeling and management of Inter-Organizational Business Processes (IOBPs). In this regard, most of the existing works based their contributions on the notion of *public* and *private* views (public and private) proposed in (van der Aalst & Weske, 2001).

Hull et al. (2009) proposed artifact-centric interoperation hubs to facilitate the process collaboration in an inter-organizational setting. The hubs provide a centralized location for autonomous organizations to communicate and synchronize their business processes by sharing their public data to achieve a common objective. The authors proposed three types of access restrictions *Windows*, *Views* and *CRUD* (Create-Read-Update-Delete) for the participating organizations to *see*, *read*, *write* and *modify* data (artifacts) from/to a hub. An inter-operation hub supports persistent visibility of artifacts for each organization participating in the collaboration.

Lohmann and Wolf (2010) introduced the concepts of *agents* and *locations-aware artifacts* based on a Petri Net model in the artifact-centric inter-organizational setting. The authors proposed a mechanism to automatically generate an interaction model, that may serve as a contract between the agents to ensure goal states are reached, by the artifacts involved in the interaction. The authors believe that their interaction model is more appropriate for artifact-centric inter-organizational business collaborations than the idea of centralized artifact hubs such as those proposed in (Hull et al., 2009).

Yongchareon and Liu (2010) introduced a process view framework for artifact-centric business processes. The framework consists of artifact-centric process (ACP) models that describe a set of *artifacts*, *services*, *rules* and *views*; a construction approach to build role-specific process views from the underlying processes based on some view requirements; and a set of consistency rules that preserve the structural and behavioural correctness of constructed view from the base process. The authors later extended this

framework (Yongchareon, Liu & Zhao, 2011) to model artifact-centric IOBPs. The Artifact-Centric Collaboration (ACC) model extends the ACP model (Yongchareon & Liu, 2010) to capture inter-organizational processes with four core constructs: *roles*, *artifacts*, *services*, and *business rules*. Here, the artifacts are classified into local (private) and shared (public) that enable collaboration between organizations. The framework supports the participated organizations to have their own view of the process and freedom to change and implement their own parts while preserving global correctness of the collaboration (Yongchareon et al., 2011). In this work, the authors also presented a mechanism to validate public and private views to avoid the compliance issues in process collaboration.

Yutian Sun et al. (2012) proposed a declarative choreography language for artifact-centric business process collaboration, which can express correlations and collaborations at process- as well as instance-level. The language is developed based on four key aspects, including: *artifact schema*, *cardinality constraints*, *messages* and *First Order based logic rules*. The authors also developed a mechanism to realize artifact-centric process choreographies specified in the proposed language.

Yongchareon et al. (2015) proposed a formal view framework for the modeling and verification of artifact-centric IOBPs. The framework is presented in Figure 2.6, which consists of an artifact-centric process meta-model, mechanisms to construct private (local process) and public (shared process) views, and a mechanism to validate changes in the local processes. The framework is designed in such a way that it facilitates participating organizations to customize their internal operations while ensuring the global correctness of their collaboration. The authors also presented a software tool, named *Artifact-M*, to help organizations to automatically construct a minimal and consistent public view from their processes.

Koutsos and Vianu (2017) proposed to extract process-centric views consisting of the sequences of services applied during the linear or branching-time runs of an

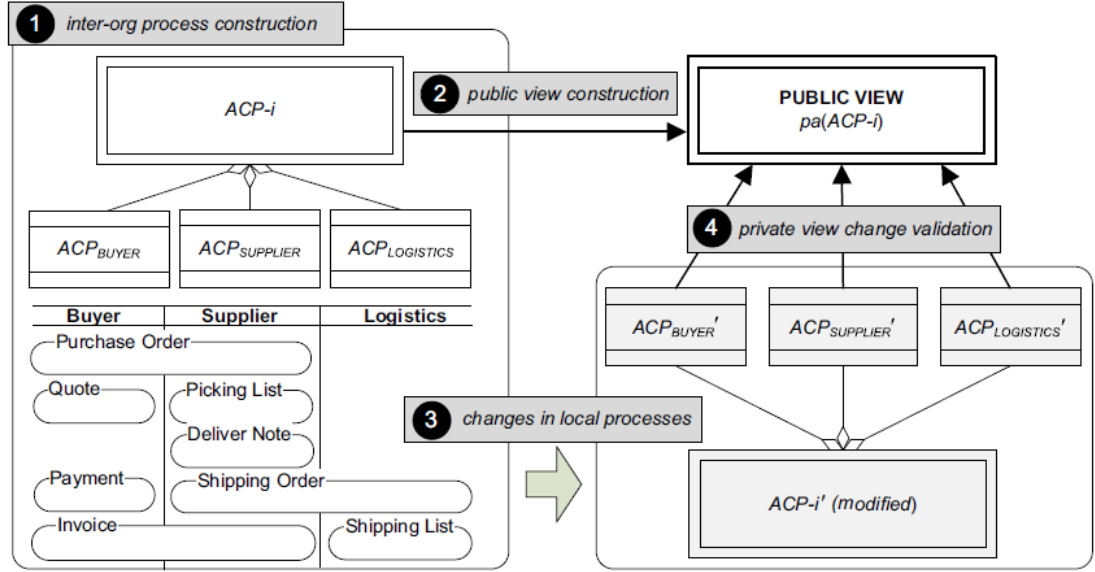


Figure 2.6: Artifact-centric View Framework (Yongchareon et al., 2015)

artifact workflow system. The authors also proposed a verification approach to check the decidability for branching-time properties expressed using Linear Time and Branching Time (LTL and CTL) Temporal Logic on these workflow systems.

2.2.5 Process Realization

According to the survey conducted by Hull in (2008), the implementation of artifact-centric processes was challenging due to the lack of executable workflow technologies and standards in support of this novel paradigm. One common approach to implement artifact-centric processes thus follows a process of transformation, where the artifact-centric process is transformed into an activity-centric process that is then implemented easily using existing workflow languages such as BPEL (Cohn & Hull, 2009; Ngamakeur et al., 2012).

Bhattacharya et al. (2009) proposed a framework for the modeling and execution of artifact-centric business processes. This framework consists of three levels: *Business Operations Model (BOM)*, *Conceptual Flow* and the *Operational Workflow*. BOM

is the basis for system implementation. BOM mainly provides a detailed logical specification for business process execution in terms of artifacts including services that specify *pre-* and *post-conditions* and Event-Condition-Action (ECA) rules. Next, the Conceptual Flow layer is for capturing the BOM in a procedural manner, while hiding the implementation details and supporting process optimization. The bottom layer is the Operational Workflow, where the executable services communicate using messages and operate on the artifacts.

Cohn and Hull (2009) state that artifact-centric models are *actionable*, meaning that they can be mapped to executable models using the IBM BELA (Business Entity Lifecycle Analysis) tool (Strosnider et al., 2008). This tool enables process designers to automatically generate executable models and run these models on tools like the IBM Web Sphere Process Server (Ferguson, 2001). The authors also propose a prototype, named *Siena* (Cohn et al., 2008) that uses a direct architecture, where the artifact model is represented as an XML document and executed through a direct interpretation. Siena also enables users to model business artifacts and processes as composite web applications, which are then run on the underlying model execution engine (2008).

Guohua Liu et.al. (2009) introduced an Artifact Conceptual Flow (ArtiFlow) model as a variant of the artifact-centric workflow model. The primitive constructs of this model include: *services*, *events*, *artifacts* and *repositories*. The authors also proposed a mapping from ArtiFlow to the BPEL in order to show that the automated realization of artifact-centric workflow is achievable. Based on the ArtiFlow model, a prototype implementation, called A-Stein, has been proposed in (D. Zhao, Liu, Wang et al., 2011) for the management of artifact-centric systems.

Abiteboul et al. (2010) presented an AXART system based on the Active XML (AXML) that extends the standard XML with embedded service calls (Abiteboul et al., 2008; Abiteboul, Segoufin & Vianu, 2009b). This system can also be used to implement artifact-centric process collaboration in a dynamic environment and to manage updates

on the corresponding artifacts.

Xu et al. (2011) proposed an artifact-centric workflow model, namely *EZ-flow*, based on *ArtiFlow* (G. Liu et al., 2009) and a mechanism that supports the run-time modification of workflow execution. The proposed mechanism aims to study on how workflow executions can be incrementally modified in order to accommodate changing requirements in artifact-centric models by using a workflow engine.

Dong Li and Wu (2011) presented an algorithm to translate an artifact-based business process model into BPEL. The authors also developed a tool to design and translate artifact-based processes. The tool takes an SVG (XML-based) representation of the artifact process and generates an executable BPEL based on a set of conversion rules.

Danfeng Zhao et al. (2011) proposed an approach for the execution and run-time monitoring of artifact-centric business processes based on the *ArtiFlow* Model introduced in (G. Liu et al., 2009). This approach also follows a translation from the *ArtiFlow* model to the BPEL document to automatically implement and run on the underlying BPEL engine.

Limonad et al. (2012) introduced a realization framework for Interoperation Hubs (I-Hubs) that had been first proposed in (Hull et al., 2009) for artifact-centric inter-organizational collaboration. I-Hubs combine the access control aspects of both data- and process-based systems. This framework has been implemented as part of the EU funded ACSI (Artifact-Centric Service Interoperation) project (Hull, 2011). The ACSI project aims to combine the artifact-centric paradigm with SOA to achieve higher levels of abstraction while integrating business processes across organizational boundaries (Dumas, 2011). The ACSI Hub system (Boaz et al., 2014), with the underlying *Barcelona* (Heath et al., 2013) and *Siena* (Cohn et al., 2008) artifact engines, enables service orchestrations using GSM-based artifact lifecycles and finite-state machine based artifact lifecycles.

Ngamakeur et al. (2012) proposed a framework for the realization of artifact-centric

process models in a service-oriented setting. The framework is presented in Figure 2.7, which consists of a workflow model and a mechanism to generate executable model from the logical specification of an artifact-centric process (ACP) model. A prototype ACP system is also developed to support automatic realization and execution, where the proposed system uses business rules to control each state of process execution. Here, the authors argue that the traditional approach to artifact-centric process realization possesses some drawbacks, particularly where the process transformation leads to difficulty in process monitoring and also reduces its flexibility. Therefore, the authors proposed a direct realization approach that does not require the transformation of artifact-centric processes into traditional activity-centric processes.

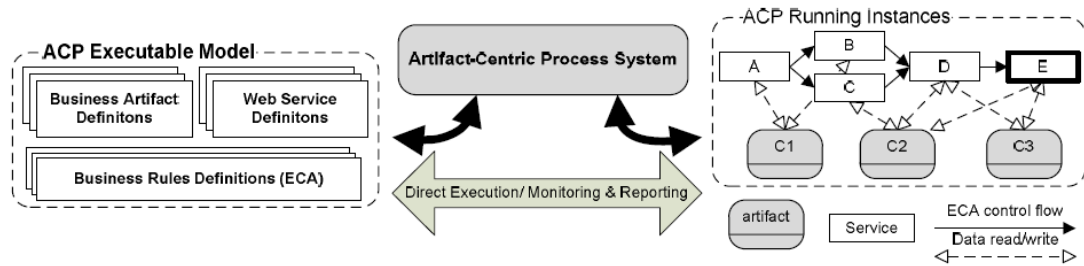


Figure 2.7: Artifact-centric Process Realization Framework (Ngamakeur et al., 2012)

Yongchareon et al. (2014) proposed a workflow execution platform for artifact-centric collaborative processes in a service-oriented setting. The platform is developed based on the view-based concepts proposed in their previous works (Yongchareon et al., 2011, 2015). By utilizing event-driven and service-oriented architectures for its design and implementation, this platform can be useful to enable centralized control in the distributed environment.

Fan et al. (2015) demonstrated the automatic translation of artifact-centric business processes into activity-centric processes to execute them on a process execution engine called JTangFlow. The authors first proposed flow models for these two types of process models and presented an algorithm to demonstrate mapping between the corresponding

flow models.

Lei et al. (2016) proposed a distributed framework for artifact-centric business process modeling by leveraging the REST (Representational State Transfer) architectural style. In this work, the authors mainly focused on improving the scalability of artifact-centric BPM systems in large-scale applications. A prototype, named ArtiREST, was also developed, which supports automated realization and monitoring of artifact-centric process models.

The above discussed existing works reveal how the artifact-centric approach can facilitate process enactment and evolution, in addition to enabling effective business process collaboration compared to the traditional activity-centric approach.

2.3 Process Model Transformation

With the momentum of the artifact-centric paradigm, the BPM research has focused on relating this paradigm with the traditional activity-centric paradigm. In this regard, the transformation of activity-centric process models into artifact-centric process models has received major attention. The proposed research is motivated from this initiative and aims to propose an automated approach to facilitate the model transformation. The transformation approach proposed in this thesis is related to two categories of existing research: *process tree generation* and *artifact lifecycle synthesis*. Therefore, the following section presents a detailed discussion on the existing works in these two categories.

2.3.1 Process Tree Generation

Vanhatalo et al. (2007) proposed a tree-based technique to analyze process models represented as workflow graphs to find control-flow errors. The proposed technique

generates a Process Structure Tree (PST), a directed tree by decomposing a block-structured process model into a special kind of process fragment named canonical single-entry-single-exit (SESE) fragments. This approach is proposed based on the notion of a *program structure tree* (Johnson, Pearson & Pingali, June, 1994), which is a hierarchical representation of program structure based on the SESE regions of the control flow graph.

Kuster et al. (2008) proposed an approach that extends the Process Structure Tree (PST) (Vanhatalo et al., 2007) to detect and resolve the differences in process models in the absence of change logs. Vanhatalo et al. (2009) presented a unique and modular process decomposition technique with an extended version of their PST called Refined Process Structure Tree (RPST), which represents the hierarchy of subworkflows to analyze structural aspects of process models.

Polyvyanyy et al. (2009) also extended PST (Vanhatalo et al., 2007) to propose an abstraction technique for managing process model complexity. The authors later proposed (2010) another abstraction approach that generates a tree representation for process models to manage the model complexity. The objective of this abstraction approach is to transform the unstructured process models represented in BPMN and EPC with an arbitrary topology into well-structured ones under fully concurrent bisimulation to facilitate the implementation of resulting processes in standards such as BPEL. This technique was implemented later in (Polyvyanyy, García-Bañuelos & Dumas, 2012).

As discussed above, there are several approaches to generate tree representations of business process models for different purposes, such as analyzing and resolving the control flow errors of process models. This section mainly discussed the techniques that are related to the transformation (synthesis) approach proposed in this thesis. In this context, the goal is different, where the approach proposed in this thesis builds the tree representation (or process tree) of an activity-centric process model that is annotated

with artifacts and states in order to generate the lifecycles of artifacts from the resulting process tree.

2.3.2 Artifact Lifecycle Synthesis

Ryndina et al. (2006) presented an approach to generate object lifecycles from business process models represented in UML activity diagrams. This approach checks the possible compliance violations of generated object lifecycles with the given lifecycles and resolves them by iteratively working on them. The proposed approach uses a set of high-level rules for generating object lifecycles from the business process models that contain object data flows. However, these rules do not consider synchronization dependencies that capture the object interactions, which are crucial to observe the consistent behaviour of the base process from its generated object lifecycles. In addition, the evaluation of this approach is limited, where only a simple insurance process model is used to demonstrate the applicability and efficiency of the proposed approach.

Kumaran et al. (2008) proposed an approach to study the duality between the object-centric processes and activity-centric processes. This approach uses a notion of *domination*, where the lifecycles are generated for those objects that are highly manipulated by the activities of activity-centric process model. The generated lifecycles are then linked to create synchronization dependencies between the corresponding objects. Liu et al. (2010) extended this approach to formally demonstrate the duality between the two modeling paradigms and to show the value of proposed transformation in achieving three key aspects, including: process componentization, reusable process design, and accelerating the IT solution development. However, this approach is limited in that it only considers the objects written by the activities, thus missing the objects that they read, for the purpose of object lifecycle generation.

Eshuis and Van Gorp (2012) proposed an automated approach to synthesize the

object lifecycles from the process models represented using UML activity diagrams. This approach consists of two steps, namely *pre-processing* and *filtering*. In the pre-processing step, the process model is transformed into a normal form to ensure that each activity node has only one incoming and outgoing edge. In the filtering step, for each object in the process model, a set of filtering rules is used to create its dedicated activity diagram by filtering out the irrelevant nodes that do not have respective object flows. The filtered models are then used to generate the lifecycles of objects. The authors presented a way to translate these object lifecycles into hierarchical statecharts and also refined and extended the proposed approach (Eshuis & Van Gorp, 2016) with synthesis rules that are used to coordinate the filtered process models based on their common object coordinator. Although this approach claims to be automatic, a manual rewriting of the process model is required when the pre-processing step fails. In addition, this approach poses restriction on the object flows, where each object must have at least one incoming and one outgoing flow from/to the activity node(s) of the process model.

Cabanillas et al. (2011) presented a model-driven approach with algorithms that support automatic generation of object lifecycles from the BPMN process models. This is a three-step approach, where the BPMN process model is first mapped to a semantic Petri Net, then a reachability graph is obtained by analyzing the Petri Net, and finally object lifecycles are generated by processing the reachability graph in a node by node fashion. The proposed approach is developed as a prototype by reusing the code of ProM (Van der Aalst et al., 2009), the process mining toolkit. However, this prototype only implements step 2 and 3 of the proposed procedure, therefore the mapping step has to be done manually. In addition, this approach is only useful for generating the unsynchronized object lifecycles and is not applicable to process models that use AND-gateways that capture the parallel dependencies between the object states.

Popova and Dumas (2012) proposed a method to translate the Petri Net models into the GSM (Guard-Stage-Milestone) models. In this work, the authors aim to address the

problem of discovering artifact-centric process models from event logs. Thus, rather than using event logs to generate the GSM models, the proposed method uses existing process mining algorithms for mining the Petri Net models from event logs. It then discovers the lifecycle for every artifact and generates GSM models from the discovered lifecycles. This method is implemented as a software plug-in for ProM, however, the authors only used a case study to demonstrate the feasibility of the proposed method. This method also ignores the synchronization dependencies between artifacts, without which it is difficult to check the behavioral consistency between the base process and its generated object lifecycles.

Meyer and Weske (2013) proposed a roundtrip transformation between activity-centric process models and artifact-centric process models by utilizing the synchronized object lifecycles that act as a mediator between both modeling paradigms. The proposed roundtrip consists of 5 high-level descriptive algorithms, which are used for the transformation of an artifact-centric process model into the synchronized object lifecycles, and then into an activity-centric process model; for enriching the activity-centric process model with attribute information; a synthesis algorithm for transforming an activity-centric model into the synchronized object lifecycles; then into an artifact-centric model with business rules. Although the synthesis algorithm of the proposed roundtrip considers both the parallel states and the synchronization dependencies of artifacts, the feasibility and applicability of this algorithm is only demonstrated using a simple order and delivery process.

As discussed above, there are various works on generating artifact-centric process models from activity-centric process models that contain artifact data flows. However, most of them have a restricted view of the artifacts and their synchronization dependencies that make it difficult to observe the process behavior. In addition, only a few of them present semi-automated/automated approaches with a limited focus on their applicability and evaluation to reveal the strengths of these approaches in practice. Therefore,

in this thesis, an automated synthesis approach is presented and how this approach addresses the aforementioned limitations to facilitate the proposed transformation is discussed.

2.4 Process Model Merging

In recent years, process merging and consolidation has emerged as a common approach to combine the business processes of two or more organizations to gain benefits, such as to create synergies, for diversification and to increase their market share (Gottschalk et al., 2008). Several approaches exist in the literature for merging/consolidating activity-centric process models. However, the objective of these approaches is different compared to the approach proposed in this thesis, which aims to merge the collaborating processes of activity-centric IOBP models in order to support the transformation of the resulting merged (or integrated) models into the artifact-centric counterparts (such as synchronized artifact lifecycles). Hereafter, existing approaches to process merging and consolidation are discussed in detail.

Shuang Sun et al. (2006) presented an approach to merge workflow nets, where the input models are merged based on the mapping between their tasks or by applying a set of merge patterns (sequential, parallel, conditional and iterative). The mapped tasks are copied into the merged model and the proposed patterns are applied for merging the differed regions of two WF-nets. According to the given merge patterns, the parts of one workflow are merged with another workflow sequentially (by replacing the common regions or inserting one into another), or parallelly (by using control flow constructs and-split and and-join), or in a conditional manner (using or-split and or-join constructs), otherwise following an iterative merge (where the parts of two workflows are considered in an iterative manner). Although the proposed approach is useful for merging both similar and non-similar workflows, it is not automated and it does not

consider the data aspect. In addition, the proposed merge is not evaluated sufficiently to provide valuable insights into real problems.

Mendling and Simon (2006) proposed a similar approach to merge two process views of the same business process represented using the Event-driven Process Chain (EPC) modeling notation. Here, the approach is threefold, where it first identifies the semantic relationships between the activities of two different EPCs; next defines a merge operator to produce an integrated EPC; and then restructures the resulting EPC based on a set of restructuring rules. However, the proposed approach is only applicable to process models that share common activities and that do not contain data aspects. In addition, this approach is not automated and only a case study was used to demonstrate its applicability.

Gottschalk et al. (2008) also focused on merging two EPCs by presenting an algorithm. The algorithm conducts the merge in three phases, which include: reducing the input EPCs into functional graphs; merging them into a single function graph; and then to convert the merged function graph back into an EPC. The algorithm has been implemented in the ProM to support automatic merging of process models. Although the proposed merge considers different EPCs, they are still required to have a similarity in their execution, and they contain no data aspects.

La Rosa et al. (2010) presented a merging algorithm to construct a configurable process model from a collection of process models that belong to multiple application domains. The proposed merge algorithm works by extracting the common fragments of input process models, creating a single copy of these common elements and appending the differences as branches of configurable connectors. A set of reduction rules have been proposed to simplify the merged process model. An extension to this algorithm to deal with process graphs containing data and resource attributes is presented in (La Rosa et al., 2013). The implementation of the merge algorithm is a tool to support the semi-automatic construction of consolidated configurable process models. However, this

algorithm is only useful to merge process models that have common process fragments, but no data flows.

Bulanov et al. (2011) presented an approach that uses Temporal Process Logic (TPL) formalization to merge process models. This approach conducts merging in four steps, which include: an encoding of both processes in terms of TPL formulas; then solving contradictions between the two process models; merging the two sets of formulas; and to reconstruct the final process model based on a set of unified formulas. The proposed approach is limited because it does not consider the data aspects, only supports semi-automated merging and it lacks thorough evaluation.

Derguech and Bhiri (2011) also proposed an approach to merge a collection of business process variants to deliver a configurable business process model. The proposed approach starts with the mapping of input process variants into EPCs. Any structural conflicts that arise during this mapping are identified and resolved as a pre-processing before merging them. The merging algorithm is then used to combine the EPCs and a set of reduction rules are used to simplify the merged model to obtain a configurable business process model. The proposed algorithm is implemented and evaluated using manually created process variants. Like the above approaches, this approach also requires common process nodes that share identical labels to merge and ensure the behaviour of process variants.

Assy et al. (2013) proposed an approach to extract and merge business process fragments around particular activities to construct a consolidated fragment for each activity. The algorithm presented in this work consolidates the process fragments in four steps. First, merging the redundant activities and edges (that belong to same source and target) that are on the same level, where the unmerged activities and edges are then linked to this merged fragment. Second, redundant activities located on different layers are merged by preserving their structural correctness. Third, the labels of merged edges are combined. Finally, an exclusive flow is assigned to other edges. The consolidated

fragment can be presented as a configurable subprocess that includes associated activity, its neighbor fragments and connection elements. The proposed approach was implemented and evaluated using a process model collection. As discussed above, this approach is also limited to process fragments without data elements and uses the common nodes for merging.

Schunselaar et al. (2014) proposed an automated approach to construct a configurable process model based on the general notion of merging business processes. The configurable process model captures the commonalities and variabilities of multiple business process models. An automated analysis technique is also proposed to analyze the consistency between the input and output process models. The proposed analysis technique is implemented as a ProM plug-in. However, only a preliminary evaluation was carried out to demonstrate the usability of this approach.

Zemni et al. (2016) proposed a mechanism to construct a process model from two or more business process fragments represented in BPMN. The mechanism is based on the notion of path matrices, which were introduced to represent the node-based graphs of business processes. The gateway path matrices are used to represent each business process fragment, where the respective elements that share similar source and target activities are merged in a pairwise manner. The proposed mechanism is implemented and evaluated using a library of industrial business process fragments. However, this approach also demands common process regions to merge process models that contain no data aspects.

Derguech et al. (2017) presented an algorithm for merging a set of capability-annotated process models into a capability-annotated configurable process model by extending the merge algorithm presented in their previous work (Derguech & Bhiri, 2011). The proposed algorithm merges process models based on their business capabilities that report on what actions each process element must achieve. The algorithm is implemented and evaluated using manually created and annotated business process

variants. Time and compression rate metrics were considered for evaluating the efficiency of the proposed merge algorithm. However, the focus of this algorithm was also on merging process models based on their common actions.

Recently, Huang et al. (2018) presented an automated approach to consolidate business processes by applying process topic clustering based on business process libraries. The proposed approach consists of a graph mining algorithm to extract process patterns by identifying frequent subgraphs under the same process topic. That information is then filled into a table, and, finally, the identified frequent subgraphs are merged to obtain the consolidated business process. The proposed approach was evaluated using SAP reference models. However, the proposed consolidation was also based on the similarity of process regions, without considering the data aspects.

Although several merge approaches exist, none of them have been proposed for merging the collaborating processes of activity-centric IOBPs that contain artifact data flows. Therefore, this thesis proposes an approach (Kunchala, Yu, Yongchareon & Liu, 2019) that does not demand common process fragments and can merge collaborating processes by considering the artifacts and states shared between them. The resulting merged activity-centric process model can then be used to generate synchronized artifact lifecycles by utilizing the synthesis approach proposed in this thesis.

2.5 Process Model Construction

In recent years, process flexibility and efficiency have been two major concerns for organizations who want to quickly adapt to their changing requirements and outperform their competitors. The BPM literature shows that imperative models (such as activity-centric process models) are more comprehensible and can offer higher efficiency, while rule-based declarative models (such as artifact-centric process models) are more flexible, when the underlying processes are not too large (Pichler et al., 2011; Caron

& Vanthienen, 2016). Therefore, several research works have been carried out to support the forward and reverse transformation of activity-centric and artifact-centric process models. As the existing approaches to forward transformation have already been discussed in Section 2.3, this section discusses the existing works on reverse transformation that aim to transform artifact-centric process models into activity-centric process models.

kuster et al. (2007) presented an automated approach to generate (activity-centric) business process models from one or more (reference) object lifecycles. The authors introduced the two notions: *conformance* and *coverage* for checking the possible compliance violations between business process models and object lifecycles. The *conformance* checks whether a business process contains only those object state transitions that are defined in the given object lifecycle; and the *coverage* to express requirements for the transitions and states in a reference object lifecycle that must be covered by a business process. However, the proposed approach uses unsynchronized object lifecycles to generate the (activity-centric) process model rather than the (object-centric) process model that contains business rules.

Redding et al. (2008) proposed a transformation from the object behaviour models represented in state machines to the process-oriented models. The transformation is based on the identification of causal relations in the state machine and encoding those relations in a heuristics net, from which a Petri Net model is generated and further used to derive the YAWL representation of the process model. The derived model is then simplified using a set of reduction rules. The proposed approach was implemented and evaluated using an asset inspection object model. However, this approach is not useful for transforming process models that contain business rules.

Meyer and Weske (2013) proposed an approach to support a roundtrip transformation between the artifact-centric processes, activity-centric processes and synchronized object lifecycles. However, this approach also uses the synchronized object lifecycles

to generate the activity-centric process models rather than the rule-based artifact-centric process models.

Prescher et al. (2014) proposed an approach to transform declarative process models into behaviourally equivalent Petri Net models. The proposed transformation first maps the declarative constraints of the input model into regular expressions, which are then transformed into the Finite State Automaton (FSA). The resulting FSA is then used to derive the Petri Net model. This approach was implemented and evaluated using a real business case. However, the proposed approach has a drawback that it produces duplicate tasks in the resulting process models that increase the complexity with a higher number of execution alternatives for the resulting process models.

De Giacomo et al. (2015) proposed an extension to BPMN constructs for modeling declarative business processes, where the extension has been named as BPMN-D. The authors also proposed algorithms to translate the declarative models into the BPMN-D models, where the proposed translation involves translating the standard Finite State Automaton of the input model to an equivalent Finite-state Constraint Automaton, then to translate it into BPMN-D. An algorithm for the direct translation of *Declare* (van Der Aalst, Pesic & Schonenberg, 2009) model into the BPMN-D is also presented. However, this approach does not consider the parallel states of objects.

Fan et al. (2015) proposed an approach to automatically translate artifact-centric business processes into activity-centric processes to execute them on a process execution engine called JTangFlow. In this regard, an algorithm is presented that follows a mapping between the two process models to achieve the translation. However, this approach is also not useful for translating the declarative process models.

As discussed above, there exist some approaches related to the transformation of artifact-centric process models into activity-centric process models. However, these approaches either use synchronized or unsynchronized object lifecycles to construct the activity-centric process models or they result in models with duplicate tasks and

can not handle parallelism. Therefore, this thesis proposes a direct model-to-model transformation approach (Kunchala, Yu, Yongchareon & Wang, 2020) that does not introduce duplicate tasks and handles parallelism in the process models.

2.6 Summary

This chapter contains an in-depth literature review to ensure familiarity with the studies that underpin this domain, existing research, challenges, possible opportunities and questions. The most relevant findings of this review are summarized in this chapter, based on them important research gaps were identified, and the objectives were formulated and stated in Section 1.2 and 1.3 of this thesis.

This chapter mainly presented the necessary background for the activity-centric paradigm. The overwhelming evolution of the artifact-centric paradigm over the last decade was also discussed, which showed how this paradigm has received attention from various academic and industrial researchers and how it has gained popularity as one of the mainstream approaches in BPM. In addition, existing works relating to the three research questions to be addressed in this thesis were discussed. It was also demonstrated that the existing literature lacks efficient approaches to address these issues, which motivated this further research on the three key issues. Therefore, this thesis proposes to resolve these issues by presenting some approaches that efficiently work for the given problems.

Chapter 3

Synthesizing Artifact Lifecycles from Activity-Centric Process Models

The artifact-centric approach to business process modeling has received considerable attention for elevating data logic to the same level as the process flow logic. The objective of artifact-centric modeling is to provide an intuitive and flexible representation to business people which is used to analyze, manage and control their business operations. With the growing significance of this modeling approach, the transformation of traditional activity-centric process models into artifact-centric process models has emerged as an important research question. The proposed transformation mainly aims to synthesize or generate indispensable units of artifacts such as their lifecycles and their synchronization dependencies.

As described in Chapter 2, although several approaches to the defined objective exist, an automated approach with a holistic view on the flow of artifacts and their synchronization dependencies is still needed to address the limitations of existing approaches and facilitate the proposed transformation. Therefore, in this chapter, a tree-based synthesis approach to automatically generate the lifecycles of artifacts from the activity-centric process models that contain a set of artifacts and states is

presented. Specifically, algorithms are proposed to construct process trees reflecting the hierarchical structure of activity-centric process models, to generate the lifecycles of artifacts by traversing through the process trees and to refine and synchronize these lifecycle models to show the interrelations between the corresponding artifacts. The feasibility of the proposed approach will be demonstrated using a case study and the applicability and performance will be evaluated through in-lab experiments using a process model collection from BPM Academic Initiative (BPMAI) (Kunze et al., 2012).

The remaining chapter is organized into seven sections. Section 3.1 introduces a motivating example based on a customer order processing scenario. Section 3.2 formulates the problem statement and defines some of the key notions used in deriving a solution for the proposed problem. Section 3.3 presents an overview of the proposed synthesis approach. Section 3.4 presents algorithms for each step of the synthesis approach. Section 3.5 demonstrates a case study, the implementation and performance evaluation of the proposed approach. Section 3.6 discusses and reviews the related work. Section 3.7 summarizes the contributions of this chapter.

3.1 Motivating Example

In this section, a motivating example on *Customer Order Processing* to demonstrate the proposed approach is presented. Figure 3.1 shows the order process that is represented in Business Process Modeling Notation (BPMN) with artifact data flow annotations. This process model ¹ is adopted from (Meyer & Weske, 2013), where the BPMN constructs such as *events*, *activities*, *gateways (XOR, AND)*, *sequence flows*, *artifacts* and their *associations* are used to capture the processing steps of customer order.

The *event*, *activity* and *gateway* elements of the process model are classified as *nodes*, as they represent nodes on the process network, while *sequence flows* represent

¹The terms *process* and *process model* are interchangeably in the rest of this thesis

edges of that process network. Here, a gateway that contains multiple outgoing sequence flows is called a *split* (or *open*) type of gateway and a gateway that contains multiple incoming sequence flows is called a *merge* (or *close*) type of gateway. The nodes can be further classified into *atomic*, that includes *event* and *activity* nodes, and *non-atomic*, such as *gateway* nodes as they have an internal structure.

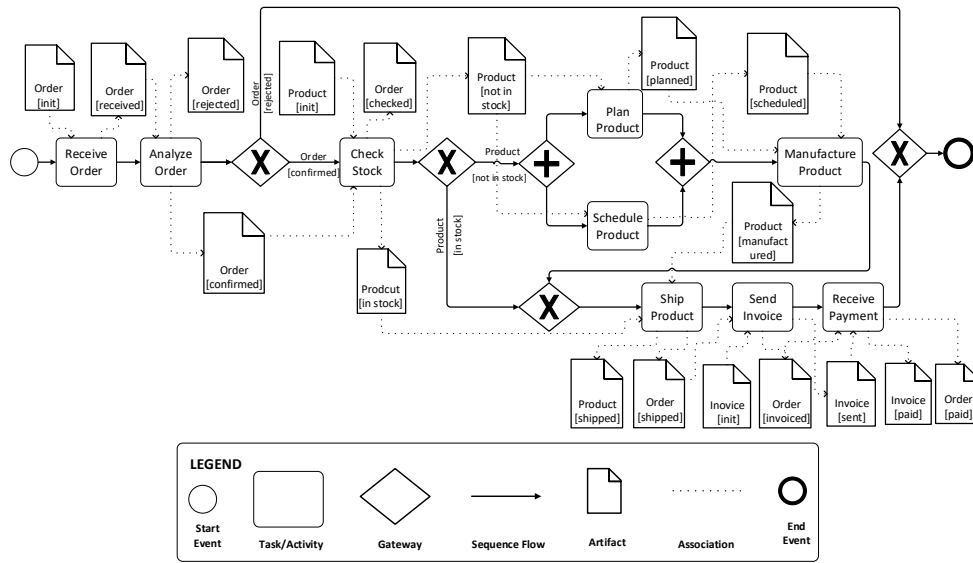


Figure 3.1: Customer Order Process

As illustrated in Figure 3.1, order processing is initiated upon receiving a customer request and completed with the order's delivery or rejection. *Order*, *Product* and *Invoice* are the three artifacts involved in fulfilling the customer request. As annotated in the process, every artifact has a *name* and a *state* that the artifact is currently residing in. The states can be changed when the activities invoke the associated artifacts. For example, after receiving a customer request, the state of the *Order* artifact is changed from *init* to *received* by the *ReceiveOrder* activity. Next, the *AnalyzeOrder* activity places the *Order* artifact in either the *confirmed* state or in the *rejected* state. In case of *confirmed* state, the *Order* artifact undergoes a set of processing steps from *CheckStock* to *ReceivePayment*, until it reaches its final state of *paid*; otherwise it enters the *rejected* state and the process terminates. Similarly, the artifacts *Product* and *Invoice* pass

through several activities and states to complete their processing.

Regarding artifact data flow, it is worth mentioning that an activity node can have multiple input and output artifacts, as illustrated in the above process. For example, the *SendInvoice* activity has two input artifacts *Order* and *Invoice*, and also outputs these two artifacts in different states. Likewise, to represent the case that an activity might output an artifact in different states, for example the *AnalyzeOrder* that outputs *Order* in either the *rejected* or *confirmed* state. For these cases, multiple artifact annotations, i.e., having the same name but in different states are used, as seen in Figure 3.1.

It is clear from the above two cases that a *split* type of gateway succeeds an activity node that produces multiple states for an artifact, where each of these states must be associated with a different branch of the gateway. Similarly, a *merge* type of gateway precedes an activity node that receives an artifact with multiple states as input from each branch of the gateway. For example, the *AND-Merge* gateway precedes the *ManufactureProduct* activity that takes *planned* and *scheduled* states of the *Product* artifact as input from different branches of the *AND-Split* gateway.

Intuitively, the artifact data flow annotation in the given process model enables the generation of the *lifecycle* or *state transition* of every artifact, whose states and their dependencies are specified in such an artifact-annotated process model (or AAPM). For example, the lifecycle of an *Order* artifact starts with *Init* state, then passes through the states *Received*, *Confirmed* (or *Rejected*), *Checked*, *Shipped*, *Sent* and finally ends in the *Paid* state.

3.2 Problem Statement and Definitions

In this section, the research problem targeted in this chapter is formally stated and all the key notions used in deriving a solution for the given problem are defined. These notions include: *Process Model*, *Artifact-Annotated Process Model*, *Process Tree*, *Artifact*

Lifecycle and Synchronized Artifact Lifecycle.

PROBLEM (ARTIFACT LIFECYCLE SYNTHESIS). Given an activity-centric process model annotated with data flow of artifacts and their states, the lifecycle of every artifact and their synchronization dependencies from this process model must be synthesized.

The scope of this chapter is to generate artifact lifecycles based on the control flow and data flow of standalone activity-centric process models. Therefore, only a subset of relevant BPMN elements is considered to model the motivating example, as presented in Figure 3.1, instead of the entire BPMN modeling constructs. For example, elements such as *swimlanes* and *message flows* are excluded here, as they are used for representing the inter-organizational business processes and their interactions. Therefore, a standalone activity-centric process model is defined as follows:

Definition 1. *Process Model:* A process model denoted with $\pi_0 = (E, A, G, F)$, where

- E is a finite set of events. Every event has a type in $E.Type = \{\text{Start}, \text{End}\}$;
- A is a finite set of activities;
- G is a finite set of gateways. Every gateway has a type in $G.Type = \{\text{XOR}, \text{AND}\}$;

and G^{open} is used to represent opening (split) gateways and G^{close} is used to represent closing (merge) gateways;

- $F \subseteq (E \cup A \cup G) \times (E \cup A \cup G)$ is the set of sequence flow relations among events, activities and gateways.

The process model is assumed to be block-structured. A block-structured process model, such as the given motivating example, contains exactly one start and one end node, all the gateway branches are properly nested, and each further node is on a path from the start to the end node (Backus et al., 1960; Kiepuszewski, Ter Hofstede & Bussler, 2000). As annotations on artifacts and data flows are optional in majority of activity-centric process models (Meyer & Weske, September, 2013), to differentiate

these models, from now on, the proposed (customer order) process model is named the Artifact-Annotated Process Model (AAPM), which is defined as follows:

Definition 2. *AAPM (Artifact-Annotated Process Model):* An AAPM Π is a superset of process model $\pi_0 = \{E, A, G, F\}$: $\Pi = \pi_0 \cup \{D, S, I, O\}$, where

- D is a finite set of artifacts;
- S is a finite set of states;
- $I = D \times S \times A$ is the input relation between an artifact at a specific state with an activity. For example, as seen in Figure 3.1, $(Order, Init, ReceiveOrder) \in I$, meaning that the artifact *Order* at state *Init* is an input to the *ReceiveOrder* activity node. For the convenience of reading, the form $ReceiveOrder.I = \{Order.init\}$ is used to represent an input relation;

- Similarly, $O = A \times D \times S$ is the output relation between an activity and an artifact at a specific state. For example, as seen in Figure 3.1, $(ReceiveOrder, Order, Received) \in O$, meaning that the artifact *Order* at state *Received* is an output of *ReceiveOrder*, and it can also be written as $ReceiveOrder.O = \{Order.Received\}$.

From the above definition, it can be seen that I and O define the data flow of artifacts over the original activity-centric process model.

Definition 3. *Process Tree:* A tree is recursively defined as a node (called the *root*) connected to the roots of other trees. A process tree is a tree with $Node = \{SEQ\} \cup \pi.E \cup \pi.A \cup \pi.G$.

A node on the process tree can be an *event*, an *activity*, or a *gateway* of an AAPM, or a special node called *SEQ*, which means its child nodes are in *sequential order*. Naturally, on a process tree the corresponding input and output artifacts and states can be attached to an activity node as its properties.

Statecharts are used to represent the lifecycle models of artifacts. Statecharts are well-known for illustrating entity behaviour as a set of *states* (simple or composite)

and their *transitions*; they also have the notion of *orthogonal regions* for specifying concurrent states. An artifact lifecycle is defined based on the formal semantics of statecharts proposed in (Mikk, Lakhnechi & Siegel, December, 1997; O. OMG, 2007).

Definition 4. *Artifact Lifecycle*: The lifecycle of artifact denoted with $\mathcal{L}_d = (d.S, \rho, T, \lambda, OG)$, where

- $d.S$ is the finite set of states of an artifact d ;
- $\rho = \{initial, choice, junction, fork, join\}$ is a finite set of pseudostates;
- $T \subseteq S \times S$ is a finite set of state transitions;
- $\lambda \subseteq (\rho \cup S) \times (S \cup \rho)$ is the set of transition relations among states and pseudostates;
- OG is the set of orthogonal regions. Every region is a finite set of states β , pseudostates $\bar{\rho}$ and transition relations $\bar{\lambda}$, where $\beta \subset d.S$, $\bar{\rho} \subset \rho$ and $\bar{\lambda} \subset \lambda$. Two states s_1, s_2 are said to be *orthogonal*, iff $\exists og^1, og^2 \in OG$, where $og^1 \neq og^2 \wedge s_1 \in og^1 \wedge s_2 \in og^2$.

Definition 5. *Synchronized Artifact Lifecycle*: A synchronized artifact lifecycle denoted with $S_{\mathcal{L}} = (\mathcal{L}_D, \Psi)$, where

- \mathcal{L}_D is the finite set of artifact lifecycles;
- Ψ is the finite set of synchronization edges between the artifact lifecycles. Every edge $\psi = (s_i, s_j)$, where $s_i \in \mathcal{L}_{d_i} \wedge s_j \in \mathcal{L}_{d_j}$.

3.3 The Synthesis Approach

The *synthesis* is defined as the derivation of synchronized artifact lifecycles from the tree representation of an artifact-annotated process model (AAPM). The overview of synthesis approach is presented in Figure 3.2, where the three steps in resolving the synthesis problem are depicted with the inputs they utilize and the outputs they produce after execution. This section briefly elaborates each of these steps, while the algorithms for these steps are presented and explained in detail in the next section.

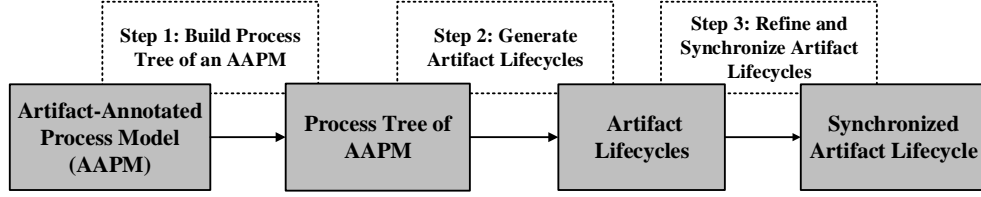


Figure 3.2: Synthesis Approach

STEP 1(Building a Process Tree). Given an artifact-annotated process model (AAPM) Π , the corresponding process tree *Node* is constructed by tracing through the sequence flow nodes of this process model.

In this step, a tree representation of the given AAPM is built. The resulting process tree hierarchically encloses the nodes of AAPM by following their direct sequence flow relations. Mainly, the tree building is initiated at the *start* node and is terminated at the *end* node of the process model. All the nodes, including the *start* and *end* nodes of AAPM, are appended to the root node (SEQ) of the resulting process tree. Prior to appending each process node, its type is checked as either *atomic* or *non-atomic*. When compared to the atomic nodes, such as *events* and *activities*, the non-atomic *gateway* nodes are tackled differently in order to capture their internal structure. Therefore, unlike the atomic nodes, for every *gateway*, the build is repeated to append all its branch nodes to the process tree.

STEP 2 (Generating Artifact Lifecycles). Given an artifact-annotated process model (AAPM) Π , the lifecycle of every associated artifact \mathcal{L}_d is generated by traversing through the tree representation of this process model.

In this step, every node of the process tree built in Step 1 is searched for states to generate the lifecycles of artifacts. Mainly, the input and output of each activity node in the process tree is examined to retrieve the states that correspond to an artifact, which are then organized to generate its lifecycle.

STEP 3 (Refining and Synchronizing Artifact Lifecycles). Given an artifact lifecycle

\mathcal{L}_d with some irrelevant elements (duplicate states and empty gateways) , the refined artifact lifecycle is obtained by removing these elements and is synchronized with other artifact lifecycles by considering their concurrent state changes in AAPM.

In this step, the artifact lifecycles resulting from Step 2 are refined in order to remove irrelevant elements. Mainly, the duplicate states that have a direct sequence flow relation and empty gateway nodes (if any) are removed from the generated artifact lifecycles. The refined lifecycles are then synchronized to show the interactions between the corresponding artifacts by considering the concurrent state changes caused by the activity nodes in the given process model.

3.4 Algorithms

In this section, algorithms for each step of the synthesis approach are presented. Prior to defining these algorithms, an updated version of the motivating example (or AAPM) is presented in Figure 3.3, where empty tasks (Phi or Φ) (Natschläger, August, 2011) are inserted at every direct sequence flow that connects a *split* type of gateway with its *merge*. An artifact with its state associated to the corresponding gateway branch is annotated as an input and output to these empty tasks. Here, the empty tasks are useful to capture the dependency (*exclusive* or *parallel*) of artifact states associated with each gateway node of AAPM in the resulting lifecycles. In the following sections, the updated AAPM, as presented in Figure 3.3, is utilized to demonstrate the key aspects of proposed algorithms.

3.4.1 Building a Process Tree

Algorithm 3.1 defines a procedure to build a tree representation of AAPM. The resulting process tree encloses the nodes of AAPM in a hierarchical structure. As mentioned in the previous section, a sequential node, SEQ, is used to represent the root node of the

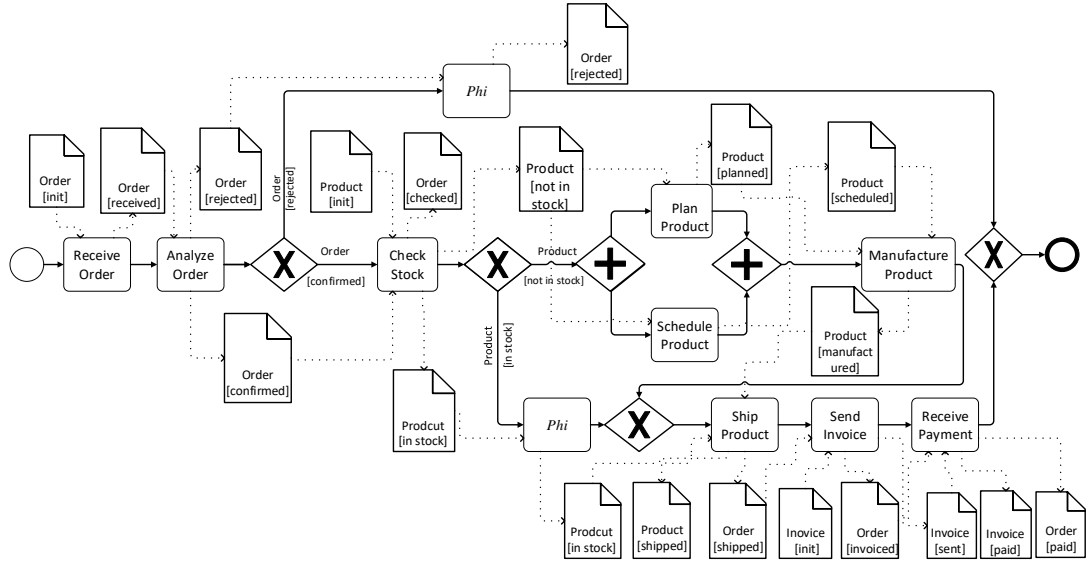


Figure 3.3: Customer Order Process (updated AAPM)

process tree. To this SEQ root, the algorithm appends all the (first-level) direct sequence flow nodes of AAPM as child nodes. For example, the nodes *Start*, *ReceiveOrder* and *AnalyzeOrder* in Figure 3.3 can be appended as child nodes to SEQ, as they have a direct sequence relation. While *start* and *end* nodes are always connected in a sequence relation in any process, it is feasible to use SEQ to represent the parent of these same level process nodes.

The *BuildTree()* recursive function defined in Algorithm 3.1 takes as input, an empty tree root *r* (SEQ) and the node set of AAPM. As stated in the algorithm, starting at the first node (*start*) of AAPM, each process node (*v*) is examined for its type and then is appended accordingly to the tree. Therefore, based on the defined conditions (in line 3, 5 and 7), a node must be atomic (*event* or *activity*) or non-atomic (*gateway* (split)), containing an internal structure. For every *activity*, the algorithm also preserves the corresponding input and output artifact annotation in the process tree as their properties. Here, the algorithm does not consider the no operation (NOP) nodes, such as the close gateway (merge) nodes, as they have no effect on the artifacts and their states.

Algorithm 3.1: Building a Process Tree of AAPM

Input : Process tree *Node* with empty root *r* (SEQ) and node set of AAPM Π

Output : Complete process tree *Node* of AAPM Π

```

1 Function BuildTree (r : Node , v :  $\Pi$ ) :
2   if v ≠ null then
3     if v ∉  $G^{Close}$  then
4       r.AppendChild(v)
5       if v ∈ A then
6         AnnotateIO(v)
7       else if v ∈  $G^{Open}$  then
8         foreach c ← GetSeqNext(v) do
9           vc ← SEQ
10          v.AppendChild(vc)
11          BuildTree(vc, c)
12        end
13        v ← GetCloseNode(v)
14      end
15      v ← GetSeqNext(v)
16      BuildTree(r, v)
17    end
18  end
19 End Function

```

The *AppendChild()* function in the algorithm is used to append a node to its parent node in the process tree. While every *event* and *activity* node is appended directly to their corresponding parent node in the tree, for every *gateway* its open (split) node is appended first and then the *BuildTree()* function is recursively invoked to append its branch nodes. The *GetSeqNext()* function in the algorithm returns the direct successor (next sequence flow node) of a process node. Likewise, the *GetCloseNode()* function is used to obtain the close (merge) gateway node of a gateway.

Figure 3.4 presents the process tree of AAPM, built using Algorithm 3.1. As shown in the figure, the first-level sequence flow nodes of AAPM: *Start*, *ReceiveOrder*,

AnalyzeOrder, *XOR-Split* gateway and *End* appear as the first-level child nodes of root SEQ, as they hold a direct sequence flow relation. For every activity node, the *AnnotateIO()* function in the algorithm annotates the associated input and output artifacts with states, which can also be seen from the process tree.

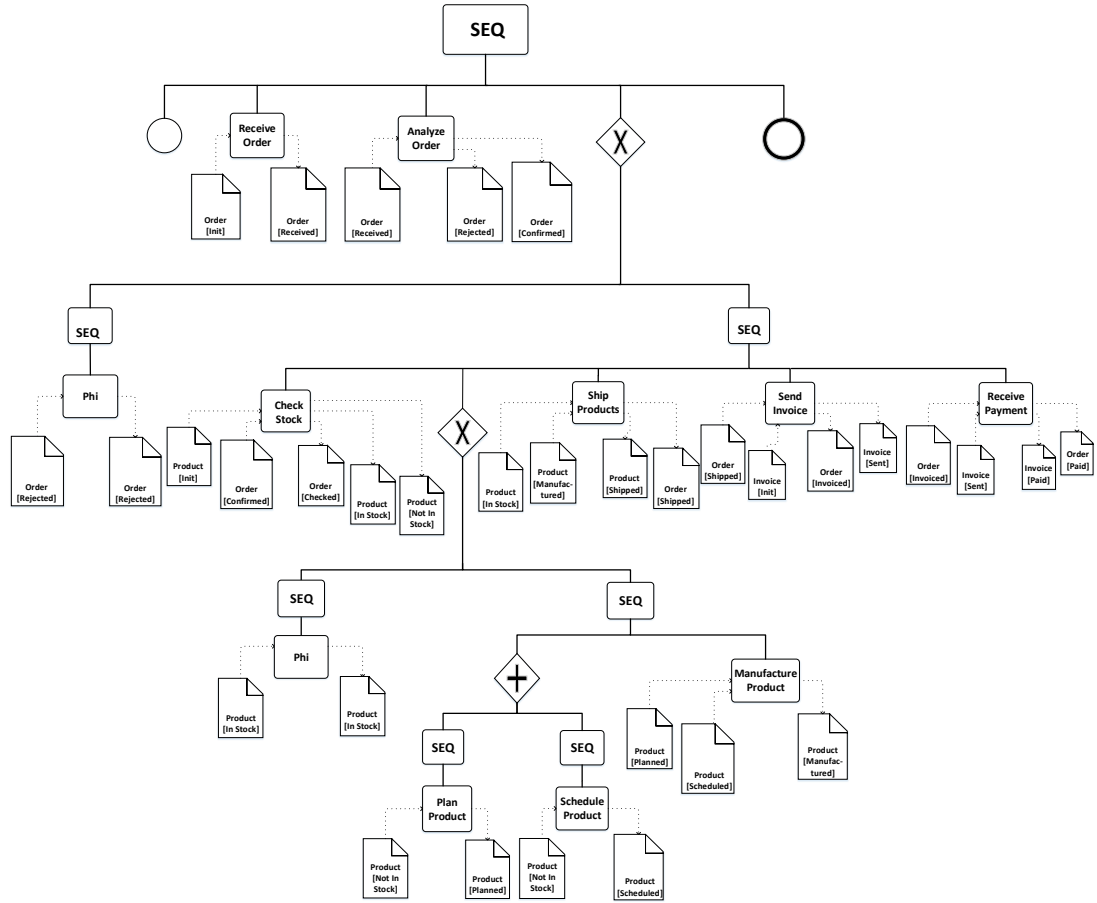


Figure 3.4: Process Tree of AAPM

Specifically, Algorithm 3.1 follows a step-by-step procedure to build a process tree, where after appending a node its next sequence flow node is obtained using the function *GetSeqNext()* and is appended in a next sibling relation. Similarly, for every sequence flow (branch) node of an open gateway (split) node, the algorithm appends a new SEQ node (in line 8 and 9), as its child node. The node v_c in the algorithm denotes this new SEQ node to which all the nodes flowing on the corresponding gateway branch are

appended. Here, the *BuildTree()* function is recursively invoked to append these branch nodes as the next-level child nodes of the new SEQ node.

Figure 3.3 presents an example, where the second *XOR-Split* gateway node of AAPM has two sequence flow nodes: *AND-Split* gateway and *Phi* activity. Therefore, for each of these nodes, a new SEQ node is appended as a child to the corresponding *XOR-Split* gateway node in the process tree. The nodes *AND-Split* and *Phi* are then appended as next-level child nodes of these two new SEQ nodes, as shown in Figure 3.4. As mentioned above, the close gateway (*XOR-Merge*) node is not appended to the process tree.

In this manner, starting at the first (start) node of AAPM, the build is continued until the last node (end) is reached. The process tree resulting from Algorithm 3.1 is used in the following step to generate the lifecycle for every artifact of AAPM.

3.4.2 Generating Artifact Lifecycles

Algorithm 3.2 defines a *GenerateLifecycle()* recursive function that generates lifecycle for every artifact present in AAPM. The input to this function is a process tree *Node* that is built using Algorithm 3.1 and an artifact *d*, while the output is an artifact lifecycle \mathcal{L}_d . According to Algorithm 3.2, for each artifact of AAPM, the recursive function traverses every node of process tree to retrieve the states that are used to generate its lifecycle. Here, the algorithm follows the *Pre-order Depth First Search (DFS)*, where, starting at the left most node (or sub tree) of root, every next node is searched until the right most node (or sub tree) of the root is reached in a depth first manner, meaning that all of the deeper level nodes are searched.

As shown in the process tree presented in Figure 3.4, all the *gateway* and *SEQ* nodes are *non-leaf* nodes, as they have at least one child node, whereas the *event* and *activity* nodes are *leaf* nodes. Similar to Algorithm 3.1, this algorithm also tackles the

non-leaf and *leaf* nodes differently (see line 3, 5 and 9). Therefore, as defined, the *start* and *end* nodes are directly added to the corresponding artifact lifecycle. Whereas, for every *activity* (lines 5-8), its input and output artifacts are searched to retrieve the required states and, as defined in the algorithm (in line 6 and 7), a state is added to the corresponding artifact lifecycle only when there is one copy of that artifact associated with the input or output of that activity.

Algorithm 3.2: Generating an Artifact Lifecycle

Input : Process tree *Node* and artifact *d*

Output : Lifecycle of artifact \mathcal{L}_d

```

1 Function GenerateLifecycle(v : Node, d :  $\Pi.D$ ) :
2   foreach v  $\in$  Node do
3     if v  $\in$  E then
4       |  $\mathcal{L}_d.AddInSeq(v)$ 
5     else if v  $\in$  A then
6       | if  $|v.I.D \cap \{d\}| = 1$  or  $|v.O.D \cap \{d\}| = 1$  then
7         |  $\mathcal{L}_d.AddInSeq(d.s)$ 
8       | end
9     else if v  $\in$   $G^{Open}$  then
10      |  $\mathcal{L}_d.AddInSeq(G^{Open})$ 
11      | foreach SEQ  $\in$  v do
12        |  $G^{Open}.AddBranch()$ 
13        |  $GenerateLifecycle(SEQ)$ 
14      | end
15      |  $\mathcal{L}_d.AddInSeq(G^{Close})$ 
16    end
17  end
18 End Function

```

The *AddInSeq()* function in the algorithm is used to add a node, such as *event* or *gateway*, and also an artifact state to the corresponding artifact lifecycle. Therefore, as the name implies, a node or a state can be added in a sequence relation to the previously inserted element (node or state) in the lifecycle. As defined in the algorithm (lines

9-16), while dealing with a *gateway* node, an empty open gateway (split) node is first inserted into the artifact lifecycle in order to show the dependency between the states that belong to different branches of the gateway. The *GenerateLifecycle()* function is then called recursively to each of its SEQ child nodes, which traverses through their deeper level nodes and adds the associated states to the lifecycle. After completing all the SEQ child nodes, a close gateway (merge) node is added in the lifecycle that merges all the branches of the corresponding open gateway.

Figure 3.5 presents the lifecycle of *Product* artifact generated using Algorithm 3.2 from the process tree presented in Figure 3.4. As described above, starting at the left most node (Start) of SEQ (root), the algorithm searches through all its next nodes *ReceiveOrder*, *AnalyzeOrder*, *XOR-Split* and *End* to find the states $\{Init, In Stock, Not In Stock, Planned, Scheduled, Manufactured, Shipped\}$ of *Product* artifact and adds them to its (empty) lifecycle, preserving their dependencies.

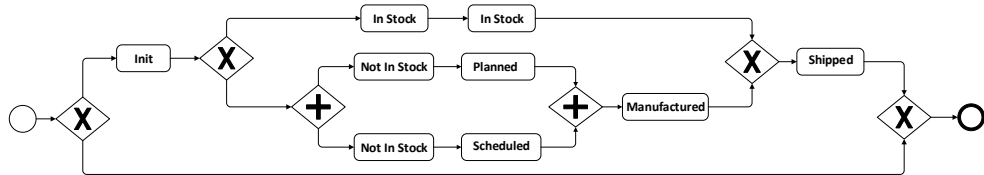


Figure 3.5: Lifecycle of Product artifact

In detail, to generate a *Product* artifact lifecycle, the *start* node is added first and then its following nodes are traversed for the states of this artifact. While the nodes *ReceiveOrder* and *AnalyzeOrder* do not contain states related to *Product* artifact, the child nodes of *XOR-Split* are traversed next. Therefore, as described above, before calling a recursive function that searches the child nodes of *XOR-Split*, an empty split node of this gateway is added in the lifecycle, as shown in Figure 3.5, and then its child nodes are traversed. Here, as only the second (right) SEQ node of the *XOR-Split* gateway has nodes (*activities*) that contain states belonging to the *Product* artifact, the other branch of this gateway node in the lifecycle appears empty. In this manner, each

activity node in the process tree is searched and the states are arranged based on their dependencies to generate the artifact lifecycles of AAPM.

Algorithm 3.3: Generating all the Artifact Lifecycles of AAPM

Input : Process tree $Node$ and set of empty artifact lifecycles \mathcal{L}_D

Output : Complete lifecycles of artifacts \mathcal{L}_D

```

1 Function GenerateLifecycle( $Node$ ,  $\mathcal{L}_D$ ):
2   foreach  $v \in Node$  do
3     if  $v \in E$  then
4       foreach  $L_d \in \mathcal{L}_D$  do
5          $\mathcal{L}_d.AddInSeq(v)$ 
6       end
7     else if  $v \in A$  then
8       if  $|v.I.D \cap \{d\}| = 1$  or  $|v.O.D \cap \{d\}| = 1$  then
9          $\mathcal{L}_d.AddInSeq(d.s)$ 
10      end
11     else if  $v \in G^{Open}$  then
12       foreach  $L_d \in \mathcal{L}_D$  do
13         // block (starts)
14          $\mathcal{L}_d.AddInSeq(G^{Open})$ 
15       end
16       foreach  $SEQ \in v$  do
17          $GenerateLifecycle(SEQ, \mathcal{L}_D)$ 
18       end
19       foreach  $L_d \in \mathcal{L}_D$  do
20          $\mathcal{L}_d.AddInSeq(G^{Close})$ 
21       end
22       // block (ends)
23     end
24   end
25 End Function

```

In Algorithm 3.3, an alternative procedure to generate the lifecycles of artifacts is presented. In comparison to Algorithm 3.2, which requires one complete traversal of the process tree to generate one artifact lifecycle, Algorithm 3.3 generates all the artifact

lifecycles in one traversal of the process tree. This goal is achieved mainly by retrieving the states and adding them to the corresponding artifact lifecycles simultaneously.

Algorithm 3.3 takes a process tree *Node* and a set of empty artifact lifecycles \mathcal{L}_D as input to generate complete artifact lifecycles. Similar to Algorithm 3.2, this algorithm also initiates at the *start* node of the process tree, and checks every node for its type, such as *event* or *activity* or *gateway* (see line 3, 7 and 11). Here, the *event* and *gateway* nodes are added to every artifact lifecycle simultaneously. Similarly, for activities, their inputs and outputs are checked for artifacts and the states associated with each of them are added to the corresponding lifecycles at the same time. The block (lines 13-22) enclosed in Algorithm 3.3 is used to preserve the dependencies between the artifact states from the process tree to the resulting lifecycles.

As demonstrated above, this algorithm requires one traversal of the process tree to generate all the artifact lifecycles. Therefore, it is clear that Algorithm 3.3 is more efficient than Algorithm 3.2, as it can generate all the artifact lifecycles in only one traversal of the process tree.

3.4.3 Refining and Synchronizing Artifact Lifecycles

The artifact lifecycles generated using Algorithm 3.2 and 3.3 may result in some duplicate states and empty gateways (i.e., the gateways that do not contain any states on their branches). This is mainly due to the activity nodes that retain their own set of input and output annotation (artifacts and states) from the process model in the corresponding process tree. For example, the *Order[Received]* appears twice, as an output of *ReceiveOrder* and also as an input of its next node *AnalyzeOrder*, in the process tree, presented in Figure 3.4. Therefore, the resulting *Order* artifact lifecycle contains two consecutive *Received* states, as shown in Figure 3.6. Similarly, every *phi* activity node in the process tree contains the same input and output artifact annotation,

which also leads to duplication in the corresponding lifecycles.

Algorithm 3.4: Refining an Artifact Lifecycle

Input : Generated artifact lifecycle \mathcal{L}_d

Output : Refined artifact lifecycle \mathcal{L}_d

```

1 Function RefineLifecycle( $v : \mathcal{L}_d$ ):
2   if  $v \neq null$  then
3     if  $v \notin G$  then
4        $v_{next} \leftarrow v.GetSeqNext(v)$ 
5       if  $v_{next} = v$  then
6          $\mathcal{L}_d.Remove(v)$ 
7       end
8     else
9       if  $v \in G^{Open}$  then
10        foreach  $v_{next} \leftarrow GetSeqNext(v)$  do
11          if  $v_{next} = G^{Close}$  then
12             $count \leftarrow count + 1$ 
13          else
14             $RefineLifeCycle(v_{next})$ 
15          end
16        end
17        if  $count = |GetAllSequenceFlows(v)|$  then
18           $\mathcal{L}_d.Remove(v)$ 
19        end
20      end
21    end
22     $v \leftarrow v.GetSeqNext(v)$ 
23     $RefineLifeCycle(v)$ 
24  end
25 End Function
  
```

In addition, empty gateways that have no states on their branches may also appear in the generated lifecycles if none of the child nodes of a gateway node in the process tree contain states that belong to the corresponding artifacts. Therefore, refinement is suggested as a post-processing step to resolve each of these possibilities, where

the empty gateways and one of the two identical states that are in direct sequence relation must be removed. To obtain the fine-grained synchronized artifact lifecycles, it is important to resolve these inconsistencies before the generated lifecycles are synchronized. Therefore, a procedure is stated in Algorithm 3.4 that generates the refined artifact lifecycles by removing all the irrelevant elements.

According to Algorithm 3.4 (lines 5-7), in an artifact lifecycle, a state that is identical to its succeeding state is removed using the *Remove()* function if they have a direct sequence relation. Similarly, a gateway (open) is removed (lines 8-21) if all its outgoing sequence flows are directing to its close gateway node. Here, a counter variable (*count*) is used to enumerate these outgoing sequence flows, based on which a gateway is removed if the value of this variable is equal to the number of branches that a gateway has. After the refinement step, the resulting lifecycles can be mapped to statechart representations, as shown in Figure 3.8.

The lifecycle of the *Order* artifact, presented in Figure 3.6 is reconsidered, as it contains some irrelevant elements as described above. The refined lifecycle of the *Order* artifact is given in Figure 3.7, where the duplicates of the states *Received*, *Rejected*, *Shipped* and *Invoiced* are removed. In addition, the empty gateways, such as the *parallel* gateway and the *exclusive* gateway (that encloses the parallel gateway) are also removed from the original lifecycle of the *Order* artifact.

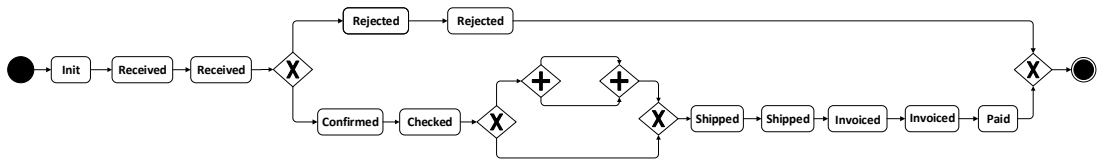


Figure 3.6: Generated Lifecycle of Order artifact

As mentioned above, the refined lifecycles can be mapped to statecharts. For example, in Figure 3.8 the refined *Product* artifact lifecycle is mapped to a statechart, where all the *XOR-Split* and *XOR-Merge* gateway nodes are replaced with *choice*

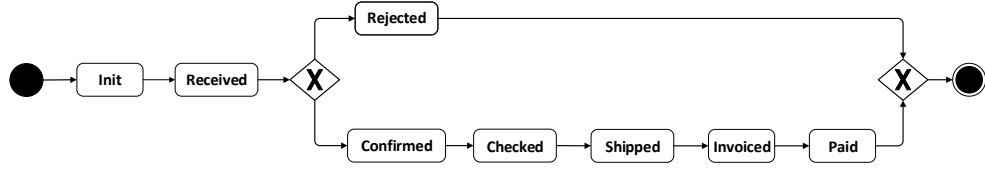


Figure 3.7: Refined Lifecycle of Order artifact

and *junction* pseudo states and all the *AND-Split* and *AND-Merge* gateway nodes are replaced with *fork* and *join* pseudo states. Also, the parallel states, such as *Planned* and *Scheduled*, are separated from the orthogonal regions of the *fork* pseudo state to show the concurrency between them.

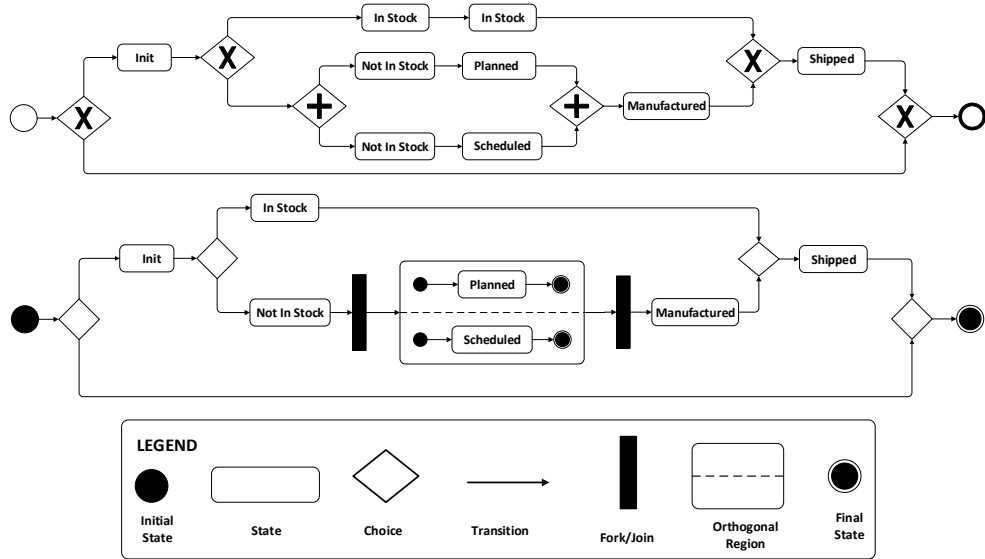


Figure 3.8: Statechart representation of Product artifact lifecycle

Next, the refined lifecycles are synchronized to show the relationship between the corresponding artifacts. The synchronization process is simple, where a set of artifact lifecycles are synchronized by connecting their states that changed simultaneously. Clearly, the states of any two artifacts are synchronized if they both are associated with the output of an activity node. Therefore, according to the following pseudo code the lifecycles \mathcal{L}_{d_i} , \mathcal{L}_{d_j} of two artifacts d_i and d_j are synchronized at states that belong to the output set (O) of the same activity v . The *Synchronize()* function is for adding a

synchronization edge between the states of any two artifact lifecycles.

foreach $v \in A$ **do**

if $(d_i, d_j) \in v.O$ **and** $(s_i \in \mathcal{L}_{d_i}$ **and** $s_j \in \mathcal{L}_{d_j})$ **then**
 $Synchronize(s_i, s_j)$

end

end for

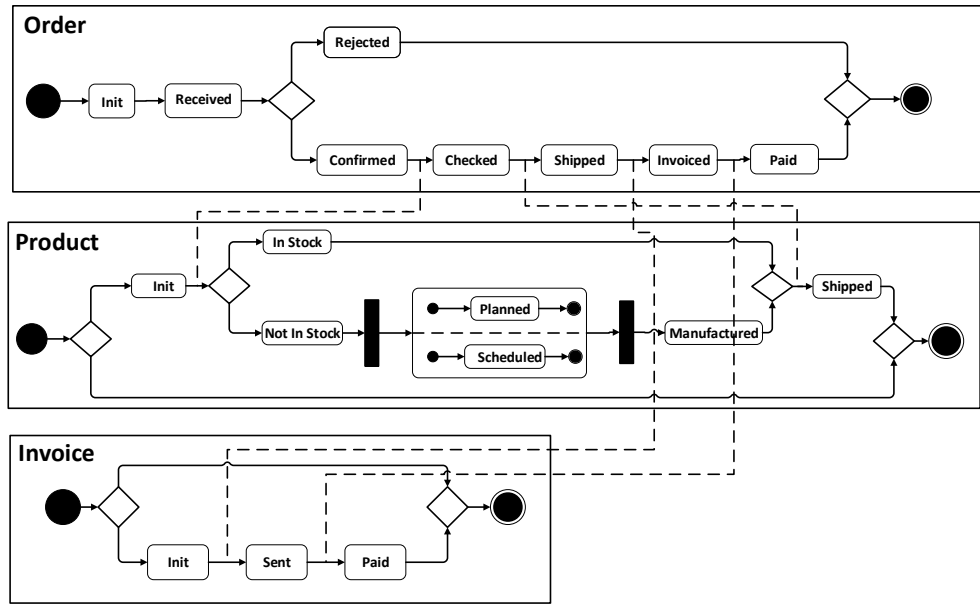


Figure 3.9: Synchronized artifact lifecycles of customer order process

The synchronized artifact lifecycles of the motivating example are presented in Figure 3.9. The figure mainly depicts the lifecycles of the *Order*, *Product* and *Invoice* artifacts that capture their end-to-end processing in a set of states with dotted lines representing their synchronization. In this figure, the first synchronization line indicates that the states *Checked*, *In stock* and *Not In Stock* belong to the output of the same activity node, *CheckStock*. Similarly, the *Shipped* state of *Order* and *Product* artifacts and the *Invoiced*, *Sent* and *Paid* states of *Order* and *Invoice* artifacts are also synchronized.

3.5 Evaluation

In this section, a case study is demonstrated to show the feasibility of the proposed approach. The implementation and performance evaluation of the proposed algorithms is also discussed.

3.5.1 Case Study

In this section, a *recruitment process model* is used to analyze the feasibility of synthesis approach. This process model represents the steps in recruiting for a job position, and is chosen from the BPMN process model collection, obtained from the repository of the BPM Academic Initiative (BPMAI) (Kunze et al., 2012). BPMAI provides process model collections in different modeling languages such as *BPMN*, *EPC* and *Petri Net* for the purposes of teaching and empirical research.

The model collection obtained from BPMAI has different types of BPMN process models, including: *standard*, *choreography* and *conversation*. These process models vary in size, connectivity, complexity as well as domain. The motivating example presented in Figure 3.1 is a standard type of process model, whereas the *choreography* and *conversation* represent the interacting processes. While there are thousands of standard process models in the model collection, it was challenging to choose an appropriate case for this study. To find an appropriate case, the process models that comply with the basic requirements specified in the previous section and closely align to Definition 1 were first identified. Then, the process models were sorted based on their connectivity and size (number of events, activities and gateways).

Figure 3.10 presents a histogram to give more statistical information on the process collection chosen for this study. As illustrated by the histogram, the process count decreases with the increase in process size (element count), which means there are only a limited number of large and complex processes available for this study. Therefore,

a moderately complex case was chosen that represents a position recruitment process from the highest range 120-140 presented in the histogram.

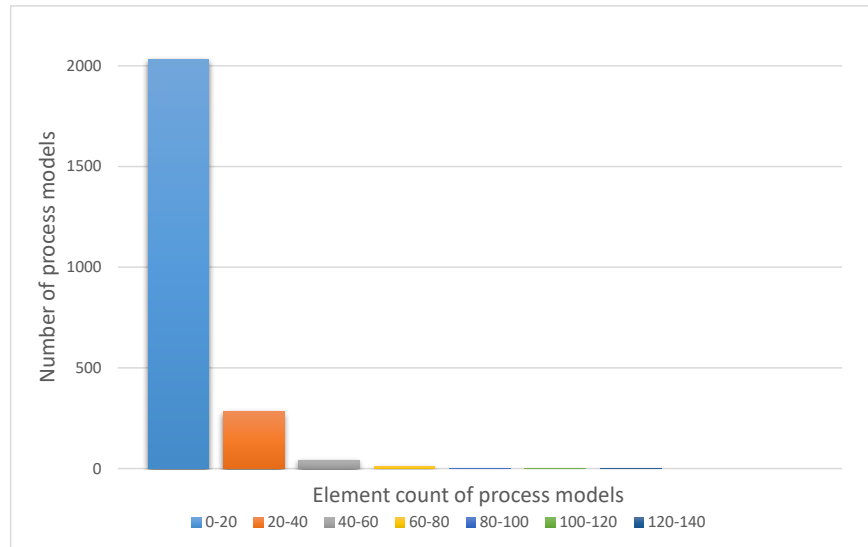


Figure 3.10: Histogram of Sorted BPMN Model Collection

Figure 3.11 shows, where the recruitment process model describes a set of activities in fulfilling a job position, mainly these include the processing steps of the key artifact *Position* of this process. For ease of understanding, this section only presents an excerpt of the recruitment process model, while the complete process model can be found in the Appendix, in Figure A.1.

It is worth mentioning that some of the structural inconsistencies of this process model were resolved before utilizing this for the proposed study. Mainly, this process model initially had no artifact data flows, as the majority of process models in the BPMN model collection are control-flow specific. Therefore, a label analysis based data extraction tool (Meyer & Weske, September, 2013) was used to enrich the *recruitment process model* with artifacts and states. The extraction tool enriched this process model with 46 artifacts and 99 states.

The proposed algorithms were then applied to the enriched model. In order to generate the lifecycle of every artifact, the algorithms first construct a tree model of this

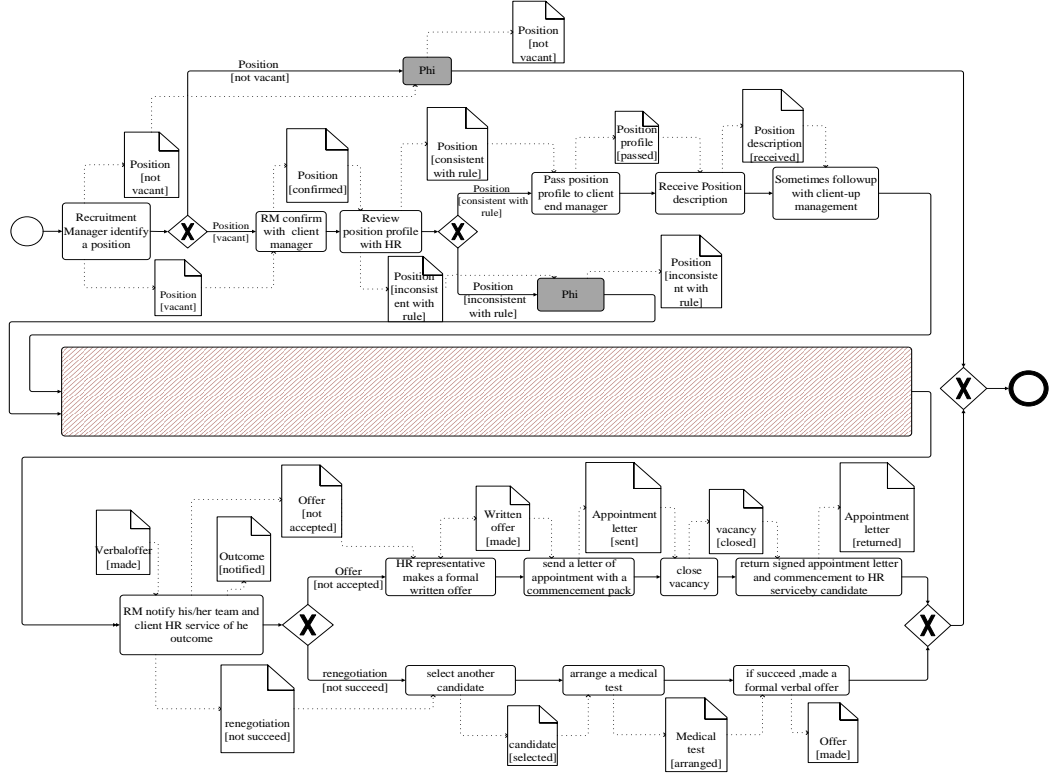


Figure 3.11: Excerpt of the Recruitment Process Model

process model. In the Appendix, Figure A.2 depicts a part of the process tree that is useful to synthesize the lifecycle of *Position* artifact of this process model. Then the lifecycles are constructed by deriving states that correspond to each artifact from the tree model; and finally, the lifecycles are refined and synchronized.

In the Appendix, Figure A.3 depicts the refined *Position* artifact lifecycle. As described in the above section, to generate the lifecycle of *Position* artifact, Algorithm 3.2 traverses each node of the process tree fragment, presented in Figure A.2, and extracts the states {*Vacant*, *Not Vacant*, *Confirmed*, *Consistent with rule*, *inconsistent with rule*, *Demand for role*, *No demand*, *New or Change existing*, *CLA5 and upwards*, *<CLA5, Approved, Determined by RM, not Determined by RM*} of this artifact and adds them according to their dependency relation. The lifecycle is then refined and synchronized with other lifecycles (see Figure A.3). The refined artifact lifecycles of

the recruitment process with states, transitions including their synchronizations revealed the feasibility of the proposed synthesis approach.

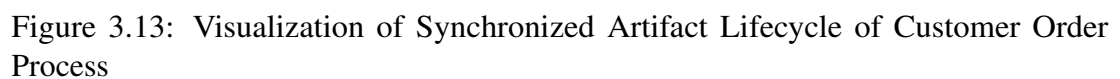
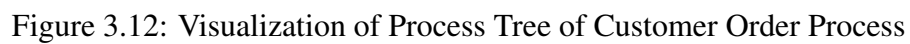
3.5.2 Implementation

This section elaborates on the prototype tool, named *Synthesis Tool*, developed to further evaluate the proposed approach. The tool takes as input an Artifact-Annotated Process Model (AAPM), specified in Extensible Markup Language (XML), and generates the synchronized artifact lifecycles. The tool is implemented using Java (Swing and AWT) concepts and XML is used in the specification of all the representations including AAPM, process tree and the synchronized artifact lifecycles. The Java API for XML Processing (JAXP) (Davidson & Mordani, 2000) is used to parse and generate each of these XML specifications. The sample of XML specification of the AAPM (BPMN) process model can be found in the Appendix, in Figure A.4.

In order to synthesize, the tool first extracts the process tree from AAPM, generates the artifact lifecycles from the process tree and then refines and synchronizes the lifecycles. The tool also provides a Graphical User Interface (GUI) to visualize all these extracted representations and also outputs their XML specifications. For example, Figure 3.12 and 3.13, present the *process tree* and *synchronized artifact lifecycles* of the motivating example (see Figure 3.1) generated using the synthesis tool.

3.5.3 Experimental Discussion

To evaluate the applicability of the proposed approach, experiments were conducted using the dataset presented in Table 3.1, which has a small collection of process models that have been chosen from the sorted (BPMAI) model collection discussed in the previous section. This dataset contains 10 process models that represent different domains including: *Travel registration (P1)*, *Business report exchange processing (P2)*,



Order delivery and payment processing (P3), Credit card transaction (P4), Risk analysis and management (P5), Donation process (P6), User account creation (P7), Online shopping (P8). This set also includes the *customer order processing(P9)* and *recruitment process(P10)* models used in the above sections to demonstrate the feasibility of the proposed approach. As given in the table below, the 10 process models range from 9 to 135 in terms of their element (nodes, artifacts and states) count and are variable in size (total number of elements) and complexity (the total number of elements and the depth of the process model). While artifact-annotation is mandatory for the proposed approach, the Data Extraction tool (Meyer & Weske, September, 2013) was used to enrich the control-flow specific process models of this set. The extraction tool is developed based on the label analysis technique introduced in the previous section.

Table 3.1: Dataset

Elements	Test Processes									
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
events	2	2	2	2	2	2	2	2	2	2
tasks	5	6	11	10	11	16	31	28	30	95
gateways	2	4	4	4	6	6	12	18	18	38
artifacts	2	5	4	10	3	10	10	15	26	46
states	7	7	13	12	18	18	32	29	36	99
Total	18	24	34	38	40	52	87	92	112	280

The Data Extraction tool transforms a process model without data annotation into a process model with explicitly defined data annotation. Mainly, this tool extracts the artifacts and states from the activity labels and data conditions associated with each *XOR-Split* gateway node in the process model. The Data Extraction tool is implemented with Natural Language Processing (NLP) capabilities, which aid the analysis of activity labels and facilitate the artifact and state extraction. Therefore, this tool was utilized to enrich the process models of the given dataset, for evaluating the proposed approach. The resulting data enriched process models are then given to the synthesis tool for

building their tree representations and to generate, refine and synchronize the resulting artifact lifecycles.

The Data Extraction tool has some limitations, in that it requires every activity label in the process model to be described in a verb-object style and every XOR gateway to be annotated with branch conditions in order to extract more than one state for an object (artifact). Due to these limitations, the evaluation was restricted to a small set of process models that satisfies the given requirements.

3.5.4 Performance Analysis

The efficiency of synthesis algorithms was first analyzed based on their time complexity and then based on the results of experiments conducted using the dataset presented in Table 3.1. As described above, the process models of this set vary in size and complexity. Mainly, the execution times of synthesis algorithms presented in Section 3.4, on each of these process models is used as a measure to analyze their efficiency. While it is obvious that the times may vary from execution-to-execution, the tests have been repeated for up to 10,000 times and the average execution time of each synthesis algorithm on each process model has been calculated and recorded during the experiments.

It is worth mentioning that complexity of a process model is used as metric to evaluate the efficiency of proposed approach. Complexity was mainly identified considering the four key perspectives including: activity, control-flow, data-flow and resource complexity of a process model discussed in (Cardoso, 2005). These perspectives focus on the number of activities that a process model has, the number of control-flow constructs including start, end nodes and the gateway elements such as splits, joins, mapping of the data objects to activities. In this context, the complexity of each process model from the given dataset is mainly derived based on the size (events, activities, gateways, artifacts and states) and depth of the process model.

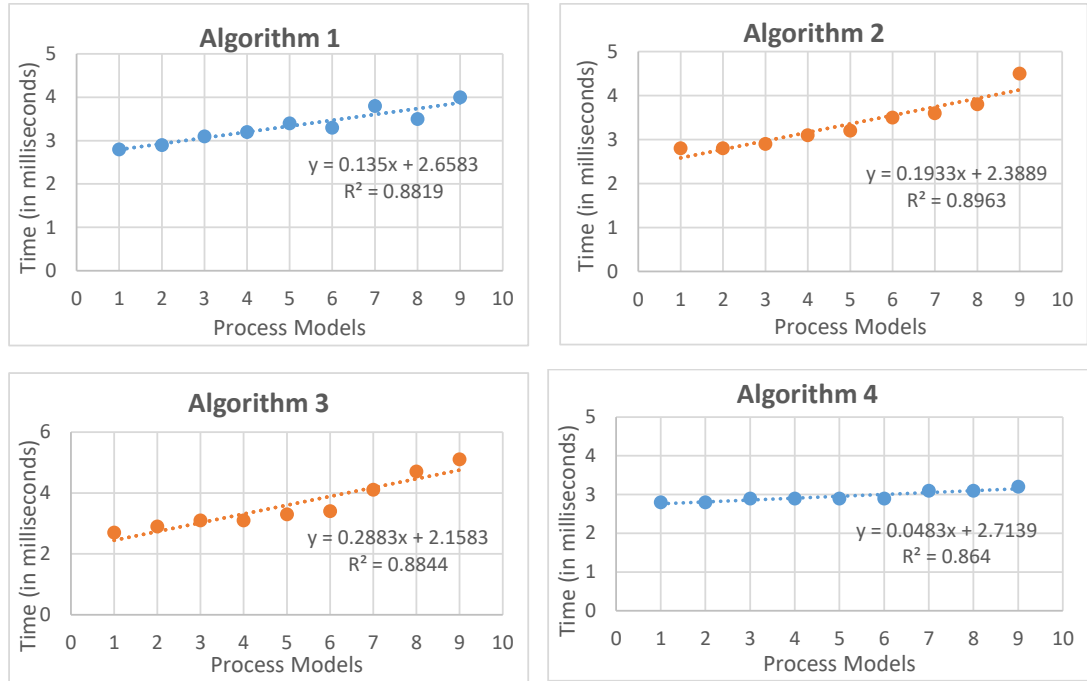


Figure 3.14: Execution times of Synthesis algorithms

The graphs presented in Figure 3.14 illustrate the execution times of each algorithm over the given dataset. It was observed that the execution times of algorithms depend on different parameters. For instance, Algorithm 3.1 depends on the number of nodes (size) and the complexity (depth) of a process model, whereas Algorithm 3.2 and 3.3 depend on the size and depth, including the number of artifacts that a process model contains. Compared to Algorithm 3.2, the execution time of Algorithm 3.3, which defines an alternative procedure to generate artifact lifecycles, is more efficient, as it takes only one traversal of the process tree to generate all the artifact lifecycles. Algorithm 3.4 mainly depends on the number of artifacts and states in a process model.

Regression analysis is used to determine how the execution times of these algorithms depend on one or more variants of the process models. The analysis results in a positive linear relationship between the two variables, which can be interpreted as the execution time increasing with the increase in process size and complexity. Therefore, it is clear from the given graphs that, the performance of each algorithm is directly proportional

to the size and complexity of the process model. In accordance with the analysis of proposed algorithms from the experiments, it can be concluded that the proposed algorithms can be performed in a nearly linear-time.

3.6 Discussion and Related Work

Although the activity-centric paradigm has been considered beneficial for control-flow specific processes, features such as backward navigation, event-driven behaviour and conversational client interaction are difficult to support with activity-centric process models (S. B. Kumaran, Liu & Wu, 2012). The *impedance mismatch* is another problem that arises due to the separation of the control flow and data aspects in activity-centric information systems (Dumas, 2011; Russo, Mecella, Patrizi & Montali, 2013; Siadat, Shokohyar & Shafahi, 2019).

The artifact-centric approach is primarily appreciated for its intuitiveness and the ability to facilitate business communication (Hull, 2008). With the seamless integration of the control flow and data aspects, this approach can resolve most of the aforementioned problems caused by the traditional approach (Dumas, 2011; S. B. Kumaran et al., 2012). Over a decade, the artifact-centric approach has been practiced and applied successfully in various domains, including *Health* (Künzle & Reichert, 2011), *Insurance* (S. Kumaran et al., 2008) and *Finance* (Chao et al., 2009). Additionally, it has been acknowledged as this approach provides higher flexibility and extensibility (Xu et al., 2011; Yongchareon et al., 2015) compared with the traditional activity-centric approach.

With the momentum of the artifact-centric approach, several works (S. Kumaran et al., 2008; R. Liu et al., 2010; Meyer & Weske, 2013; Eshuis & Van Gorp, 2016) have emphasized the generation of artifact-centric counterparts from activity-centric processes that contain artifact data flows. However, as discussed in Chapter 2, the existing approaches have either a limited view of the artifacts and their synchronizations

or they only support semi-automatic transformation. When compared to the existing approaches, this thesis presents a fully automated approach that is not only useful for generating artifact lifecycles, but also to synchronize the generated lifecycles. In addition, the details of implementation and comprehensive evaluation that reveal the applicability and efficiency of the proposed approach in practice have been presented.

The synthesis approach proposed in this chapter is related to the one presented in our work (Kunchala, Yu, Sheng, Han & Yongchareon, 2015), with a major extension, where this chapter improved the synthesis algorithms for efficiency, evaluated them extensively with a case study and implementation, and also analyzed the performance using the BPMAI process model collection.

3.7 Summary

In this chapter, the first research question (RQ1) is addressed by presenting an automated approach that aims to transform traditional activity-centric process models into artifact-centric process models. While recursive algorithms are ubiquitous for their intuitive nature, such algorithms were presented and it was demonstrated how they can facilitate the proposed transformation. To synthesize the lifecycles of artifacts, the proposed algorithms extract a tree representation of the activity-centric process model, generate each artifact lifecycle by traversing through the process tree, and, finally, refine and synchronize the artifact lifecycles. A case study is demonstrated using a recruitment process model to show the feasibility of the proposed algorithms. The implementation and performance evaluation of the proposed algorithms by utilizing a process model collection was also discussed.

Chapter 4

Merging the Collaborating Processes of Activity-Centric Inter- Organizational Business Process Models

With the emergence of the artifact-centric paradigm, several transformation approaches have been proposed to synthesize the lifecycles of *artifacts* from the standalone activity-centric process models. However, the synthesis is challenging for the inter-organizational business process (IOBP) models, as their artifacts and states are shared among their collaborating processes. Thus, unlike a standalone process model, the synthesis of artifact lifecycles from an IOBP requires the process interactions to be captured while preserving the dependencies between the involved artifacts and states in the resulting lifecycles.

This chapter aims to propose an approach that can be used to solve the problem of the transformation of activity-centric IOBP models into artifact-centric process models, remaining from chapter 3, where a transformation approach was presented

for the standalone activity-centric process models. The approach presented in this chapter is based on the notion of process merging. As discussed in chapter 2, most of the existing works in this regard focus on combining the process models that have common process fragments and that do not contain the artifacts and states. Therefore, an automated approach that aims to merge the collaborating processes of an activity-centric IOBP model that contains artifact data flows is proposed (Kunchala et al., 2019). The proposed merge approach is comprised of algorithms that combine the nodes of two or more collaborating processes to generate an integrated (or merged) process model which can be used to synthesize the artifact lifecycles pertinent to an IOBP. The proposed algorithms are also implemented and evaluated using the BPMNI process model collection.

The remaining chapter is organized into seven sections. Section 4.1 introduces a motivating example based on an Order Processing Scenario. Section 4.2 formulates the problem statement and defines the key notions used in deriving a solution for the proposed problem. Section 4.3 presents an overview of the proposed merge approach and defines the patterns for process interactions. Section 4.4 presents algorithms for the merge approach. Section 4.5 demonstrates a case study and the implementation and performance evaluation of the proposed approach. Section 4.6 discusses and reviews the related work. Section 4.7 summarizes the contributions of this chapter.

4.1 Motivating Example

In this section, a Buyer-Seller e-business process scenario is presented (Kunchala, Yu, Yongchareon & Han, 2017) to demonstrate the proposed approach. The process model given in Figure 4.1 is an IOBP, represented using the BPMN modeling notation. As shown in the figure, the IOBP consists of collaborating processes *Buyer* and *Seller* that belong to two different organizations.

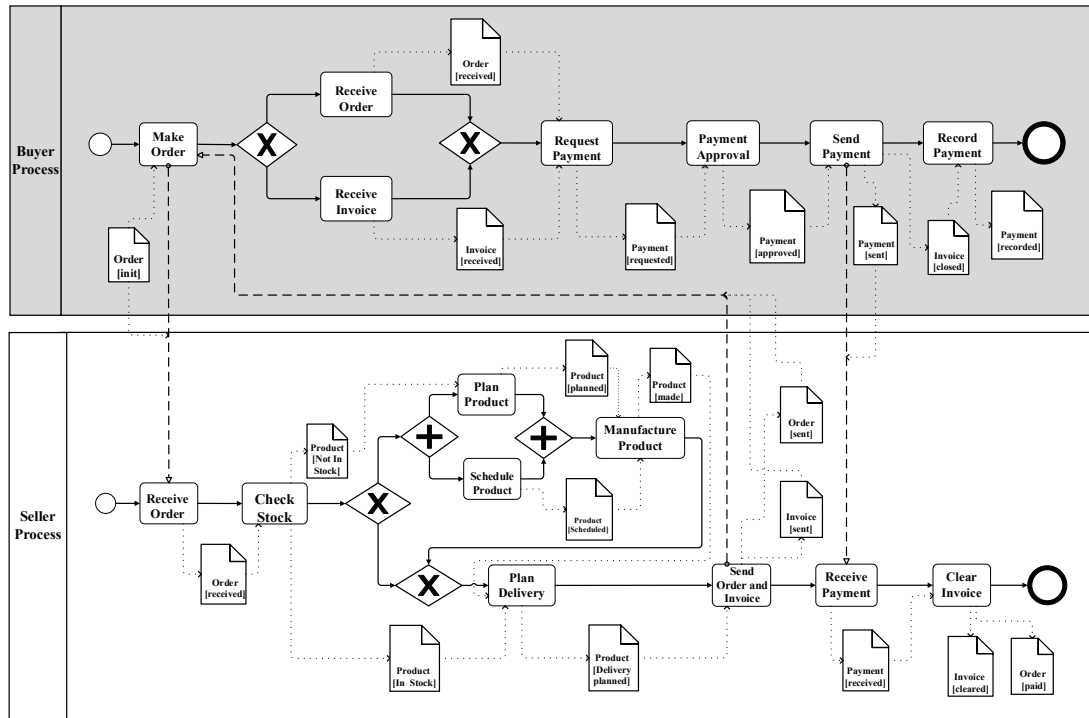


Figure 4.1: Buyer-Seller Inter-organizational Business Process (IOBP)

An organized set of activities depicted in the given Buyer-Seller IOBP are implemented by the two organizations to fulfil their common business objective. In this chapter, an artifact-annotated version of IOBP (shortly AAIIOBP) is used, which is an IOBP annotated with artifacts and states. The process highlighted (grey) in the figure is a main process (Buyer) that initiates the collaboration with its participating process (Seller). The dotted lines with an arrow head represent the interactions between the *Buyer* and *Seller* processes. Every interaction represents a message flow that is associated with artifact(s) and state(s) that are sent and received between the two collaborating processes to perform a business activity.

The Buyer-Seller collaboration here aims to fulfil the order-delivery process. Therefore, the *Buyer* of one organization initiates the collaboration by placing an order with the *Seller* of another organization and it ends with the successful delivery and payment of the product. On the other end, the *Seller* process starts with receiving an order from

the *Buyer*, and ends with successful delivery of the product and invoice clearance.

In order to fulfill their objectives, the *Buyer* and *Seller* processes interact at several stages by sending and receiving corresponding artifacts and states. For example, in the given IOBP the first interaction is from the *Buyer* process, where a message is sent to order a product from the node *MakeOrder* to the node *ReceiveOrder* of the *Seller* process, by passing an *Order* artifact with its associated state *init*. In the same way, the *Seller* process sends back the two artifacts *Order* and *Invoice* from its *SendOrderandInvoice* node when the product is ready. In this manner, as shown in Figure 4.1, the *Buyer* and *Seller* processes interact to complete the order processing.

4.2 Problem Statement and Definitions

In this section, the research problem targeted in this chapter is first stated and then some of the key notions of the proposed solution are defined. These notions include *Process Model*, *Artifact-Annotated Inter-Organizational Business Process (AAIOBP) Model* and *Integrated Process Model*.

PROBLEM (PROCESS MERGING). Given an activity-centric IOBP model with artifact data flows, an integrated process model that not only contains the nodes and interactions (data flows) of the collaborating processes, but also preserves their structure and behaviour must be derived.

To address the above defined problem, the merge is considered as an ideal solution that is used to combine (or integrate) the collaborating processes of IOBP. However, the proposed merge should not only combine the nodes of collaborating processes but also their interactions and must preserve their structure and behaviour in the resulting integrated process. When properly merged, the resulting integrated process model can enable the synthesis of behaviour preserved artifact lifecycles.

It is worth mentioning that the motivating example (AAIOBP), presented in Figure

4.1, combines all the fundamental elements of BPMN, therefore each (collaborating) process model of the given AAIIOBP and an AAIIOBP itself is formally defined as follows:

Definition 6. (Process Model). A process model denoted with $\pi = (E, A, G, F)$, where

- E is a finite set of events. Every event has a type in $E.Type = \{\text{Start}, \text{End}\}$;
- A is a finite set of activities;
- G is a finite set of gateways. Every gateway has a type in $G.Type = \{\text{XOR}, \text{AND}\}$;

and G_S is used to represent split (opening) gateways and G_M is used to represent merge (closing) gateways;

- $F \subseteq (E \cup A \cup G) \times (E \cup A \cup G)$ is the set of sequence flow relations among events, activities and gateways.

Definition 7. (Artifact-Annotated IOBP (AAIOBP) Model). An AAIIOBP model denoted as $\Pi = \{\pi\} \cup \{D, S, I, O, M\}$ is a set of collaborating business processes, where

- D is a finite set of artifacts;
- S is a finite set of states;
- $I = D \times S \times A$ is the input relation between artifacts, states and activities.
- $O = A \times D \times S$ is the output relation between artifacts, states and activities.
- M is a set of message flow relations between the nodes of collaborating processes.

Every collaborating process of AAIIOBP is assumed as block-structured and structurally sound. It is worth noting that, to avoid behavioural inconsistencies in the resulting integrated process, a pair of nodes of the main process can only interact with a pair of nodes of the collaborating process that have a similar dependency. Meaning that, the nodes that have a sequence dependency between them can only interact with nodes that have the same dependency, similarly the nodes that have either parallel or exclusive dependency (for example: the nodes inside a parallel or exclusive gateway of one process)

can only interact with nodes that are either parallel or exclusive (inside a parallel or exclusive gateway of other process).

Next, definitions are provided for a structured process model and well-behaved process model based on work on structure workflow modeling (Kiepuszewski et al., 2000).

Definition 8. (Structured Process model (SPM)). A structured process model is inductively defined as below, meaning that a SPM can take one of the following forms.

1. A process consisting of a single activity node is a SPM.
2. Let P and M be SPMs and G_S and G_M be the *split* and *merge* nodes of a *parallel* (AND) gateway. The process with an initial node G_S and final node G_M , and with sequence flows between G_S and the activity node of P , between the activity node of P and the G_M , between G_S and the activity node of M , and between the activity node of M and the G_M , is also a SPM (See Figure 4.4).
3. Let P and M be two SPMs. The merge of these processes, where the activity node of P has a sequence flow to the activity node of M is then also a SPM. The initial node of this SPM is the activity node of P and its final node is the activity node of M (See Figure 4.5).
4. Let P and M be SPMs and G_S and G_M be the *split* and *merge* nodes of a *parallel* gateway. The process with an initial node G_S and final node G_M , and with sequence flows between G_S and the activity node of P , between the activity node of P and the G_M ; between G_S and the activity node of M , and between the activity node of M and the G_M ; between the G_M and the next activity node of M is also a SPM by extending Definition 3(2)(See Figure 4.9).

Definition 9. (Well-behaved Process Model). A process model is well-behaved if that model never leads to deadlock nor results in multiple active instances of the same activity. Every structured process model is therefore a well-behaved process model.

4.3 The Merge Approach

The collaborating processes of AAIIOBP have a decentralized peer-to-peer (P2P) interaction, where every participating process will have the same capabilities and can initiate the interaction. As described above, the interaction is through messages, mainly between the activity nodes of collaborating processes, where a process initiates an interaction by sending messages to another process.

In order to systematically carry out the merge, it is necessary to classify the process interactions into patterns that uniquely represent each interaction. Therefore, W3C message exchange patterns (MEPs) (Chinnici et al., 2007) are adopted and classified into two types of process interaction patterns: *synchronous* and *asynchronous*. Each of these interaction patterns are clearly described in the following section.

4.3.1 Process Interaction Patterns

Based on the message exchange patterns (MEPs) proposed in (Chinnici et al., 2007), different patterns are derived, namely *Out-then-In* and *In-then-Out* patterns, and *Only-Out* and *Only-In* patterns for defining the synchronous and asynchronous process interactions. Below, an interaction pattern is formally defined as a 3-tuple.

Definition 10. (Interaction Pattern). $I \subseteq \Pi.M$ denote a set of interaction patterns (or message flows) between the nodes of collaborating processes. Every interaction pattern $i \in I$ is associated with a tuple $(Itype, d, s)$, where $Itype$ refers to one of the two types of interaction patterns $\{sync, async\}$, whereas d and s represent the associated artifacts and states, respectively, where $d \subseteq \Pi.D \wedge s \subseteq \Pi.S$.

We can see an example from Figure 4.2.a, where the *Out-then-In* and *In-then-Out* patterns represent the synchronous interaction between the nodes of collaborating processes, while the patterns *Out-Only* and *In-Only* represent the asynchronous interaction.

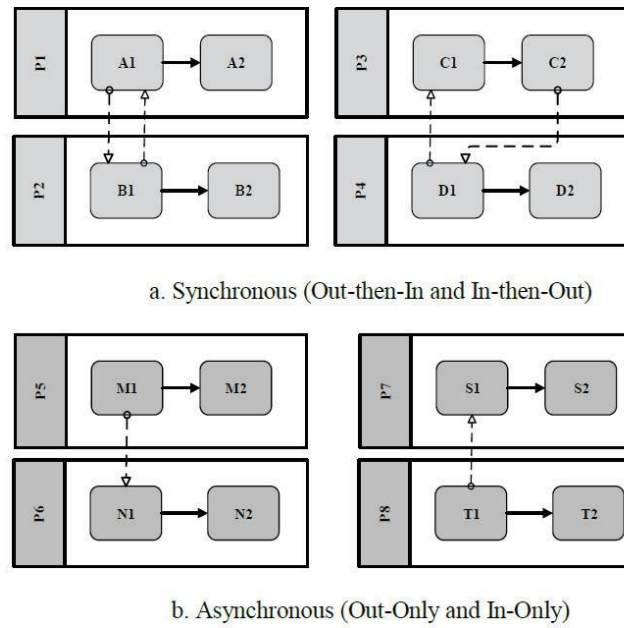


Figure 4.2: Process Interaction Patterns

For ease of understanding, this figure omits the artifacts and states associated with each message flow.

Synchronous interaction (*Out-then-In*, *In-then-Out*). The synchronous interaction is a blocked or two-way interaction between processes. In this type of interaction, the main process is blocked from execution after sending a message and is resumed after receiving a return message from the collaborating process. Here, the main process is said to have an *Out-then-In* pattern, as it sends first and then receives a return message from the collaborating process, while the collaborating process will have an *In-then-Out* pattern, as it receives first and then returns a message back to the main process.

A simple example from Figure 4.2 may be considered to further comprehend these patterns, where the processes *P1* and *P4* have the *Out-then-In* patterns as they are sending and then receiving the messages to/from the collaborating processes *P2* and *P3* which in turn have the *In-then-Out* patterns.

Asynchronous Interaction (*Out-Only*, *In-Only*). The asynchronous interaction is a non-blocked or one-way interaction, in which the main process can continue its

execution after sending a message to the collaborating process. Here, this interaction can be defined using the two asynchronous patterns, where the sending process is said to have an *Out-Only* pattern, as it only sends the message and the receiving process contains an *In-Only* pattern, as it only receives the message.

As shown in Figure 4.2.b, the sending processes *P5* and *P8* contain *Out-Only* patterns and the receiving processes *P6* and *P7* contain *In-Only* patterns.

From here onwards, the nodes associated with the interaction patterns are referred to as *sync* nodes, while the nodes that have no interaction patterns are referred to as *non-sync* nodes. Therefore, these nodes are formally defined as follows.

Definition 11. (Sync and Non-Sync Nodes). An activity node a_i is a *sync node*, if $a_i \in \pi_i.A$ and there exists a mapping $m : a_i \rightarrow i$ or $m : i \rightarrow a_i$, where $i \in I$ is an interaction pattern, otherwise it is referred to as a *non-sync* node.

For a sync node for instance a_i , $a_i.Type$ returns its node type such as *sync* or *non-sync* and $a_i.Itype$ returns its interaction pattern such as synchronous (sync) or asynchronous (async). A gateway can also be called as a sync node if at least one of its branch nodes have an interaction pattern.

A pair of sync nodes can be referred to as *sender sync node* and *receiver sync node*, where the sender sync node contains either synchronous *Out-then-In* or asynchronous *Only-Out* pattern and the receiver sync node contains either a synchronous *In-then-Out* or an asynchronous *Only-In* pattern. For example, referring to Figure 4.2.a, the nodes *A1*, *D1*, *M1* and *T1* are *sender sync nodes*, while the nodes *B1*, *C1*, *N1* and *S1* are *receiver sync nodes*. Here, we can see another *sync node* *C2* that participates in the synchronous interaction with *C1*. The remaining nodes are the *non-sync* nodes, as they do not have any interaction patterns.

Definition 12. (Sync and Non-Sync Sets). The sync set $c_i \subseteq \pi_i$ is a set of sequence flow nodes (subprocess) of the receiving (can be main or collaborating) process that

Table 4.1: Summary of symbols used

Symbol	Description
\parallel	Merge
$m \rightarrow c$	Message flow between nodes m and c
m_1, \dots, m_k	Subprocess of π_m
$m \rightarrow c$	Flow relation between m and c
G_S	Split node of parallel (AND) gateway
G_M	Merge node of parallel (AND) gateway
S_c	Split Condition
\models	Satisfies (Ex. $m \models S_c$)
m^{pre}, m^{post}	The <i>pre</i> and <i>post</i> split nodes of m

is involved in a specific synchronous interaction pattern, while we can refer to a set of sequence flow nodes that have no participation in the interaction, as a non-sync set.

A sync set (Receive Order, ..., Send Order and Invoice) can be observed from Figure 4.1, which is a subprocess of *Seller* process that participates in the synchronous interaction with the *Make Order* node of *Buyer* process.

4.3.2 Merge Overview

The *merge* is defined as a process of combining or integrating the nodes of any two collaborating processes of an Artifact-Annotated Inter-Organizational Business Process (AAIOBP) model following their *synchronous* and *asynchronous* interaction patterns. It is worth mentioning that, after the merge, the message flows between the collaborating processes will turn as the flow relations between their nodes in the resulting integrated process model and the associated artifacts and states are annotated as input and output to the nodes that send and receive them.

In the following sections, the merge is demonstrated over a pair of collaborating

processes. Without a loss of generality, more than two collaborating processes can be merged in a two by two manner.

Definition 13. (Merge). Let π_m and π_c be the two collaborating processes of AAIOBP Π , then their merge is expressed as $\pi_m || \pi_c = \Pi^I = (E_i, A_i, G_i, D_i, S_i, F_i, I_i, O_i)$, where the integrated process Π^I is a finite sets of events, activities, gateways, artifacts, and states, including the sequence flow relations, and the input and output artifact and state relations of activities. In simple terms, the *merge* is a union of all the collaborating processes of an AAIOBP model.

From the above definition, the definition for the integrated process model can be derived as follows.

Definition 14. (Integrated Process Model). An integrated process $\Pi^I = \{\pi_0 \cup \pi_1 \cup \dots \cup \pi_n\}$ is an integration of all the collaborating processes of Π an AAIOBP model, which can be expressed as $\bigcup_{i=1}^n \pi_i$.

4.3.3 Types of Merge

The merge is classified into two types: *Parallel Merge* and *Interactive Merge*, as illustrated in Figure 4.3. Here, the classification is mainly to differentiate the nodes that participate in the collaboration with those that have no participation. Therefore, the *parallel merge* emphasizes the combining of the process nodes that do not send and receive messages.

While on the other hand, the *interactive merge* combines the interacting nodes based on their interaction patterns. Therefore, as shown in the figure, the interactive merge is divided into *synchronous* and *asynchronous* types, which are further classified to deal with splitting and non-splitting of the nodes based on their message flows.

Next, the above merge types are briefly elaborated and the algorithms for each of them are defined.

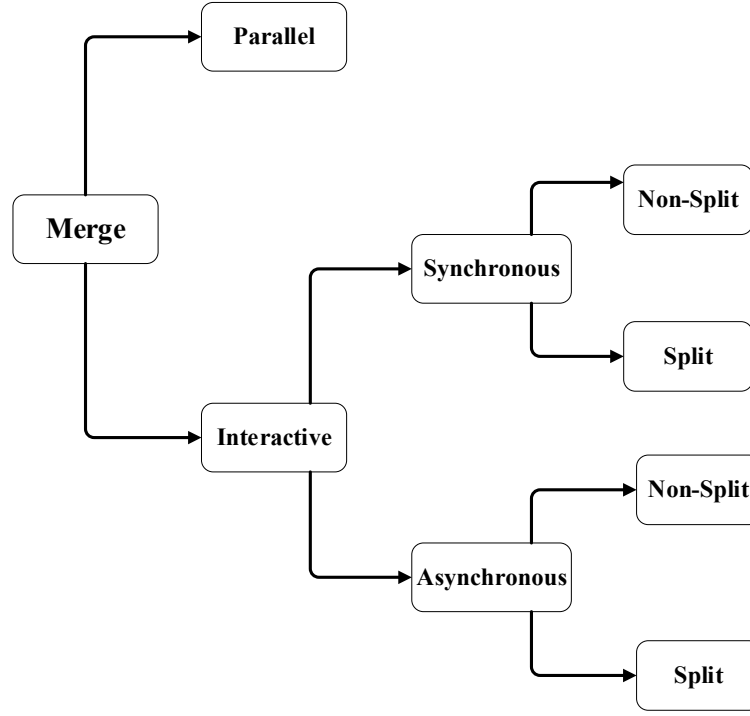


Figure 4.3: Types of Merge

Parallel Merge. This is a technique for merging the non-interacting (non-sync) nodes of two collaborating processes, meaning that the nodes that do not contain interaction patterns. As no dependencies exist among non-sync nodes, a new *parallel* (AND) gateway node is used to merge them, where the non-sync node from each of the collaborating processes is added to a different branch of this new gateway. A rule to merge two nodes in parallel is defined below.

Definition 15. (Parallel Merge). Let $m \in \pi_m$, $c \in \pi_c$ be the nodes of two collaborating processes that have no interaction, then the parallel merge \parallel^P of these nodes is formulated by using the rule given below.

$$m \parallel^P c = \frac{m, c}{m, c, G_S \rightarrow m, G_S \rightarrow c, m \rightarrow G_M, c \rightarrow G_M} \quad (4.1)$$

Where G_S and G_M are the split (open) and merge (close) nodes of a new parallel (AND) gateway.

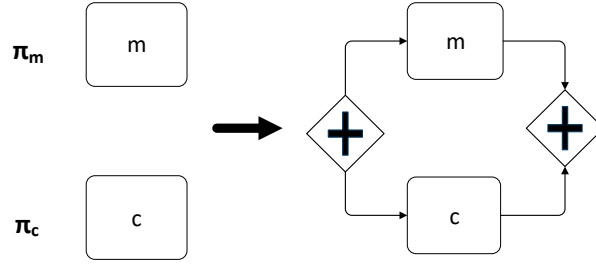


Figure 4.4: Parallel Merge

An example is presented in Figure 4.4, where a new *parallel* (AND) gateway is used to support the parallel execution of non-sync nodes, as they do not require waiting for messages associated with artifacts and states. According to the given rule, the two non-sync nodes must be attached to different branches of the gateway so that they can run in parallel. Here, the branches of *And-Split* gateway node are merged when a new sender sync node or an end node is reached in either of the collaborating processes.

Interactive Merge. This is a technique for merging the interacting (sync) nodes of two collaborating processes, meaning that the nodes contain interaction patterns. An interactive merge always starts from the *sync* node that contains a synchronous *Out-then-In* pattern or asynchronous *Out-Only* pattern. The other important aspect of this merge is to split the *sync* node depending on a condition defined below.

Definition 16. (Split Condition). An activity node $a_i \in \pi_i.A$ is split into a_i^{pre} and a_i^{post} , if it satisfies a condition that, $\forall (d_i, s_j) \in O_i(a_i)$, if $(d_i, s_k) \in R_i(a_i) \wedge s_j \neq s_k$. Here $O_i(a_i)$ is the set of output artifacts of a_i and $R_i \subset O_i(a_i)$ is the set of artifacts returned to a_i . When split, for example the node a_i is referred to as a *split node*.

According to the above split condition, an activity node of main process is split into *pre* and *post* nodes, if each of its output artifacts and states are different from the artifacts and states that are returned to it from the activity node of collaborating process. Here a node must be split, when artifact states need to be inserted between the beginning

and the end of the execution node.

A. Synchronous Merge. This merge follows a sequential approach, meaning that it connects two *sync* nodes, such as the *sender sync node* and the *receiver sync node*, in a sequence relation. As mentioned at the start of this section, synchronous merge is further divided into two types: *non-split* and *split*.

i. Synchronous Non-Split Merge. The synchronous non-split merge is applied when there is no splitting of a sender sync node while merging it, as its output artifact and state is the same as the one it receives from the receiver sync node. The synchronous non-split merge of two collaborating nodes is defined as follows.

Definition 17. (Synchronous Non-Split Merge). Let $m \in \pi_m$, $c \in \pi_c$ be the nodes of two collaborating processes that participate in a synchronous interaction, then the synchronous non-split merge \parallel^{S^n} of these nodes is formulated as the following rule.

$$m \parallel^{S^n} c = \frac{m, c, m \rightarrow c, c \rightarrow m}{m, c, m \rightarrow c} \quad (4.2)$$

An example can be seen in Figure 4.5, where the two interacting nodes m and c that have a synchronous interaction pattern are merged in sequence, which means they are connected in sequence in the resulting integrated process model.

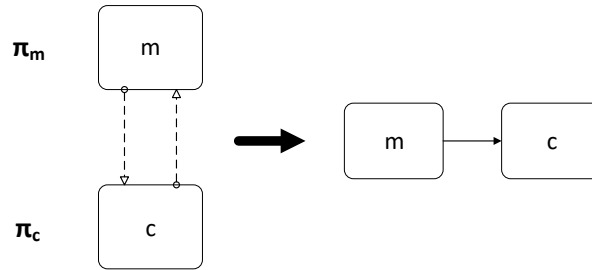


Figure 4.5: Synchronous Non-Split Merge

Next, a rule to merge a node (sender sync node) and the subprocess (receiver sync set) of two collaborating processes that have a synchronous interaction is defined.

Let $m \in \pi_m$, $c_1, \dots, c_n \subseteq \pi_c$ be a node and the subprocess of two collaborating processes that participate in a synchronous interaction, then the synchronous non-split merge \parallel^{S^n} of these process nodes is based on the rule given below.

$$m \parallel^{S^n} c = \frac{m, (c_1, \dots, c_n), m \rightarrow c_1, c_n \rightarrow m}{m, c, m \rightarrow c_1, c_1 \rightarrow (c_2, \dots, c_n)} \quad (4.3)$$

Figure 4.6 demonstrates how a sender sync node of the main process is merged with a sync set (or subprocess) of the collaborating process that participates in the synchronous interaction. Similar to rule 4.2, the above rule also follows a sequential approach to merge these nodes, where the complete block of the subprocess is connected in sequence to the sender sync node as shown in the figure. For simplicity, artifact annotation is omitted in this figure.

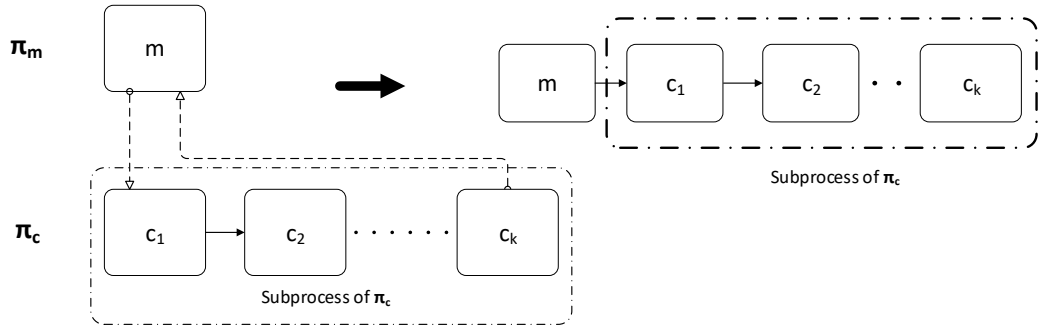


Figure 4.6: Synchronous Non-Split Merge

ii. Synchronous Split Merge. The synchronous split merge is a splitting of the *sender sync node* into *pre* and *post* nodes while merging it, as its output artifact and state are not the same as the one it receives from the *receiver sync node*. When split, the *receiver sync node* becomes a part of the *sender sync node*, which takes input from the *pre* node of *sender sync node* and passes input to the *post* node of *sender sync node*.

Definition 18. (Synchronous Split Merge). Let $m \in \pi_m$, $c \in \pi_c$ be the nodes of two collaborating processes that participate in a synchronous interaction, where m satisfies

the split condition S_c then the synchronous split merge $\|^{S^s}$ of these nodes is formulated by using the rule given below.

$$m \|_c^{S^s} = \frac{m, c, m \rightarrow c, c \rightarrow m, m \models S_c}{m, c, m^{pre} \rightarrow c, c \rightarrow m^{post}} \quad (4.4)$$

Figure 4.7 presents an example of synchronous split merge, where the node m is split into m^{pre} and m^{post} , as its output artifact and state is not same as the one it receives from the node c . Here the splitting of a sender sync node is necessary in order to complete its post-processing after receiving an input returned by its receiver sync node. As shown in the figure, when m is split, the node c becomes a part of m , which takes input from the *pre* node of m and passes input to the *post* node of m .

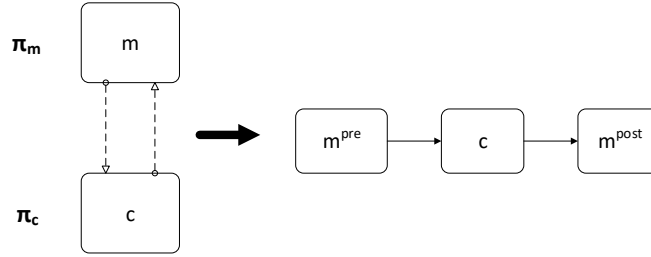


Figure 4.7: Synchronous Split Merge

B. Asynchronous Merge. The asynchronous merge specifies how the sync nodes with asynchronous interaction pattern are merged. Similar to the synchronous merge, this merge also follows either a sequential approach or parallel approach, depending on whether the nodes satisfy the given split condition or not. The key difference here is, instead of the sender sync node, the receiver sync node is split into *pre* and *post* nodes if it satisfies the given split condition. The two merge notions of this type, including *Asynchronous Split Merge* and *Asynchronous Non-Split Merge*, are clearly described below.

i. Asynchronous Non-Split Merge. This merge follows a sequential approach, where the *sender sync node* is connected in sequence to the *receiver sync node*, as the receiver

sync node does not satisfy the given split condition.

Definition 19. (Asynchronous Non-Split Merge). Let $m \in \pi_m$, $c \in \pi_c$ be the nodes of two collaborating processes that have an asynchronous interaction, then the asynchronous non-split merge \parallel^{A^n} of these nodes is formulated as the following rule.

$$m \parallel^{A^n} c = \frac{m, c, m \rightarrow c}{m, c, m \rightarrow c} \quad (4.5)$$

An example of asynchronous non-split merge is given in Figure 4.8, where the two nodes m and c that contain an asynchronous interaction pattern are connected in sequence, as the receiver sync node c does not satisfy the given split condition.

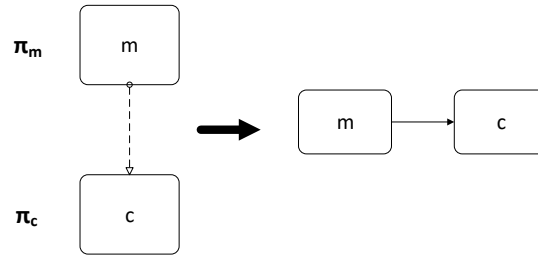


Figure 4.8: Asynchronous Non-Split Merge

ii. Asynchronous Split Merge. This merge is based on the notion of splitting the *receiver sync node* into *pre* and *post* nodes if it satisfies the given split condition. Similar to the parallel merge, here a new *parallel* gateway is used to merge the *sender sync node* with the split nodes of the *receiver sync node*. Mainly the *pre* node of the *receiver sync node* and the *sender sync node* are connected to different branches of the *parallel* gateway in order to execute simultaneously and to pass input to the *post* node of *receiver sync node*.

Definition 20. (Asynchronous Split Merge). Let $m \in \pi_m$, $c \in \pi_c$ be the nodes of two collaborating processes that have an asynchronous interaction, where c satisfies the split condition S_c then the asynchronous split merge \parallel^{A^s} of these process nodes is formulated

as below.

$$m \parallel^A c = \frac{m, c, m \rightarrow c, c \models S_c}{m, c, G_S \rightarrow m, G_S \rightarrow c^{pre}, m \rightarrow G_M, c^{pre} \rightarrow G_M, G_M \rightarrow c^{post}} \quad (4.6)$$

An example of an asynchronous split merge is presented in Figure 4.9, where the receiver sync node c is split into c^{pre} and c^{post} nodes, as it satisfies the given split condition. According to the given rule, the sender sync node m and the pre node of receiver sync node c must be connected in parallel and the $post$ node of receiver sync node c must be connected in sequence to these nodes after merging the parallel gateway.

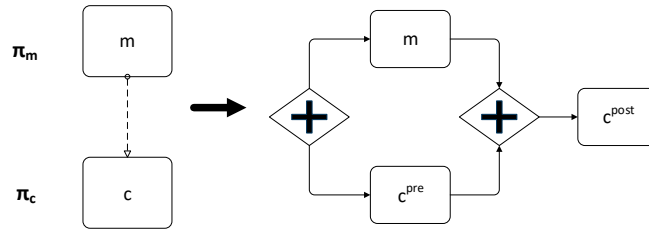


Figure 4.9: Asynchronous Split Merge

4.4 Algorithms

Algorithm 4.1 defines a procedure to merge the collaborating processes of AAIOPB. The input to this algorithm is the two collaborating processes and an empty integrated process model Π^I , while the output is a completely integrated process model. *Merge()* is a recursive function that recursively calls upon a pair of nodes starting from the *start* nodes of two processes and is terminated after merging their *end* nodes.

As defined in the algorithm (in line 4 and 5), the *Merge()* first combines the *start* nodes of two processes and adds it to the integrated process model. Then, it considers the next pair of nodes for merging and, depending on their node types (such as *sync* or *non-sync*), the functions *MergeSyncNodes()* and *MergeNon-SyncNodes()* for *parallel*

and *interactive* merge are called. According to the algorithm (see lines 10-14), the function *MergeSyncNodes()* is called when the two process nodes m and c are *sync* nodes, meaning that they have an interaction between them, while the *MergeNon-SyncNodes()* is called when one among the two process nodes is a *sync* node. As mentioned in the previous section, the sync node can be an activity or a gateway that encloses one or more interacting nodes.

Algorithm 4.1: Merge the Collaborating Processes of AAIOBP

Input : Two collaborating processes π_m and π_c of AAIOBP (Π), an empty integrated process model Π^I

Output : Complete integrated process model Π^I

```

1 Function Merge ( $m : \pi_m, c : \pi_c, \Pi^I$ ) :
2   if  $m \neq null$  and  $c \neq null$  then
3     if  $m \in \pi_m.E$  and  $c \in \pi_c.E$  then
4       if  $m = Start$  and  $c = Start$  then
5          $\Pi^I \leftarrow \Pi^I \cup \{Start\}$ 
6       else if  $m = End$  and  $c = End$  then
7          $\Pi^I \leftarrow \Pi^I \cup \{End\}$ 
8       end
9     else
10      if  $m.Type = sync$  and  $c.Type = sync$  then
11        MergeSyncNodes( $m, c, \Pi^I$ )
12      else
13        MergeNon-SyncNodes( $m, c, \Pi^I$ )
14      end
15    end
16    Merge(GetSeqNext( $m$ ), GetSeqNext( $c$ ),  $\Pi^I$ )
17  end
18 End Function

```

An example from the AAIOBP presented in Figure 4.1, where after merging the *start* nodes of Buyer and Seller processes, the next pair of nodes *MakeOrder* and *ReceiverOrder* are considered for merging. According to Algorithm 4.1, the node type of these nodes (*sync* or *non-sync*) is first identified. As these nodes are of type *sync*, the

MergeSyncNodes() function defined in Algorithm 4.3 is invoked by passing these two sync nodes and the integrated process model.

4.4.1 Parallel Merge: Merging Non-Synchronous Nodes

The procedure to merge the non-interacting (or non-sync) nodes is presented in Algorithm 4.2. The recursive function *MergeNon-SyncNodes()* defined in the algorithm is based on rule 4.1, formulated for *parallel merge*, where the two non-sync nodes are merged in parallel by using a new *parallel* gateway. Input to this recursive function is a pair of process nodes m and c , a parallel (AND) gateway G^{AND} (with G_S and G_M , the *split* and *merge* nodes) and the integrated process model Π^I . The output of this function is the integrated process model Π^I with the newly merged fragment (G^{AND}).

According to Algorithm 4.2 (line 2), the type of each node (*sync* or *non-sync*) is tested before it is connected to the new *parallel* gateway. Here, only the *non-sync* nodes are added to the gateway, otherwise they are handled by Algorithm 4.1 in the following run. The *AddInSeq()* function is used to connect every *non-sync* node in sequence to the *parallel* gateway and the *AnnotateIO()* function is used for annotating the input and output artifacts and states for each merged node.

The parallel merge is based on different conditions defined in the algorithm (see line 2, 11 and 18). Therefore, according to first condition, when both nodes m and c are *non-sync* nodes, then each of them are connected in sequence to the *split* node (G_S) of new parallel gateway and then sequence flows are added from these nodes to the *merge* node (G_M) of the parallel gateway. Upon finishing the merge of these two non-sync nodes, the *Merge()* recursive function defined in Algorithm 4.1 is invoked with the next sequence flow nodes of m and c .

Figure 4.10 presents an example of the above case based on the motivating example, where the two non-interacting nodes *RecordPayment* and *ClearInvoice* are merged using

a parallel gateway. As shown in the figure, the two non-interacting nodes are connected to different branches of the *parallel* gateway with the corresponding artifacts and states.

Algorithm 4.2: Parallel Merge of Non-Synchronous Nodes

Input : Two nodes m, c , a new parallel gateway G^{AND} with G_S (split) and G_M (merge) nodes, an Integrated process model Π^I

Output : Integrated process model Π^I with complete parallel gateway G^{AND}

```

1 Function MergeNon-SyncNodes ( $m : \pi_m, c : \pi_c, G^{AND}, \Pi^I$ ) :
2   if  $m.Type \neq sync$  and  $c.Type \neq sync$  then
3      $G_S.AddInSeq(m)$ 
4      $AnnotateIO(m)$ 
5      $m = MergeNextNodes(m, G_M)$ 
6      $G_S.AddInSeq(c)$ 
7      $AnnotateIO(c)$ 
8      $c = MergeNextNodes(c, G_M)$ 
9      $\Pi^I \leftarrow \Pi^I \cup G^{AND}$ 
10     $Merge(GetSeqNext(m), GetSeqNext(c), \Pi^I)$ 
11  else if  $m.Type \neq sync$  and  $c.Type = sync$  then
12     $G_S.AddInSeq(m)$ 
13     $AnnotateIO(m)$ 
14     $m = MergeNextNodes(m, G_M)$ 
15     $G_S.AddInSeq(G_M)$ 
16     $\Pi^I \leftarrow \Pi^I \cup G^{AND}$ 
17     $Merge(GetSeqNext(m), c, \Pi^I)$ 
18  else
19     $G_S.AddInSeq(G_M)$ 
20     $G_S.AddInSeq(c)$ 
21     $AnnotateIO(c)$ 
22     $c = MergeNextNodes(c, G_M)$ 
23     $\Pi^I \leftarrow \Pi^I \cup G^{AND}$ 
24     $Merge(m, GetSeqNext(c), \Pi^I)$ 
25  end
26 End Function

```

If the nodes satisfy the second condition (line 11), one among the two nodes is a *non-sync* node and the other is a *sync* node. In this case, only the non-sync node for

instance m is added in sequence to the *split* node of *parallel* gateway. Then a sequence flow is added from m to the *merge* node of *parallel* gateway and also from the *split* node to the *merge* node. In this case, the sync node, for instance, c and the next sequence flow node of non-sync node m are considered next for merging.

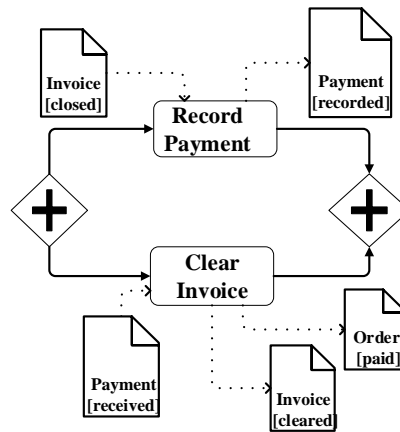


Figure 4.10: Parallel Merge

An example for this case can be found in Figure 4.11, where a non-sync node such as an XOR gateway block of the given AAIOPB, is appended to a branch of a new *parallel* gateway. Here other branch of the *parallel* gateway is empty, as the second node *ReceivePayment* that is paired for merging with this node is a *sync* node. As mentioned above, the *ReceivePayment* sync node is considered for merging with the next sequence flow node of an exclusive gateway in the following iteration of merge.

While in the last case (line 18), where the node c is a *non-sync* node and m can be a *sync* node or an *end* node of one process. In this case, the node c is merged using a new *parallel* gateway as explained above, and the node m is considered for merging with the next node of c in the following run.

The function *MergeNextNodes()* defined below, is used in the algorithm to merge the next sequence flow nodes of a *non-sync* node (non-sync set), which are also of type *non-sync*. This function is useful, where instead of using a new *parallel* gateway, one gateway can be sufficient to merge any following non-sync nodes. Therefore, as defined

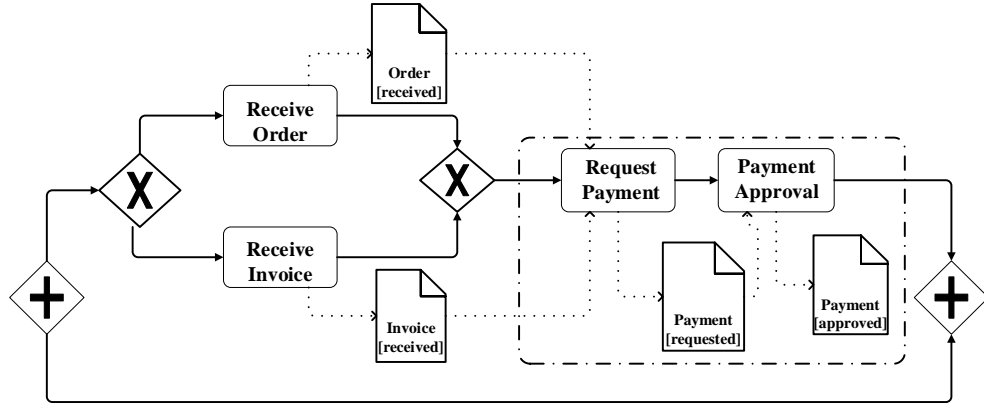


Figure 4.11: Parallel Merge

in the function, if the next sequence flow node *next* of a *non-sync* node *n* is also of type *non-sync*, then this node is added in sequence to its previous node without the use of a new *parallel* gateway. Here the merge is continued until a new *sync* node or an *end* node in that process is reached. The parallel gateway is closed using the merge node G_M before this function is terminated.

Function: MergeNextNodes($n : \Pi$, $G_M : G^{AND}$)

$next = GetSeqNext(n)$

while $next.Type \neq sync$ and $next.Type \neq End$ **do**

$n.AddInSeq(next)$

$AnnotateIO(next)$

$n = next$

$next = GetSeqNext(next)$

end

$n.AddInSeq(G_M)$

return n

In Figure 4.11, an example for the above function can be found, where the two *non-sync* (sequence flow) nodes *RequestPayment* and *PaymentApproval* of the XOR gateway, which are also of type *non-sync* (as none of its branch nodes have interactions) are added to the same branch of the parallel gateway in sequence with the XOR gateway.

4.4.2 Interactive Merge: Merging Synchronous and Asynchronous Nodes

Algorithm 4.3 defines a procedure to merge the interacting (or sync) nodes of two collaborating processes. As described previously, the interactive merge is for merging the nodes that contain either *synchronous* or *asynchronous* interaction patterns.

Algorithm 4.3: Merge Interacting Nodes

Input : Two nodes m, c and the integrated process model Π^I

Output : Integrated process model Π^I with nodes m and c

```

1 Function MergeSyncNodes ( $m : \pi_m, c : \pi_c, \Pi^I$ ) :
2   if  $m \in \pi_m.A$  and  $c \in \pi_c.A$  then
3     if  $m \rightarrow c$  or  $c \rightarrow m$  then
4       if  $\rightarrow.IType = Sync$  then
5         if  $m \rightarrow c$  then
6           |  $SynchronousMerge(m, c, \Pi^I)$ 
7         else if  $c \rightarrow m$  then
8           |  $SynchronousMerge(c, m, \Pi^I)$ 
9         end
10      end
11      else if  $\rightarrow.IType = Async$  then
12        if  $m \rightarrow c$  then
13          |  $AsynchronousMerge(m, c, \Pi^I)$ 
14        else if  $c \rightarrow m$  then
15          |  $AsynchronousMerge(c, m, \Pi^I)$ 
16        end
17      end
18    else if ( $m \in \pi_m.A$  or  $\pi_m.G$ ) and  $c \in \pi_c.G$  then
19      |  $\Pi^I \leftarrow G_S$ 
20      |  $IterateOverGatewayNodes(m, c, \Pi^I)$ 
21      |  $\Pi^I \leftarrow G_M$ 
22    end
23 End Function

```

The $MergeSyncNodes()$ function takes a pair of sync nodes, m and c , and the

integrated process model Π^I as input. The dotted arrow between m and c represents their interaction, whose type (sync or async) can be obtained as defined in the algorithm (in line 4). This function receives either two activity nodes or two gateways (*AND* or *XOR*) or an activity node and a gateway that participate in the same interaction from Algorithm 4.1. In case of two activity nodes (lines 2-17), the algorithm identifies the type of interaction between them, and then invokes the corresponding merge function that is either *SynchronousMerge()* or *AsynchronousMerge()* defined in Algorithm 4.4 or 4.5.

When Algorithm 4.3 receives an activity and a gateway (lines 18-22), then the *IterateOverGatewayNodes()* function is used to iterate over the branches of the gateway to find the node that interacts with the activity and to call the *Merge()* function by passing the two interacting nodes. As defined in the algorithm, the split (G_S) node of the corresponding gateway must be first inserted into the integrated process model to which the merged fragments are added in sequence, then a merge (G_M) node is added. Any branch node that remains (for being single) after this iteration can be inserted between the split and merge nodes of the corresponding gateway in sequence. A similar approach is followed for merging the nodes of two gateways, where every pair of their branch nodes is merged and the resulting merged fragment is enclosed between the *split* and *merge* nodes of the corresponding gateway in the integrated process model.

A. Synchronous Merge: Non-Split and Split

As introduced in the above section, the synchronous *non-split* and *split* merge is based on the flow of artifacts between the nodes of two collaborating processes. This merge type mainly considers the artifacts and states exchanged between the sync nodes, and splitting or non-splitting the *sender sync node* based on the artifacts and states that it outputs or receives in return from the *receiver sync node*.

Algorithm 4.4: Synchronous Merge: Non-Split and Split**Input** : Two interacting nodes x, y and the integrated process model Π_I **Output** : Integrated process model Π^I with nodes m and c

```

1 Function SynchronousMerge ( $x, y, \Pi^I$ ) :
2   if  $x \rightarrow y$  then
3     if  $d_{out}(x) = d_{ret}(x)$  then
4        $p \leftarrow x.AddInSeq(y)$  ▷ Non-Split Merge
5        $AnnotateInput(x)$ 
6        $AnnotateIO(y)$ 
7       while  $\neg y \rightarrow x$  do ▷ if y is not a return node
8          $y_{next} = GetSeqNext(y)$ 
9          $p \leftarrow p \cup y.AddInSeq(y_{next})$ 
10         $AnnotateIO(y_{next})$ 
11         $y = y_{next}$ 
12      end
13    else ▷ Split Merge
14       $Split(x) \Rightarrow \{x^{pre}, x^{post}\}$ 
15       $p \leftarrow x^{pre}.AddInSeq(y)$ 
16       $AnnotateInput(x^{pre})$ 
17       $AnnotateIO(y)$ 
18      if  $y \rightarrow x$  then
19         $p \leftarrow p \cup y.AddInSeq(x^{post})$ 
20      else
21        while  $\neg y \rightarrow x$  do ▷ if y is not a return node
22           $y_{next} = GetSeqNext(y)$ 
23           $p \leftarrow p \cup y.AddInSeq(y_{next})$ 
24           $AnnotateIO(y_{next})$ 
25           $y = y_{next}$ 
26        end
27         $p \leftarrow p \cup y.AddInSeq(x^{post})$ 
28      end
29       $AnnotateIO(x^{post})$ 
30    end
31     $\Pi^I \leftarrow \Pi^I \cup p$ 
32  end
33 End Function

```

Algorithm 4.4 is for both *split* and *non-split* merging of *sync* nodes that have synchronous interaction patterns. According to the split condition defined in the algorithm (in line 3), the output data (artifacts and states) of sender sync node x is tested against the data (artifacts and states) returned by the receiver sync node y . If they are the same, then the algorithm follows a *non-split* merge approach, where the two sync nodes x and y are merged in sequence.

Otherwise, the algorithm follows a split type of merge, where the sender sync node x is split into x^{pre} and x^{post} , as it is satisfying the given split condition that the output data is different from the data that is returned by the receiver sync node y . In this case, as defined in the algorithm the receiver sync node y is added in sequence to the *pre* node x^{pre} of sender sync node x and then the post node x^{post} of sender sync node x is added in sequence to the receiver sync node y to have a sequence relation.

Algorithm 4.4 considers another case (see lines 17-24), where the receiver sync node is a subprocess, the first node y of this subprocess receives data from the sender sync node x , whereas the last node of this process returns the data back to the sender sync node x . In this case, the algorithm connects every next node y_{next} of the receiver sync node y in sequence to it, until the return node is reached.

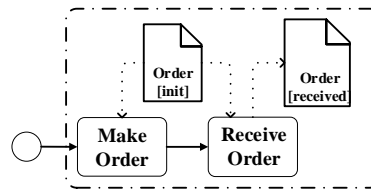


Figure 4.12: Synchronous Merge

Figure 4.12 presents an example for the synchronous non-split merge over a pair of interacting nodes *MakeOrder* and *ReceiverOrder* of the given AAIOPB. It can be seen in Figure 4.1, the output artifact and state of *MakeOrder* is the same as the one it receives from the *ReceiveOrder* in return. Therefore, the *MakeOrder* node is not required to be split according to the given algorithm, so these nodes are connected in

sequence, as shown in Figure 4.12.

To preserve the order of artifacts and states, Algorithm 4.4 implements the *AnnotateInput()* function to annotate only the input (artifacts and states) to the sender sync node, while the receiver sync node is annotated with both the input and output using the *AnnotateIO()* function in the resulting merged fragment as shown in the figure. It is worth noting that, the restriction on the input and output annotation of activity nodes is made by assuming that every artifact and state is an input and output to at least one activity node in the process model.

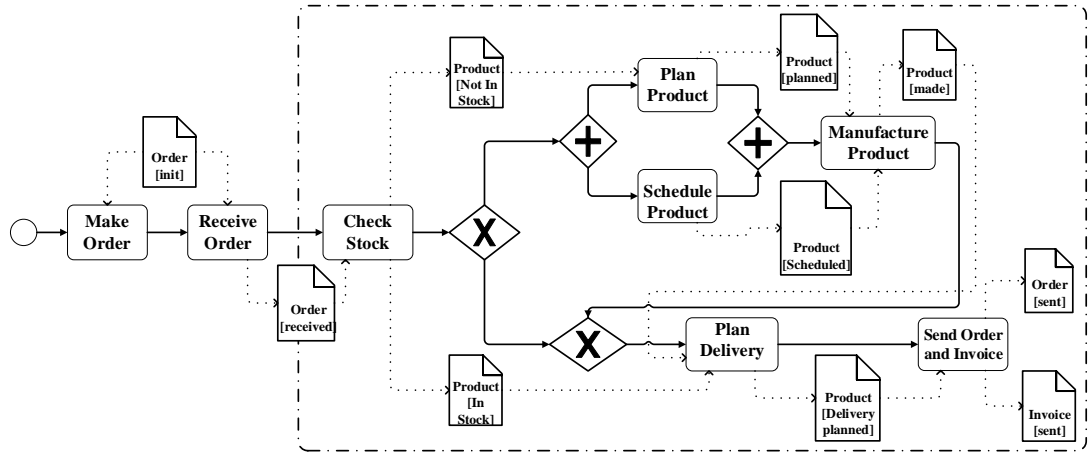


Figure 4.13: Synchronous Merge

Another example is presented in Figure 4.13, where the *sender sync node* (Make-Order) receives a return message from the (subprocess or sync set) next sequence flow node of the *receiver sync node*. Here, after merging the *receiver sync node* (Receive-Order), each next sequence flow node of this node is merged in sequence to the *sender sync node*, until the return node that is sending the return message is identified. Therefore, as defined in the algorithm, the merge is continued until the return node is found and attached in sequence to the *sender sync node*. Then, the next pair of nodes which is the next node (XOR gateway) of *MakeOrder* and the next node (ReceivePayment) of sync set (*ReceiveOrder*,...,*SendOrderandInvoice*) are considered for merging.

B. Asynchronous Merge: Non-Split and Split

The algorithm to merge two nodes based on their asynchronous interaction patterns is presented in Algorithm 4.5. The procedure of both the *non-split* and *split* merge is embedded as part of this algorithm. According to the *non-split* merge defined in the algorithm, the sender sync node x and the receiver sync node y must be connected in sequence. Similar to Algorithm 4.4, this algorithm also restricts the input and output annotation of these merged nodes using the *AnnotateInput()* and *AnnotateIO()* functions.

Algorithm 4.5: Asynchronous Merge: Non-Split and Split

Input : Nodes x, y , a new parallel gateway G^{AND} with G_S (split) and G_M (merge) nodes, and an integrated process model Π^I

Output : Integrated process model Π^I with G^{AND}

1 **Function** AsynchronousMerge (x, y, G^{AND}, Π^I) :

```

2   if  $x \rightarrow y$  then
3       if  $d_{out}(y) = d_{ret}(y)$  then
4            $p \leftarrow x.AddInSeq(y)$                                 ▷ Non-Split Merge
5           AnnotateInput( $x$ )
6           AnnotateIO( $y$ )
7            $\Pi^I \leftarrow \Pi^I \cup p$ 
8       else
9            $Split(y) \Rightarrow \{y^{pre}, y^{post}\}$                         ▷ Split Merge
10           $G_S.AddInSeq(x)$ 
11          AnnotateInput( $x$ )
12           $G_S.AddInSeq(y^{pre})$ 
13          AnnotateIO( $y^{pre}$ )  $x.AddInSeq(G_M)$ 
14           $y^{pre}.AddInSeq(G_M)$ 
15           $G_M.AddInSeq(y^{post})$ 
16          AnnotateIO( $y^{post}$ )
17           $\Pi^I \leftarrow \Pi^I \cup G^{AND}$ 
18      end
19  end

```

18 **End Function**

In case of the *split* merge, the receiver sync node is split into *pre* and *post* nodes,

as it satisfies the given split condition. When split, a new *parallel* gateway is used, where x and the *pre* node (y^{pre}) of y must be added in sequence to the *split* node (G_S) of this new parallel gateway. Then, the sequence flows are added from these nodes to the *merge* node (G_M) of the parallel gateway. The *post* node (y^{post}) is then added in sequence from this *merge* gateway node.

Figure 4.14 provides an example of the asynchronous split merge based on the motivating example. Figure 4.1 shows where the nodes *SendPayment* and *ReceivePayment* have an asynchronous interaction between them. Therefore, according to the given algorithm, the output artifact and state of *ReceivePayment* node is checked with the artifact and state it is receiving from *SendPayment*. These artifacts and states are different, meaning that the *ReceivePayment* node is satisfying the split condition, therefore the procedure for asynchronous split merge, as defined in the algorithm, is used to merge these nodes.

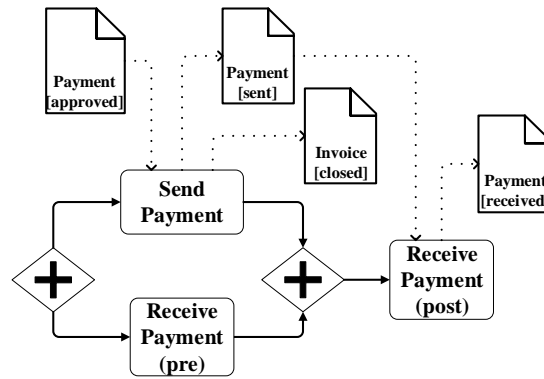


Figure 4.14: Asynchronous Split Merge

According to the asynchronous split merge defined in Algorithm 4.5, the *ReceivePayment* is split into *pre* and *post* as shown in Figure 4.14. Then the nodes *SendPayment* and the *pre* node of *ReceivePayment* are connected in parallel using the *split* node of new *parallel* gateway, then the sequence flows are added from these nodes to the *merge* node of the *parallel* gateway. The *post* node of *ReceivePayment* is then added in sequence to the *merge* gateway node.

4.5 Evaluation

In this section, a case study and the prototype implementation and evaluation of the proposed approach is discussed. This section also demonstrates how the proposed approach can preserve the structure and behaviour of the base process in the resulting integrated process model.

4.5.1 Case Study

An AirTravel process model was first utilized to study the feasibility of the proposed approach. This process model was also chosen from the BPMAI model collection, and it represents a collaboration among five entities, including: *Passenger*, *Airline Check-in Operator*, *Airline Departure-control System*, *Airline Service-desk Personnel* and the *Airport Service-desk Personnel* in fulfilling the passenger's travel request. The travel process is commenced, when a passenger simultaneously requests the airport check-in operator for an outgoing passenger card and the security immigration instructions, to collect the boarding pass. The airline check-in operator provides the requested information and simultaneously finalizes the check-in and registers the finalization in the airline departure-control system, which notifies the airline check-in operator of the status of completed check-in. After acquiring the completed check-in status, the airline check-in operator conducts the baggage check and provides the boarding pass if the baggage is under limit, otherwise instructing the passenger to proceed to an appropriate service counter. Then, the passenger either goes to the airport service-desk or the airline sales-desk or airport premium counter to receive an explanation for excess baggage payment and then proceeds to pay the excess baggage fee and to deposit the excess baggage. After paying the excess baggage fee, the passenger receives the payment receipt either from the airline or airport service-desk. This receipt is either submitted to airport service-desk personnel or the airline check-in operator to acquire the boarding

pass. Upon acquiring the boarding pass, the passenger proceeds to board the flight.

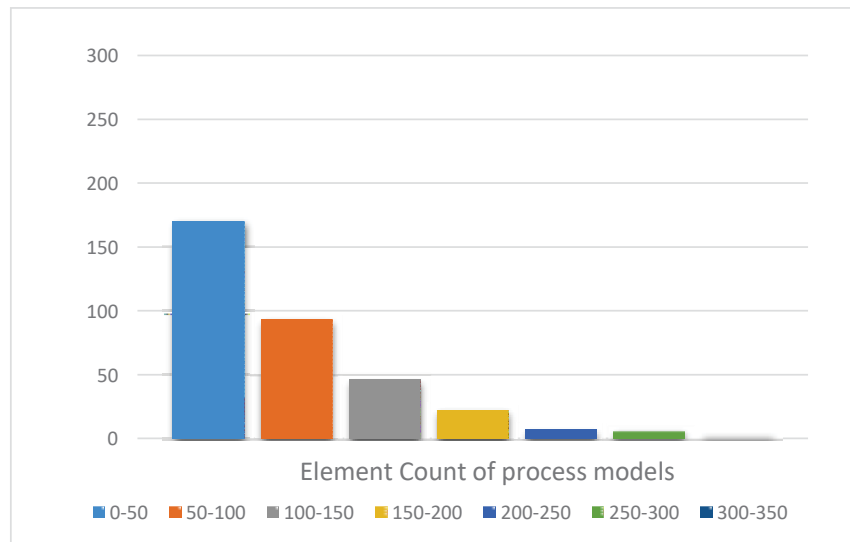


Figure 4.15: Histogram of Sorted BPMN Choreography Process models

Figure 4.15 presents the histogram of BPMN process models that were sorted to choose (from range 250-300) the AirTravel model for this study. This process model is large, therefore an excerpt of this model is presented in Figure 4.16 for ease of understanding and to demonstrate the merge over its collaborating processes. The complete model can be found in Figure A.5 of the Appendix. There are 8 artifacts involved in the travel process, including *Flight*, *Outgoing Passenger Card (OPC)*, *Security Immigration Instruction (SII)*, *Baggage*, *Check-in*, *Airline*, *Payment* and *Boarding Pass*. Some of these artifacts can be observed from the above process model, which are shared between the collaborating processes. As mentioned in the previous section, more than two collaborating processes are merged in a two by two manner. Therefore, only a pair of collaborating processes are merged in every iteration. As described above, initially the *passenger* and the *airline check-in operator* interact in a synchronous way, where the passenger provides the flight details to request some documents necessary to acquire the boarding pass and the operator provides the requested documents. Therefore, the

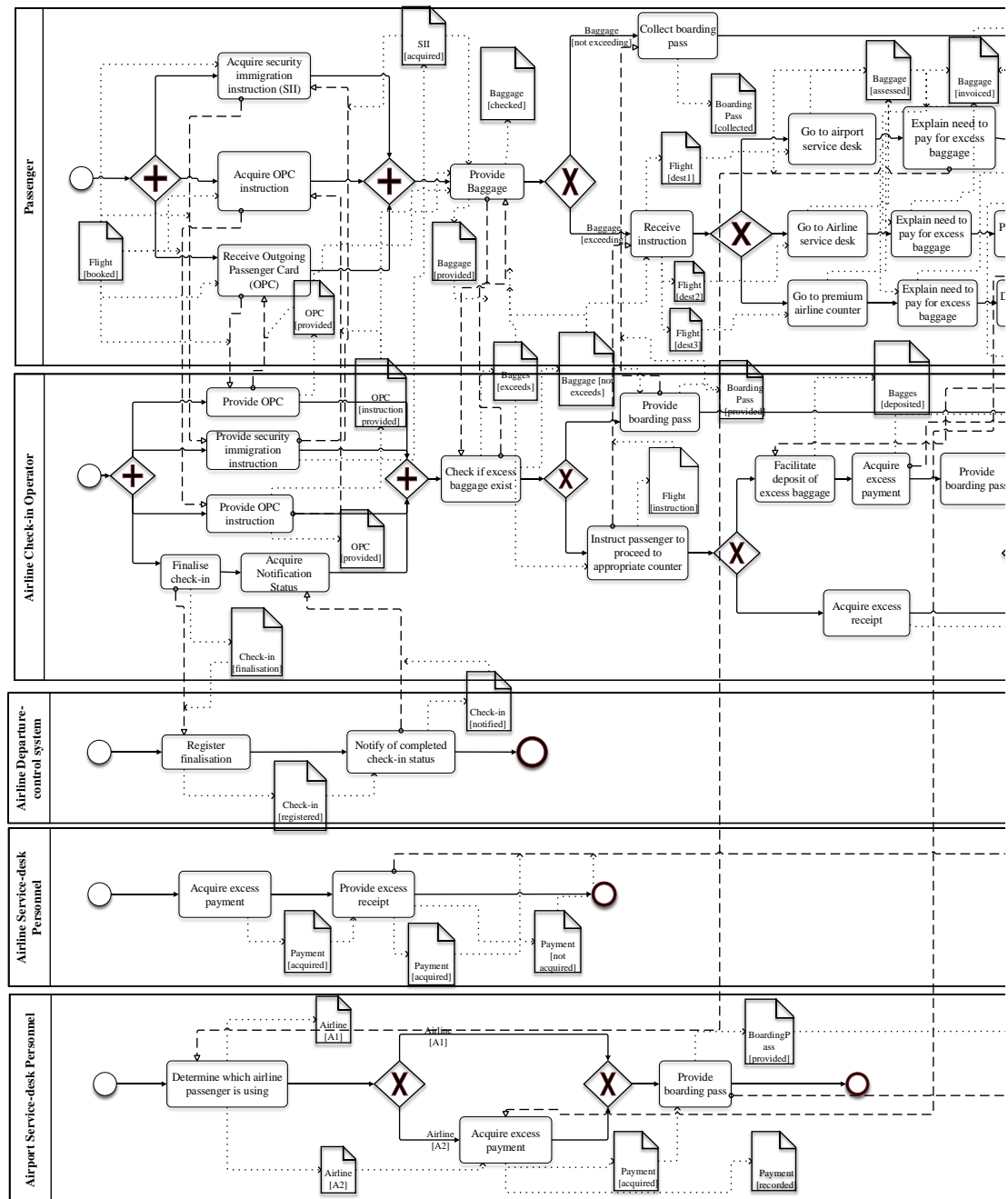


Figure 4.16: Excerpt of AirTravel Process Model

interacting nodes of these two collaborating processes participating in the synchronous interaction are merged based on the rules (4.2 or 4.3 or 4.4) formulated for the synchronous merge.

According to the proposed algorithms, after merging the *start* nodes of the *Passenger* and *Airline Check-In Operator* processes, the parallel (AND) gateways of these processes are considered for merging. As they both have interactions from their branch nodes, each of their branch nodes are again considered in a pairwise manner for merging. For example, the sender sync node *AcquireSecurityImmigrationInstruction* is connected in sequence to the receiver sync node *ProvideSecurityImmigrationInstruction* following the synchronous merge rules (4.2 or 4.3), as they have a synchronous interaction between them. Here the sender node is not split into *pre* and *post* nodes, as its output artifact (SII) and its state (issued) is same as the one (SII[issued]) it received from the receiver sync node. Similarly, the other two interacting nodes *AcquireOPCInstruction* and *ReceiveOPC* are also not split, as they share the same artifacts and states. Thus, each node that belongs to the AND gateway of the *Operator* process that interacts with the nodes of *Passenger* process are connected in sequence based on the rule defined for synchronous non-split merge (4.2). The above process model shows that the last branch of the AND gateway in the *Operator* process has no corresponding branch in the AND gateway and no interactions with the nodes of *Passenger* process, therefore in this case a new branch is added to the corresponding AND gateway in the integrated process model to which the two nodes *FinaliseCheck-in* and *AcquireNotificationofStatus* are added in the same sequence relation. As described in the previous section, each pair of these merged nodes is enclosed between the *split* and *merge* nodes of the AND gateway in the integrated process model, as they belong to the same type of gateway in the base collaborating processes.

After merging the two parallel (AND) gateways, the next nodes *ProvideBaggage* and *CheckIfExcessBaggageExists* are merged using the synchronous split merge rule (4.4), as

the artifact *Baggage* and its state *checked* is not part of the returned message. Therefore, *ProvideBaggage* is split into *pre* and *post* nodes, where the *pre* node is only annotated with input artifacts and states, whereas the *post* node is annotated with all the output artifacts and states of this node. Then the XOR-Split gateways of the two processes are chosen for merging, where the two split nodes are first merged and added into the integrated process model. Then the merge is repeated for each pair of their branch nodes. Therefore, the non-interacting nodes are merged in sequence, as discussed above, and for each interacting node of one gateway, the corresponding collaborating node is identified by iterating over the branches of another gateway in the collaborating process, and are merged according to their interaction type. The asynchronous interaction between the nodes *CollectBoardingPass* and the *ProvideBoardingPass* can be observed, where these nodes are merged in sequence according to the asynchronous non-split merge rule (4.5). Similarly, the *ReceiveInstruction* and *InstructPassenger* are merged according to the asynchronous split merge rule (4.6), as the *ReceiveInstruction* has the *Flight* artifact with states *dest1,dest2,dest3*, which the *InstructPassenger* node has not returned. In this manner, starting at the *start* nodes of two collaborating processes, the merge continues until the *end* nodes of those processes are reached and merge into the integrated process model.

The resulting integrated process is then merged with the *Airline Departure-control System* process, as this process first interacts with the *Operator* process in the main *Air-Travel* process. Therefore, the nodes *RegisterFinalisation* and *NotifyCompletedCheck-inStatus* are merged with the *FinaliseCheck-in* and *AcquireNotificationStatus* nodes using the asynchronous non-split merge rule (4.5), as they have the asynchronous interactions and do not satisfy the split condition. In a similar manner, the nodes of the remaining collaborating processes *Airline Service-desk Personnel* and *Airport Service-desk Personnel* are also merged with the nodes of the above-resulting integrated process model to obtain the complete integrated process model of the *AirTravel* process model.

4.5.2 Implementation

The proposed algorithms were developed into a prototype tool that supports the automatic merging of the collaborating processes of inter-organizational business processes (IOBPs). The tool is developed based on Java concepts including Swings and AWT. XML is used for specifying the IOBPs, while JAXP API is used in parsing these process specifications and to generate an XML representation of the integrated process model. The XML structure of the Buyer-Seller IOBP and its integrated process model generated by using the proposed prototype, can be observed in Figure 4.17. A web-based visualization tool (Ltd, 2002) is used to visualize the integrated process model in Figure 4.18, as the current version of the proposed prototype supports this feature partially.

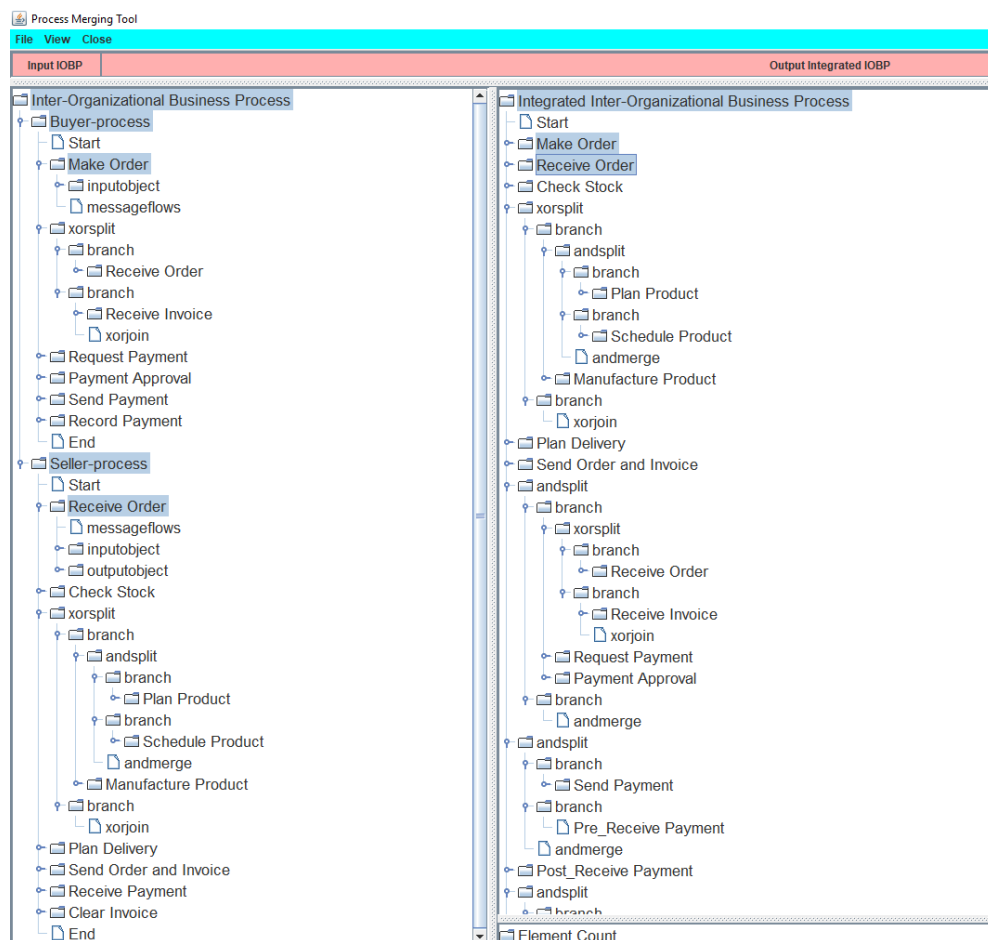


Figure 4.17: Buyer-Seller IOBP and its Integrated Process Model

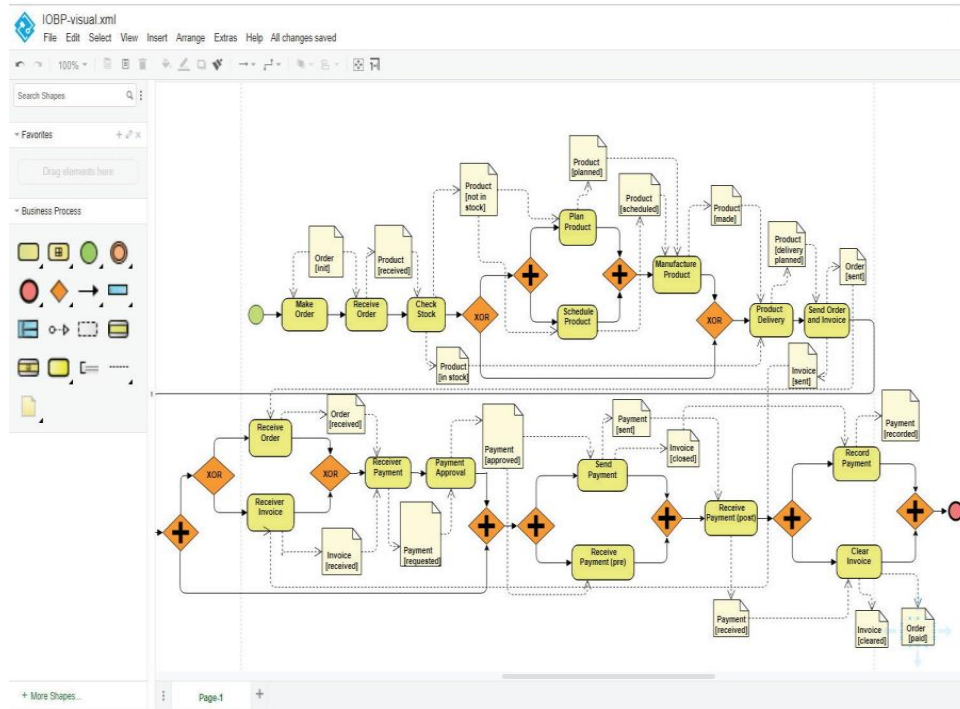


Figure 4.18: Integrated Process Model of Buyer-Seller IOBP

While the main objective of this research is to generate the lifecycles of artifacts from the AAIIOBP, the extension of this merge tool can be proposed, where this tool can be extended with the synthesis tool that implements the synthesis approach presented in Chapter 3. In this regard, a prerequisite is suggested, to merge the collaborating processes of an AAIIOBP in order to generate the lifecycles of artifacts pertinent to that process model. This can be achieved, as the resulting integrated process model also shares the similar structural and behavioural properties like a standard process model. Therefore, the synthesis tool can be used to generate the artifact lifecycles from these integrated process models.

4.5.3 Experimental Discussion

To evaluate the applicability of the proposed approach, experiments have been conducted on the merge prototype using a small set of inter-organizational business process models

presented in Table 4.2, each of which represent the processes of different business domains. These process models also vary in the number of collaborating processes, their nodes and the involved artifacts. This set also includes the Buyer-Seller IOBP and the AirTravel process model utilized to study the feasibility of the proposed approach.

Table 4.2: Data Set

IOBP Domain	Collaborating Processes	Nodes	Artifacts
Buyer-Seller	2	26	3
Job Recruitment	3	46	5
Ticket Booking	3	75	4
PayPal Invoice	4	42	7
Air Travel	5	66	8

One of the major requirements in process merging is to preserve the structural and behavioural aspects of base process models in the resulting integrated process models. Therefore, two theorems are proposed and proved to demonstrate that the proposed approach can achieve this structural and behavioural consistency. The theorems are proved constructively based on the definitions and rules presented in Sections 4.3 and 4.4 and follow the proof style as in (S. Sun et al., 2006).

Theorem 1. *If two structured process models are merged based on the parallel, synchronous and asynchronous merge rules, the resulting integrated process model must also be a structured process model.*

Proof. The proof is by construction and relies on Definition 8, where the different forms of structured process models are explained. All the merging rules formulated above are based on the notion of combining structured process models to create integrated process models that are also structured. Therefore, the merge here is performed by taking a structured region of one process model, and combining it with the structured region of another process model to result in a new structured region. The proof is derived from a

case-by-case analysis.

Case (a) Parallel Merge. When two structured process models are merged in parallel based on rule 4.1 using the *split* and *merge* gateway nodes, the resulting process model is also structured, according to Definition 8(2).

Case (b) Synchronous Merge. When two process models are merged by synchronous rules 4.2, 4.3 and 4.4, a structured process fragment is connected in sequence to another structured process fragment. Since each fragment is structured, the resulting merged process is also structured, based on Definition 8(3).

Case (c) Asynchronous Merge. When two structured process fragments are combined according to rule 4.5 in sequence or rule 4.6 in parallel using the *split* and *merge* gateway nodes, the resulting workflow is also structured, according to Definitions 8(3) and 8(4).

Thus, the merged process fragment or process model resulting from each type of merge is structured, if the corresponding merge regions themselves are structured. \square

Theorem 2. *If the merged fragments of two structured process models are structured, then the integrated process that results from the rules defined for parallel, synchronous and asynchronous merge notions is well-behaved.*

Proof. The proof follows from Theorem 1, and Definition 9, which states every structured process model is well-behaved. Thus, we conclude that this theorem is true. \square

Based on the results of the experiments, it can be concluded that the proposed approach guarantees the structural and behavioural aspects of IOBP models in the resulting integrated process models, as long as each of their collaborating processes are structured and well-behaved and their interactions are non-overlapping. For example, this can be observed from the integrated process model of the Buyer-Seller IOBP presented in Figure 4.18, which preserves the structure and behaviour of two collaborating processes of the Buyer-Seller IOBP given in Figure 4.1.

4.5.4 Performance Analysis

The dataset presented in Table 4.2 is used to analyze the performance of the proposed merge algorithms. As described above, this set represents process models from different business domains that vary in the number of collaborating processes, elements (nodes, artifacts and states) and their interactions(or message flows). While the proposed merge algorithms are interdependent, their overall average execution times were recorded, which can be seen in the graph presented in Figure 4.19.

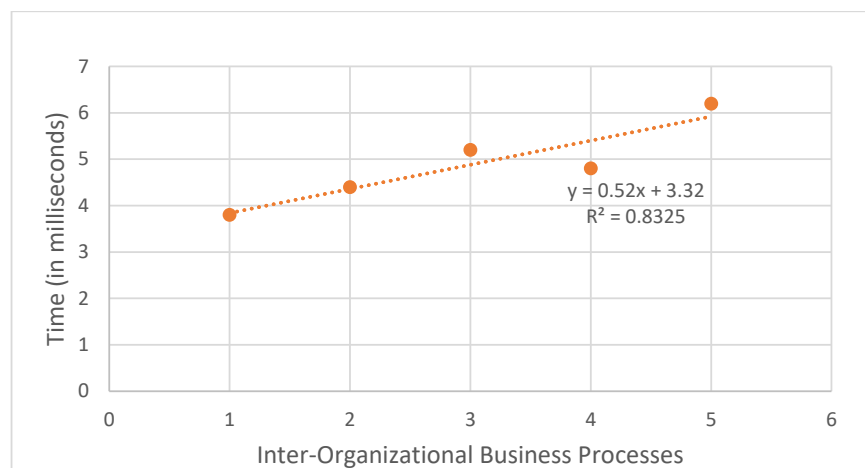


Figure 4.19: Execution times of Merge algorithms

The graph illustrates the execution times of the merge algorithms over the given process models. It can be observed that the execution times of the merge algorithms depend on different factors, such as the number of collaborating processes, nodes and their interactions. From the graph, one may interpret that the execution times of these algorithms increase with an increase in process complexity. However, it is anticipated that the experiments conducted using the given dataset are not sufficient to draw conclusions over the performance of the proposed algorithms.

4.6 Discussion and Related Work

Modeling business processes from *artifacts* and their interacting *lifecycles* is the core objective of an artifact-centric modeling paradigm. As discussed above, the existing transformation approaches have focused on synthesizing the artifact-centric counterparts (lifecycles) from standalone activity-centric process models, while ignoring the process models that represent business processes which cross organizational boundaries, such as IOBP models. Therefore, this chapter proposed an approach to support the transformation of activity-centric IOBP models into artifact-centric process models. The proposed approach extends the scope of the synthesis solution presented in chapter 3 to the activity-centric IOBP models.

The solution proposed in this chapter is based on our work (Kunchala et al., 2017), which first introduced the merge concepts for IOBP models. However, this chapter compliments the existing work with the formalization of merge concepts, proposing algorithms for automatic merging and also performing a thorough evaluation using theorems and prototype implementation and experiments (Kunchala et al., 2019).

The proposed approach is related to existing research on *process merging* and *consolidation*. As discussed in Chapter 2, although there exists several merge approaches (S. Sun et al., 2006; Mendling & Simon, 2006; Gottschalk et al., 2008; La Rosa et al., 2010; Bulanov et al., 2011; La Rosa et al., 2013), they consider standalone process models with common process fragments for merging. In addition, none of them have been proposed for merging the collaborating processes of inter-organizational business process models that contain artifact data flows. Therefore, when compared to these works, the merge approach presented in this chapter is novel, as it does not require any common process fragments and can merge collaborating processes by considering the involved artifacts and states while also preserving their structure and behaviour of base processes in the resulting integrated process.

4.7 Summary

In this chapter, the second research question (RQ2) is addressed by presenting an approach that aims to merge the collaborating processes of IOBP models for the purpose of generating artifact lifecycles from the resulting integrated process model. The approach consists of a set of algorithms that enables the automatic process merging. The algorithms were implemented and evaluated using a subset of IOBP models to demonstrate the feasibility and applicability of the proposed approach. Moreover, the structural and behavioural requirement for the resulting integrated process model was validated using theorems. To the best of our knowledge, the proposed approach makes the first contribution towards merging the collaborating processes of IOBP models that builds on artifact data flows.

Chapter 5

Constructing Activity-Centric Process Models from Artifact-Centric Process Models

In recent years, many organizations have integrated their business processes with web services to address the growing needs of their potential customers. In this regard, process models present the design view of business processes, whereas web services correspond to their executable view. Intuitive process models are of higher priority for organizations to understand, document, communicate and improve their business strategies. There exists a multitude of intuitive graphical notations, including BPMN for modeling activity-centric business processes, where the resulting process models can be naturally mapped to executable languages such as BPEL to achieve improved efficiency.

On the other hand, artifact-centric process models represent the flow of artifacts, where the invocation of activities on the associated artifacts can be restricted using constraints, such as business rules (Yongchareon & Liu, 2010). Flexibility is the major advantage of such declarative approaches (Caron & Vanthienen, 2016), as they do not

enforce specific execution flows, thus offering more freedom to design and modify the resulting process models. However, the drawback of such declarative modelling is that it is difficult to understand the structure and flow of these process models compared to activity-centric process models. Therefore, a reverse transformation is proposed, where an approach to transform artifact-centric process models into activity-centric process models is presented. The proposed approach consists of algorithms to achieve this objective and also to analyze the consistency between both the base and the constructed process models (Kunchala et al., 2020). A case study is used to demonstrate the feasibility of the proposed approach, while prototype implementation and evaluation is used to demonstrate the applicability of the proposed approach.

The remaining chapter is organized into seven sections. Section 5.1 introduces a motivating example based on an Order Processing Scenario. Section 5.2 formulates the research problem and defines some key notions. Section 5.3 presents an overview of the proposed transformation approach. Section 5.4 presents algorithms for the proposed approach. Section 5.5 discusses the implementation and evaluation of the proposed approach. Section 5.6 discusses and reviews the related work. Section 5.7 summarizes the contributions of this chapter.

5.1 Motivating Example

In this section, an artifact-centric process model is presented, which describes a customer order processing scenario, which is closely related to the one used in (Meyer & Weske, 2013), to demonstrate the proposed approach. The order process in Table 5.1 is described based on the ACP model introduced in (Yongchareon & Liu, 2010), which consists of three core constructs, including: *artifacts*, *services* and *business rules*. Artifacts represent key business entities, which contain a finite set of attributes and states. The *Order*, *Product* and *Invoice* are the three artifacts of the given process model that have a

Table 5.1: Artifact-centric Process Model (ACP Model)

Artifacts: Order, Product, Invoice
Activities: InitiateOrder (A), ReceiveOrder (B), AnalyseOrder (C), ConfirmOrder (D), CheckStock (E), PlanProduct (F), ScheduleProduct (G), ManufactureProduct (H), ShipProduct (I), SendInvoice (J), ReceivePayment (K), CloseOrder (L)
Business Rules: R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11
R1: Initiate order request Pre-condition: λ (Order, wait) Activities: InitiateOrder (Order) Post-condition: λ (Order, init)
R2: Receive order from the customer Pre-condition: λ (Order, init) Activities: ReceiveOrder (Order) Post-condition: λ (Order, received)
R3: Analyse the customer order Pre-condition: λ (Order, received) Activities: AnalyseOrder (Order) Post-condition: λ (Order, confirmed) \vee λ (Order, rejected)
R4: Confirm the customer order Pre-condition: λ (Order, confirmed) Activities: ConfirmOrder (Order) Post-condition: λ (Product, init)
R5: Check stock for the product Pre-condition: λ (Product, init) Activities: CheckStock (Product) Post-condition: λ (Product, in stock) \vee λ (Product, not in stock)
R6: Plan and schedule for manufacturing the product Pre-condition: λ (Product, not in stock) Activities: PlanProduct (Product) ScheduleProduct(Product) Post-condition: λ (Product, planned) \wedge λ (Product, scheduled)
R7: Manufacture the product Pre-condition: λ (Product, planned) \wedge λ (Product, scheduled) Activities: ManufactureProduct (Product) Post-condition: λ (Product, made)
R8: Ship product to the customer Pre-condition: λ (Product, in stock) \vee λ (Product, made) Activities: ShipProduct (Product, Order) Post-condition: λ (Product, shipped) \wedge λ (Order, shipped) \wedge λ (Invoice, init)
R9: Send invoice to the customer Pre-condition: λ (Product, shipped) \wedge λ (Order, shipped) \wedge λ (Invoice, init) Activities: SendInvoice (Order, Invoice) Post-condition: λ (Order, invoiced) \wedge λ (Invoice, sent)
R10: Receive payment from the customer Pre-condition: λ (Order, invoiced) \wedge λ (Invoice, sent) Activities: ReceivePayment (Order, Invoice) Post-condition: λ (Invoice, paid) \wedge λ (Order, paid)
R11: Close the customer order Pre-condition: λ (Order, rejected) Activities: CloseOrder (Order) Post-condition: λ (Order, closed)

finite set of attributes and states. The attributes are not specified in this process model, as we consider them at the implementation level. However, an example of attributes such as *OrderID* and *CustomerID* of *Order* artifact, which are defined in parallel to its states.

Services represent business activities or tasks that take some artifacts in some states as input, and output some artifacts and/or modify their attributes and/or states. The given process model has a set of activities, each of which are labelled with an alphabet to easily refer to them throughout the discussion. Business rules are for associating artifacts with activities that read/update the attributes and states of these artifacts. A business rule specifies conditions (pre, post) for an activity to invoke on the associated artifact(s) by following the declarative Condition-Action style. Rules are used to prescribe, under what pre-condition the activities are executed on the associated artifacts and the post-condition that the activities have to satisfy, after their execution. In the given process model, the complete set of business rules specify the control flow of ordering process from its initiation to the completion.

It can be observed from Table 5.1, the process starts with the initiation of an order request by the customer. After receiving the order request for a product, the customer information is analyzed to decide whether to confirm the order or to reject it, in which case the corresponding order is immediately closed, and the process is terminated. When confirmed, the order is analyzed to check the availability of the ordered product. The order is shipped to the customer if the requested product is in stock, otherwise the product is planned, manufactured and then shipped. Then, it is the time to invoice the order and send that to the customer for payment. In this case, the process is terminated after receiving the payment from the customer.

Table 5.2 presents the process log that records the execution traces of the artifact-centric process model presented in Table 5.1. These traces can be the recorded traces during the process execution or manually defined traces by analyzing the process model.

Table 5.2: ACP Log

Log Traces	Business Rules	Activities
1	R1R2R3R4R5R6R7R8R9R10	ABCDEFGHIJK
2	R1R2R3R4R5R8R9R10	ABCDEIJK
3	R1R2R3R11	ABCL

The process log is compliant with the given model, meaning that it contains every possible execution trace of the ACP model. Each process trace represents the execution sequence of business rules i.e., the activities and their associated input (pre) and output (post) artifacts. This process log is utilized to check the consistency between the ACP model and the activity-centric process model constructed from this model.

5.2 Problem Statement and Definitions

In this section, the research problem targeted in this chapter is first stated and then some of the key notions of proposed solution are formally defined, which include: *Artifact-Centric Process Model*, *Activity-Centric Process Model*, *Process Log* and *Trace Match*.

PROBLEM (PROCESS MODEL CONSTRUCTION). Given an artifact-centric process model, an activity-centric process model that preserves the behaviour of base process model must be constructed.

Here, an artifact-centric process (ACP) model is defined that is closely related to existing definitions in (Yongchareon & Liu, 2010; Meyer & Weske, 2013).

Definition 21. (Artifact-Centric Process (ACP) Model). An ACP model Π is a three tuple (Z, V, B) where,

- Z is a finite set of artifacts. Every artifact has a set of attributes, set of states S , an initial state and a set of final states;

- V is a finite set of activities. Every activity contains a label and a finite set of artifacts manipulated by this activity;
- B is a finite set of business rules. Every rule consists of a pre-condition α , post-condition β and activities that manipulate the associated artifacts. In addition, the *pre- and post-conditions* comprises a set of in-state (denoted with λ) and defined (denoted with δ) functions connected by logical operators \wedge, \vee .

The logical operators can reveal the relation between artifacts or dependencies between their states, when used between different artifacts or the states of same artifact that correspond to either a *pre- or post-condition* of a business rule. From such a relation/dependency, the control flow relation of activities (executed in parallel or in exclusive) that take these artifacts and states as input or produce them as output, can be derived.

For example, the states *confirmed* and *rejected* of *Order* artifact have an exclusive relationship between them as they are connected using the \vee operator, therefore the corresponding activities *ConfirmOrder* and *CloseOrder* must also execute in this exclusive manner. Similarly, the states *planned* and *scheduled* of *Product* artifact have a parallel relationship, as they are connected using the \wedge operator, thus the corresponding activities *PlanProduct* and *ScheduleProduct* must also execute in parallel.

The modeling elements of BPMN graphical notation are used to represent the constructed activity-centric process model. Therefore, the following definition is based on this notation.

Definition 22. (Activity-Centric Process (or ACT) Model). An activity-centric process model denoted with $\Pi^P = (E, A, G, F, D, S_p, I, O)$, where,

- E is a finite set of event nodes. Every event is in $E.Type = \{Start, End\}$;
- A is a finite set of activity nodes;
- G is a finite set of gateways. Every gateway is in $G.Type = \{XOR, AND\}$; and

G^{Open} and G^{Close} are used to represent opening and closing gateway nodes;

- $F \subseteq (E \cup A \cup G) \times (E \cup A \cup G)$ is the set of sequence flow relations among event, activity, and gateway nodes;

- D is a finite set of artifacts;

- S_p is a finite set of states;

- $I = D \times S_p \times A$ is the input relation among artifacts and activities;

- $O = A \times D \times S_p$ is the output relation among artifacts and activities;

The *events*, *activities* and *gateways* of the activity-centric process model are referred to as nodes, to differentiate them from the modeling constructs of an ACP model.

Definition 23. (Process Log). A process log P is a finite set of execution traces T_E , where E is the finite set of activities of P . Every trace $t_e = e_1, e_2, \dots, e_n$ is a finite non-empty sequence of activities, where $e_i \in E$ and $t \in T_E$.

Definition 24. (Trace Match). Let $t \in \Pi.T$ and $t_p \in \Pi_P.T_A$, we say t matches t_p that is $t \cong t_p$ if for every $a_i \in t$, $\exists a_j \in t_p$ and $a_i = a_j$. The \cong symbol is used to represent the consistency between two process traces or two process models.

5.3 Approach Overview

In this section, an overview of the proposed transformation approach is presented. The approach has two phases: *model construction* and *model consistency checking*. As shown in Figure 5.1, Step 1 corresponds to the first phase, whereas Step 2 and Step 3 correspond to the second phase of the proposed approach.

In the first phase, an activity-centric process model is constructed from the artifact-centric process model presented in Table 5.1, by obtaining the activities and associated artifacts with states from the business rules and arranging them based on their input and output artifact dependencies. In the second phase, the execution traces of the resulting

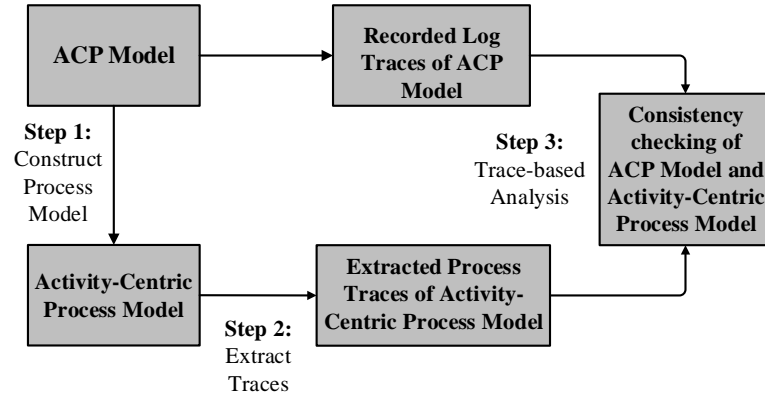


Figure 5.1: Transformation Approach

activity-centric process model are extracted and matched with the traces recorded in the execution log of the ACP model given in Table 5.2, to check the consistency between the ACP model and the constructed ACT model.

5.4 Algorithms

In this section, algorithms are presented for each phase of the proposed transformation approach and demonstrated using the motivating example presented in Table 5.1.

5.4.1 Model Construction

Algorithm 5.1 and 5.2 define the procedure to construct an activity-centric process (ACT) model from an artifact-centric process (ACP) model. It is noted that Algorithm 5.2 is a part of Algorithm 5.1, though the model construction procedure is divided into two parts for ease of understanding. The *ConstructACTModel()* function defined in Algorithm 5.1 is recursively invoked for every business rule of the ACP model, from which the activities and the associated input and output artifacts and states are retrieved to construct the ACT model.

Algorithm 5.1: Construction of ACT Model from the ACP Model**Input** : ACP (Π), ACT Model (Π^P) with start node (n)**Output** : Complete ACT Model (Π^P)

```

1 Function ConstructACTModel (rule :  $\Pi$ , n :  $\Pi^P$ , cond : rule) :
2   if rule  $\neq$  null then
3     if rule is not marked as completed then
4       if rule. $\alpha$  = cond then
5         AddActivities(rule,  $\Pi^P$ );
6         cond  $\leftarrow$  rule. $\beta$ 
7       else if rule. $\alpha \subset$  cond then
8         foreach z.s  $\in$  cond do
9           r  $\leftarrow$  GetRule(rule, cond)
10          while r  $\neq$  null do
11            if z.s  $\in$  r. $\alpha$  then
12              ConstructACTModel(r,  $\Pi^P$ , r. $\alpha$ )
13            end
14            r  $\leftarrow$  GetNext(r)
15          end
16        end
17      else if cond  $\subset$  rule. $\alpha$  then
18        foreach z.s  $\in$   $\alpha$  do
19          if  $\exists d.s^p \in \Pi^P$  then
20            if  $|d.s^p| = |z.s|$  then
21              if  $d.s^p \in \Pi^P.n_i$  then
22                if  $n_i \in G.branch_i$  then
23                  MergeBranches(G)
24                end
25                ConstructACTModel(rule,  $\Pi^P$ , rule. $\alpha$ )
26              end
27            end
28          end
29        end
30      end
31    end
32    ConstructACTModel(GetNext(rule),  $\Pi^P$ , cond)
33  end
34 End Function

```

The *ConstructACTModel()* function takes as input the ACP model, an activity-centric process (ACT) model with a *start* node and a data condition (*cond*), based on which a business rule of ACP model is invoked. Initially, the pre-condition of the first

business rule is assigned to the *cond* (data condition) and is updated with the pre- or post-condition of the corresponding business rule in each invocation of this function.

Algorithm 5.1 defines three conditions, where every business rule needs to satisfy one of these conditions in order to be invoked by the *ConstructACTModel()* function. Initially, as defined in line 3, a business rule is invoked only if it is not marked as completed, meaning that the activities of this rule were not added in the ACT model. According to the three conditions defined in the algorithm (in line 4, 7 and 17), a business rule is invoked if its pre-condition is the same as the given *cond*, or it is a subset of *cond* meaning that the pre-condition has one or more artifacts and states as the *cond*, or the *cond* itself is a subset of the pre-condition of that business rule.

As in line 5, the *AddActivities()* function defined in Algorithm 5.2 is called, when the pre-condition of a business rule matches the given data condition (*cond*). This function is used for adding all the activities of the corresponding business rule with their associated input and output artifacts and states into the ACT model. For example, to construct the ACT model from the ACP model presented in Table 5.1, Algorithm 5.1 starts at business rule R1 of this process model, as its pre-condition matches with the given *cond*, the *AddActivities()* function (defined in Algorithm 5.2) is called by passing the business rule R1 and the ACT model (that only contains the *start* node). The *AddActivities()* function adds the *InitiateOrder* activity of R1 in sequence with the *start* node of the ACT model with the associated input and output artifact *Order* and its states *wait* and *init*. Algorithm 5.2 also uses the gateway nodes (XOR, AND) to split and merge the control flow. For example, an XOR gateway is used when a rule contains an activity whose output has an artifact with more than one state, in which case the activities of other business rules that use the same artifact and states as input are added to different branches of the XOR gateway. Similarly, an AND gateway is used when a rule has more than one activity that may use the same artifact(s) and state(s) as input, in which case these activities are added to different branches of the AND gateway to

execute in parallel.

After completing the rule R1, the *cond* is updated with the post-condition of R1 as in line 6 of Algorithm 5.1, and the *ConstructACTModel()* function is called for the next rule R2 (line 32). Therefore, the pre-condition of R2 is checked with the *cond*, as there is a match, the node *ReceiveOrder* is added to the ACT model with its associated input and output artifacts and states using the *AddActivities()* function defined in Algorithm 5.2. Similarly, the activity *AnalyzeOrder* of rule R3 is added, as its pre-condition satisfies the post-condition (*cond*) of R2. These nodes can be seen in Figure 5.2, which illustrates the ACT model constructed using Algorithm 5.1 and 5.2.

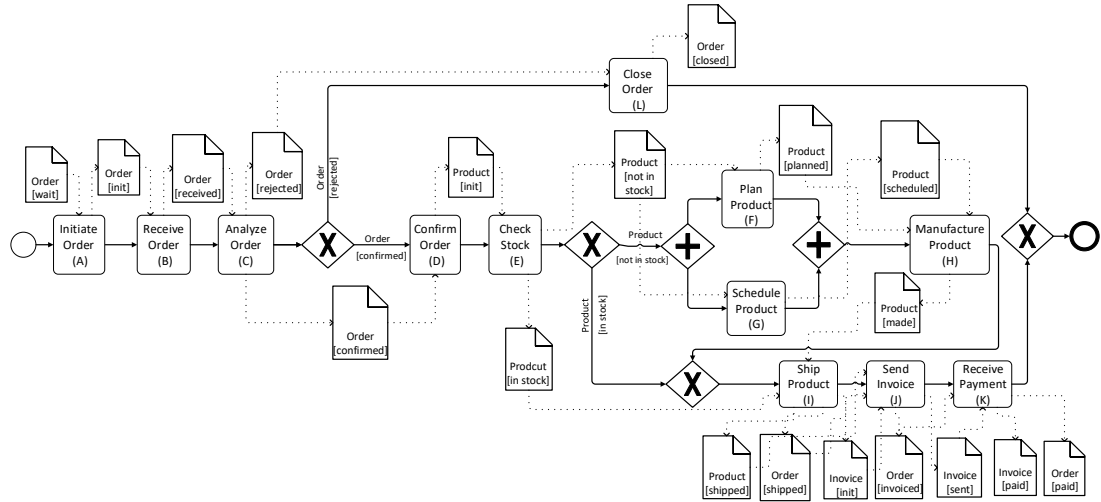


Figure 5.2: The constructed Activity-Centric Process Model

In case the pre-condition of a business rule is a subset of *cond* (line 7), then for one or more artifact(s) and state(s) of this *cond*, every next rule is traversed to check if its pre-condition matches with the artifact(s) and state(s) of the given *cond*. Therefore, the *ConstructACTModel()* function is called for every such rule whose pre-condition is a subset of *cond*, as in line 12. For example, the business rule R4 satisfies the condition, where the pre-condition of this rule is a subset of *cond* that contains the post-condition of rule R3. In this case, as defined in the algorithm (from line 8 to 16), for every artifact and state of *cond* for instance (Order, confirmed), the next rules of R3 are traversed to

find rule R4, which contains this artifact and state in its pre-condition and the function *ConstructACTModel()* is invoked for this rule.

As defined in line 17, the third condition is where the *cond* is a subset of the pre-condition of a business rule, then the algorithm checks that for every artifact and state of this pre-condition, there an artifact and state associated with an activity node in the ACT model. If these artifacts and states are associated with different activity nodes that belong to different branches of a gateway (XOR or AND), then all its open branches are merged using the *MergeBranches()* function before adding the activities of this business rule into the ACT model. The *MergeBranches()* function adds a sequence flow from each branch (activity) node to the close gateway node. Then the recursive function is called with this rule and its pre-condition, including the ACT model. For example, rule R8 satisfies such a condition, where *cond* that holds (Product, made) from the previous invocation is a subset of the pre-condition of rule R8. Therefore, here the algorithm checks if the other artifacts and states of the pre-condition already exist in the ACT model. If they exist, the algorithm checks if these are associated with different activity nodes that belong to an open gateway. In this case, the algorithm first merges those open branches and then calls the *ConstructACTModel()* function for adding the activities of R8 in sequence with this gateway node. Otherwise, it simply calls this recursive function with the same business rule and its pre-condition without merging the branches of that open gateway. In this manner, Algorithm 5.1 continues until every following rule is invoked and completed.

Algorithm 2 defines the *AddActivities()* function for adding the activities of each business rule into the ACT model and also for annotating their input and output artifacts and states. Therefore, the input to this function is a business rule of the ACP model and the ACT model. This function is called from Algorithm 5.1, when the pre-condition of a business rule satisfies the given data condition as described above.

Algorithm 5.2: Construction of ACT Model from the ACP Model

Input : A business rule of ACP (Π) Model and the ACT Model (Π^P)
Output : Activity-centric process model (Π^P) with new nodes

```

1 Function AddActivities (rule :  $\Pi$ , n :  $\Pi^P$ ) :
2   if  $|rule.v| = 1$  then
3     Function AnnotateIO (v : rule) :
4        $n \leftarrow v$ 
5        $\Pi^P.AddInSequence(n)$ 
6        $n.AnnotateInput(rule.\alpha)$ 
7        $n.AnnotateOutput(rule.\beta)$ 
8       if  $z.s \in v.O$  and the states of z are in exclusive relation then
9          $\Pi^P.AddInSequence(XOR.G^{Open})$ 
10        foreach  $z.s \in v.O$  do
11           $G^{Open}.AddBranch(branch)$ 
12           $branch.AddCondition(z.s)$ 
13           $ConstructACTModel(GetNext(rule), \Pi^P, z.s)$ 
14        end
15         $MergeOpenGatewayBranches(XOR.G^{Close})$ 
16      end
17      End Function
18   else if  $|rule.v| > 1$  then
19      $\Pi^P.AddInSequence(AND.G^{Open})$ 
20     foreach  $v \in rule$  do
21        $G^{Open}.AddBranch(branch)$ 
22        $call : AnnotateIO(v)$ 
23       if  $z.s \in v.O$  and the states of z are not in exclusive relation then
24          $ConstructACTModel(GetNext(rule), \Pi^P, v.O)$ 
25       end
26     end
27      $MergeOpenGatewayBranches(AND.G^{Close})$ 
28   end
29    $Mark(rule)$ 
30 End Function

```

Similar to Algorithm 5.1, Algorithm 5.2 also defines conditions (line 2 and 18) for adding these activities. As in line 2, when a business rule contains a single activity, that can be added directly to the ACT model in sequence with the existing node (line 4). The *AnnotateIO()* function defined from line 3-17, is for annotating the activity node with input and output artifacts and states after adding that node in the ACT model. This

function first retrieves the corresponding artifacts and states from the *pre-* (α) and *post-* (β) conditions of the business rules and then annotates them to the activity nodes. This function is also used to check if two states of an artifact belong to the post-condition of a business rule and for adding the corresponding activities into the ACT model, as described below.

As defined in line 8, when the two states of an artifact in the post-condition of an activity are in exclusive relation, an exclusive (XOR) open gateway node is added in sequence to this activity node in the ACT model. Then, for every artifact and state that is logically connected using the \vee operator, a new branch with that data (artifact and state) condition is added to the gateway as defined in lines 8-12. For example, the ACP model in Table 5.1 shows where the post-condition of *AnalyzeOrder* that belongs to rule R3 has a logical connection between the states *confirmed* and *rejected* of its output artifact *Order*. Therefore, a new XOR gateway is added in sequence to the *AnalyzeOrder* activity node in the ACT model and two branches with conditions [Order, confirmed] and [Order, rejected] are added to this gateway, which can be seen from Figure 5.2.

Next, as defined in line 13, to add activities to the gateway branches, every next business rule in the ACP model is traversed until all the activities associated with pre-conditions that match branch conditions are found and added to the corresponding branches of the gateway in the ACT model. The *MergeOpenGatewayBranches()* function is for merging those branches that contain no nodes, as their branch condition do not match the pre-condition of any rule in the ACP model. For example, after adding the XOR gateway with branch conditions as described above, the following rules are iterated for every logically connected artifact and state using the *ConstructACTModel()* function defined in Algorithm 5.1. In this case, the activities *ConfirmOrder* and *CloseOrder* of rules R4 and R11 are added to different branches of the XOR gateway, as they satisfy the corresponding branch conditions.

Similarly, Algorithm 5.2 uses an AND gateway, as defined from line 18-27, when a business rule has more than one activity. In this case, each activity of a business rule is added to a different branch of the AND gateway and is annotated with the associated artifacts and states. For example, rule R6 satisfies such a condition where it contains *PlanProduct* and *ScheduleProduct* activities that have to be executed in parallel. Therefore, these activities are added in parallel using an AND gateway in the ACT model, as shown in Figure 5.2. As defined in line 23 and 24, for each of these branch nodes the *ConstructACTModel()* function can be called, if the output of any of these branch nodes contain logically connected states using the \vee operator. This invocation is mainly to add the activities that must be added in sequence to these branch nodes.

As described above, the *ConstructACTModel()* function is recursively invoked for each business rule until all the business rules in the given ACP model are completed. In this manner, starting at the first rule (R1) of the ACP model presented in Table 5.1, every next rule is traversed until the last rule (R11) of that process model is obtained and the corresponding activities are added into the ACT model presented in Figure 5.2. An *end* node is added to the ACT model, after completing all the business rules.

5.4.2 Extract Model Traces

The procedure to build the execution traces of the constructed ACT model is given in Algorithm 5.3. The *BuildTrace()* function is recursively called to extract each execution trace of this process model. As defined in the algorithm, the function starts at the *start* node of the ACT model and traverses through every next sequence flow node until it reaches the *end* node of that process model and adds the activity nodes with the associated input and output artifacts that it finds on the way, to generate a process trace.

Algorithm 5.3: Extract the Activity-Centric Process Traces**Input** : The ACT Model (Π^P), its Log (Π_L^P) with empty trace τ^p **Output** : Complete Log (Π_L^P) of ACT Model (Π^P)**1 Function** BuildTrace ($n : \Pi^P, \tau^p : \Pi_L^P$) :**2** **if** $n \neq null$ **then****3** **if** $n \in A$ **then****4** $\tau^p.AddInSeq(n)$ **5** **else if** $n \in G^{Open}$ and n is not marked **then****6** **if** $n.type = AND$ **then****7** **foreach** $bn \in n$ **do****8** **if** bn is not marked **then****9** $BuildTrace(bn, \tau_p)$ **10** **end****11** **end****12** **else if** $n.type = XOR$ **then****13** **foreach** $bn \in n$ **do****14** **if** bn is not marked **then****15** $BuildTrace(bn, \tau_{new})$ **16** $\tau^p.AddInSeq(\tau_{new})$ **17** $(\Pi_L^P).Add(\tau^p)$ **18** **end****19** **end****20** **end****21** **end****22** $Mark(n)$ **23** $n \leftarrow GetSeqNext(n)$ **24** $BuildTrace(n, \tau^p)$ **25** **end****26 End Function**

As defined in line 3 and 5, the type of each node is checked first, such as *activity* or an open *gateway* node. In the case of activity, the node is added to the process trace. For example, the algorithm starts with an empty trace $\{\}$ from the *start* node of ACT model given in Figure 5.2, and traverses to the next node *InitiateOrder* (A), as this node is an activity, it is added to the empty trace $\{A\}$. Then the next nodes *ReceiveOrder* (B) and *AnalyzeOrder* (C) are also added $\{ABC\}$ in sequence to this trace.

For a gateway type, the algorithm defines two conditions, where in case of an *AND* gateway, as defined in lines 6-11, for every branch of this gateway the *BuildTrace()* function is called with the same process trace that is used to add all the branch nodes of that gateway, as they must always be executed in parallel. For example, to trace through the branch nodes *PlanProduct* and *ScheduleProduct*, the algorithm uses the existing trace instead of creating a new one.

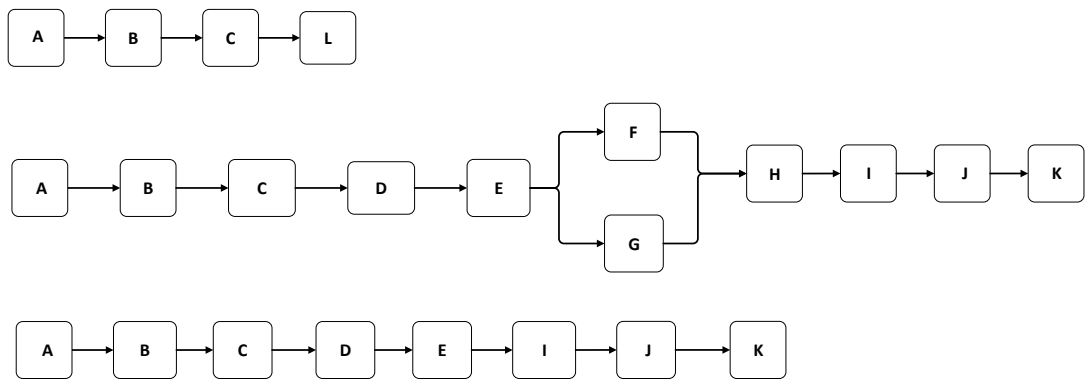


Figure 5.3: Execution Traces of Activity-Centric Process Model

For an XOR gateway node, as defined in lines 12-20, for every branch a new trace is used to traverse through the corresponding branch nodes. After traversing a branch of this gateway, the new trace is appended in sequence to the existing process trace, which is then added to the process log. For example, for the XOR gateway branch that contains a branch condition *Order[rejected]*, a new empty trace is created and the *CloseOrder (L)* node is added {L}. As there is no activity or an open gateway node following this node, the new trace {L} is added to the existing trace {ABCL} and is added to the process log. Then a new empty trace is created for the branch that contains *Order[confirmed]*, where the branch nodes *ConfirmOrder (D)* and *CheckStock (E)* are added to the new trace {DE} and is added in sequence to the existing trace {ABCDE}. For the next XOR gateway, a similar procedure is followed and the new traces {IJK} and {FGHIJK} are created and added in sequence to the existing trace {ABCDE}, which results in two other traces {ABCDEIJK} and {ABCDEFHIJK}.

As defined in line 22, every activity node is marked as visited, once that completes the traversal, while a gateway is marked after all its branches are traced. The extracted traces of the ACT Model are shown in Figure 5.3, where the activities are referenced with their labels for ease of understanding.

5.4.3 Trace-based Analysis

Algorithm 5.4 defines the *ConsistencyCheck()* function for the trace-based analysis of both process logs. This function takes as input the execution traces of the ACP and ACT process models, and a *count* variable that is initialized to zero and outputs the resulting consistency of the two models.

Algorithm 5.4: Analyzing the execution traces of ACP and ACT Models

Input : ACP Process Log (Π_L), ACT Process Log (Π_L^P)

Output : Consistency (\cong) of ACP and ACT Models

```

1 Function ConsistencyCheck ( $\tau : \Pi_L, \tau^p : \Pi_L^P, count$ ) :
2   foreach  $\tau \in \Pi_L$  do
3     foreach  $\tau^p \in \Pi_L^P$  do
4       if  $\tau^p \cong \tau$  then
5          $count \leftarrow count + 1$                                 ▷ initialized to 0
6       end
7     end
8   end
9   if  $count = |\Pi_L.\tau|$  then
10     $\Pi_L \cong \Pi_L^P$ 
11  end
12 End Function

```

According to the algorithm (lines 2-8), for a trace in the ACP process log, if there is a trace that contains the same set of activities in the same execution sequence in the ACT process log, then the value of count is incremented. After traversing all the process traces, if the count equals the number of traces in the ACP process log (lines 9-11), then it is clear that the constructed ACT model is consistent with the ACP model.

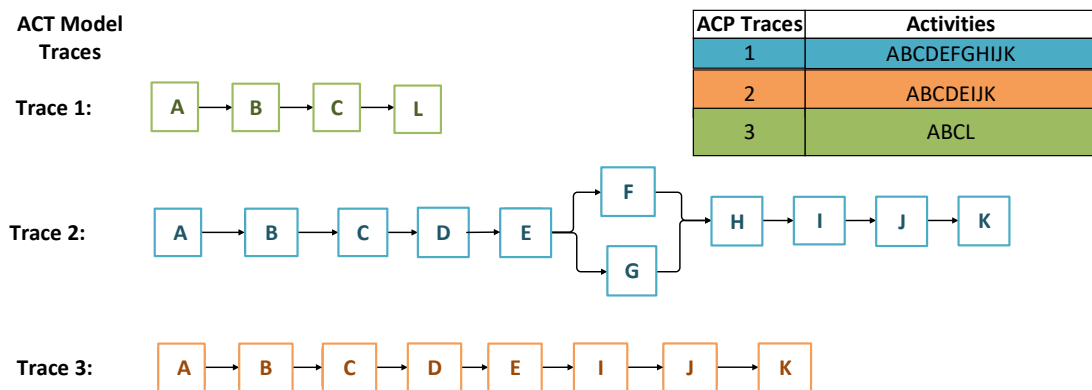


Figure 5.4: Process Traces and Analysis

Each of the recorded traces of the ACP model presented in Table 5.1 can be analyzed over the extracted execution traces of the ACT model, given in Figure 5.3 using Algorithm 5.4 to check their consistency. As shown in Figure 5.4, every trace of the ACP model has a matching trace in the ACT model, therefore it can be concluded that the constructed ACT model is consistent with the given ACP model. Here, the algorithm allows any execution order between the parallel activities.

5.5 Evaluation

In this section, a case study is demonstrated, as well as the prototype implementation and evaluation of the proposed approach.

5.5.1 Case Study

The ACP model presented in Figure 5.5 was utilized to demonstrate the feasibility of the proposed approach. The process model is taken from (Yongchareon et al., 2015), which illustrates the purchasing process in the supply chain domain. As shown in the figure, the purchasing process model is used to represent the inter-organizational process collaboration. The purchasing process is modeled as a set of lifecycles of

artifacts, including: Quote (Q), PurchaseOrder (PO), PickingList (PL), DeliveryNote (DN), ShippingOrder (SO), ShippingList (SL), Invoice (IV), Payment (P), and their synchronization dependencies. In the inter-organizational setting, these artifacts are classified into *local* and *shared* artifacts that represent the internal artifacts of an organization and those that are commonly agreed to be shared by every organization.

As illustrated in the figure, the purchasing process is initiated with the creation of the Quote and Purchase Order artifacts once the buyer places the order. When the Quote is approved, the Purchase Order is confirmed and sent to the supplier for acquiring the goods for that order. The supplier then creates the Picking List and checks for the ordered goods. The Quote is rejected and the Purchase Order is cancelled if the goods run out, otherwise the Purchase Order is filled, the Delivery Note is prepared, and the Shipping Order is created and sent to the logistics. Upon receiving the Shipping Order, the Shipping List is created and used to pickup the goods from the shipping point and also to deliver the goods to the buyer. Then the Invoice is created and sent to the buyer for payment. The Purchase Order is closed, once the Shipping Order is marked as arrived and the Invoice is cleared, thus completing the purchasing process.

Due to the unavailability of ACP models for empirical research, this process model is assumed in a single organizational setting. Therefore, the three roles, including Buyer, Supplier and Logistics are assumed to belong to the same organization and work together to fulfill the objectives of their organization. The ACP model of the purchasing process described using the business rules is presented in Table 5.3. As is shown, this model has 20 business rules and 21 activities that invoke the 8 listed artifacts. The execution traces of this model presented in Table 5.4 are recorded by analyzing each business rule as well as the involved activities and the pre- and post-conditions. The ACP model presented in Table 5.3 is further used to demonstrate the feasibility of the proposed algorithms. In order to construct the activity-centric process (ACT) model, Algorithm 5.1 starts at rule R1 of the ACP model, as its pre-condition matches with the

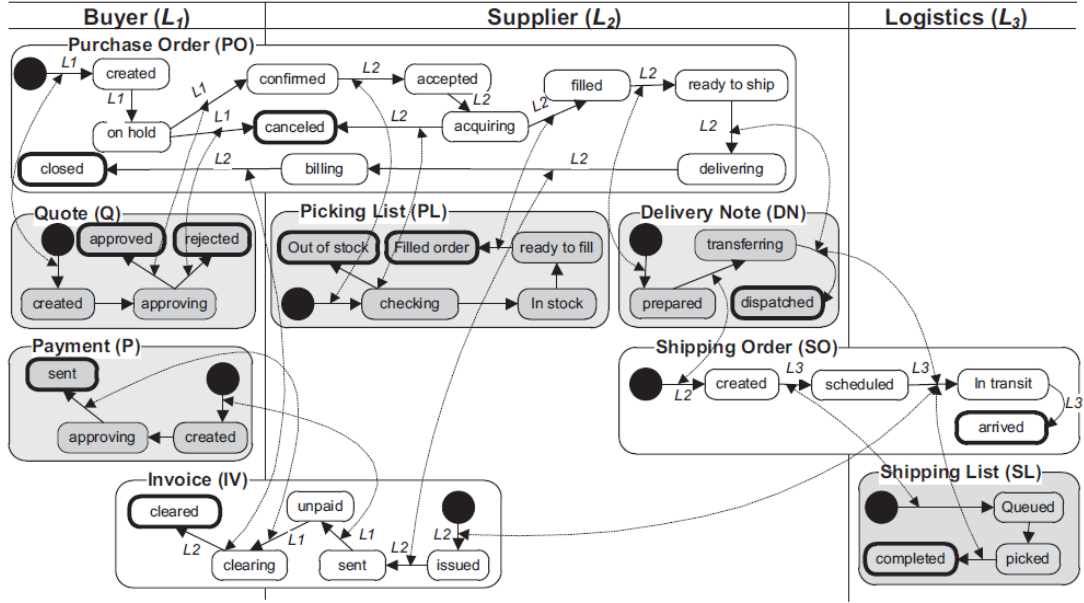


Figure 5.5: Purchasing process model (Yongchareon et al., 2015)

given data condition (*cond*) that contains the same pre-condition upon the start of this algorithm. Therefore, this algorithm invokes the *AddActivities()* function, as defined in Algorithm 5.2, by passing R1 and the ACT model (with start node). The *AddActivities()* function adds the *Create* activity of R1 in sequence with the *start* node of the ACT model with the associated input and output artifacts Quote (Q) and Purchase Order (PO) and their states *init* and *created*.

Algorithm 5.1 then updates the *cond* with the post-condition of R1 and calls the *ConstructACTModel()* function for the next business rule, R2. As the pre-condition of R2 is a subset of *cond*, the algorithm finds and invokes the *ConstructACTModel()* function for every rule whose pre-condition contains one or more artifacts and states of the *cond*. For example, there is no rule in the ACP model that contains both (Q, *created*) and (PO, *created*) in its pre-condition, therefore the algorithm traverses rules R3 and R2 that contain these artifact(s) and state(s) in their pre-conditions and adds the corresponding activities *SendQuote* and *Hold* in sequence with the *Create* node in the ACT model. Then the *cond* is updated with the post-condition of completed rule R2.

Table 5.3: ACP Model of Purchasing Process

Artifacts: Quote (Q), PurchaseOrder (PO), PickingList (PL), DeliveryNote (DN), ShippingOrder (SO), ShippingList (SL), Invoice (IV), Payment (P)

Activities: Create (T1), SendQuote (T2), Hold (T3), Analyse (T4), Accept (T5), Acquire (T6), Check (T7), Prepare (T8), FillOrder (T9), PrepareShipping (T10), CreateShipping (T11), ScheduleShipping (T12), Pick (T13), DispatchGoods (T14), IssueInvoice (T15), Transmit (T16), SendInvoice (T17), CreatePayment (T18), ApprovePayment (T19), ClearInvoice (T20), CompleteOrder (T21)

Business Rules: R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R20

R1: Buyer approves Quote Q to confirm PurchaseOrder PO for a selected supplier

Pre-condition: $\lambda(Q, \text{init}) \wedge \lambda(PO, \text{init})$

Activities: Create (PO, Q)

Post-condition: $\lambda(Q, \text{created}) \wedge \lambda(PO, \text{created})$

R2: Supplier accepts PurchaseOrder PO

Pre-condition: $\lambda(PO, \text{created})$

Activities: Hold (PO)

Post-condition: $\lambda(PO, \text{on hold})$

R3: Send Quote Q for approval

Pre-condition: $\lambda(Q, \text{created})$

Activities: SendQuote(Q)

Post-condition: $\lambda(Q, \text{approving})$

R4: Analyse the Quote Q and the PurchaseOrder PO

Pre-condition: $\lambda(Q, \text{approving}) \wedge \lambda(PO, \text{on hold})$

Activities: Analyse (Q, PO)

Post-condition: $\lambda((Q, \text{approved}) \wedge (PO, \text{confirmed}) \wedge (PL, \text{init})) \vee \lambda((Q, \text{rejected}) \wedge (PO, \text{cancelled}))$

R5: Accept the PurchaseOrder PO and Check the PickingList PL

Pre-condition: $\lambda(PO, \text{confirmed}) \wedge \lambda(PL, \text{init})$

Activities: Accept (PO, PL)

Post-condition: $\lambda(PO, \text{accepted}) \wedge \lambda(PL, \text{checking})$

R6: Acquire the PurchaseOrder PO

Pre-condition: $\lambda(PO, \text{accepted})$

Activities: Acquire(PO)

Post-condition: $\lambda(PO, \text{acquiring})$

R7: Check the PickingList PL for PurchaseOrder PO

Pre-condition: $\lambda(PO, \text{acquiring}) \wedge \lambda(PL, \text{checking})$

Activities: Check(PO, PL)

Post-condition: $\lambda((PL, \text{out of stock}) \wedge (PO, \text{cancelled})) \vee \lambda(PL, \text{in stock})$

R8: Prepare the PickingList PO for filling the order

Pre-condition: $\lambda(PL, \text{in stock})$

Activities: Prepare(PL)

Post-condition: $\lambda(PL, \text{ready to fill})$

R9: Fill the PickingList PL and PurchaseOrder PO

Pre-condition: $\lambda(PL, \text{ready to fill}) \wedge \lambda(PO, \text{acquiring})$

Activities: FillOrder (PL, PO)

Post-condition: $\lambda(PL, \text{Filled Order}) \wedge \lambda(PO, \text{filled}) \wedge \lambda(DN, \text{init})$

R10: Prepare the DeliveryNote DN for shipping the PurchaseOrder POPre-condition: λ (PO, filled) \wedge λ (DN, init)

Activities: PrepareShipping(PO, DN)

Post-condition: λ (PO, ready to ship) \wedge λ (DN, prepared) \wedge λ (SO, init)**R11: Supplier creates ShippingOrder SO from DeliveryNote DN**Pre-condition: λ (DN, prepared) \wedge λ (SO, init)

Activities: CreateShipping (DN, SO)

Post-condition: λ (DN, transferring) \wedge λ (SO, created) \wedge λ (SL, init)**R12: Schedule the ShippingOrder SO and place the ShippingList SL in queue**Pre-condition: λ (SO, created) \wedge λ (SL, init)

Activities: ScheduleShipping (SO, SL)

Post-condition: λ (SO, scheduled) \wedge λ (SL, queued) \wedge λ (IV, init)**R13: Pick up the ShippingList SL from the queue**Pre-condition: λ (SL, queued)

Activities: Pick(SL)

Post-condition: λ (SL, picked)**R14: Supplier dispatches goods for PurchaseOrder PO that to be shipped by ShippingOrder SO, and simultaneously issues Invoice iv to the Buyer**Pre-condition: λ (PO, ready to ship) \wedge λ (DN, transferring) \wedge λ (SO, scheduled) \wedge λ (SL, picked) \wedge λ (IV, init)

Activities: DispatchGoods(PO, DN, SO, SL) IssueInvoice(po, iv)

Post-condition: λ ((PO, delivering) \wedge (DN, dispatched) \wedge (SO, In transit) \wedge (SL, completed)) \wedge λ (IV, issued)**R15: Send ShippingOrder SO to Logistics**Pre-condition: λ (SO, in transit)

Activities: Transmit (SO)

Post-condition: λ (SO, arrived)**R16: Send invoice for PurchaseOrder PO**Pre-condition: λ (PO, delivering) \wedge λ (IV, issued)

Activities: SendInvoice(PO, IV)

Post-condition: λ (PO, billing) \wedge λ (IV, sent) \wedge λ (P, init)**R17: Create Payment P for the Invoice IV**Pre-condition: λ (IV, sent) \wedge λ (P, init)

Activities: CreatePayment (IV, P)

Post-condition: λ (IV, unpaid) \wedge λ (P, created)**R18: Approve the Payment P**Pre-condition: λ (P, created)

Activities: ApprovePayment (P)

Post-condition: λ (P, approving)**R19: Send Payment P and clear the Invoice IV**Pre-condition: λ (P, approving) \wedge λ (IV, unpaid)

Activities: ClearInvoice (P, IV)

Post-condition: λ (P, sent) \wedge λ (IV, clearing)**R20: Clear the Invoice I and close the PurchaseOrder PO**Pre-condition: λ (IV, clearing) \wedge λ (PO, billing)

Activities: CompleteOrder (PO, IV)

Post-condition: λ (IV, cleared) \wedge λ (PO, closed)

Table 5.4: Execution Log of Purchasing Process Model

Traces	Business Rules	Activities
1	R1R3R2R4R5R6R7	T1T2T3T4T6T7
2	R1R3R2R4R5R8R9R10 R11R12R13R14R15R16R17 R18R19R20	T1T2T3T4T6T7T8T9T10 T11T12T13T14T15T16T17 T18T19T20T21
3	R1R3R2R5	T1T2T3T4

As defined in Algorithm 5.2, every rule (R1, R2, R3) is marked as completed after adding all the activities of that rule into the ACT model. Then, the next the rule R4 is obtained by Algorithm 5.1, as this rule satisfies the third condition that is *cond* is a subset of the pre-condition of R4. The algorithm checks if the artifact(s) and state(s) of this pre-condition already exist in the ACT model as input or output of the activity nodes. If they exist, then the activity *Analyse* of rule R4 is added in sequence to the *Hold* activity node of ACT model. In this manner, Algorithm 5.1 finds the rules that contain these post artifacts and states in their pre-conditions and retrieves the corresponding activities and adds them into the ACT model. The ACT model of purchasing process model constructed using Algorithm 5.1 is presented in Figure 5.6. Due to the space limitation, the artifacts and states are not shown in this figure, however the complete process model can be found from the Appendix, in Figure A.6.

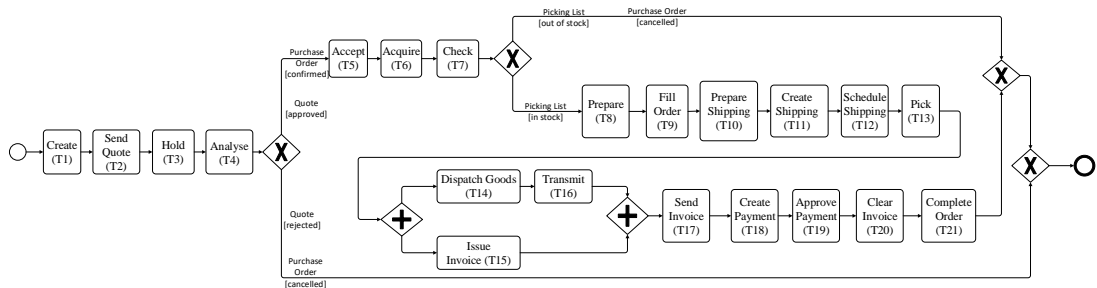


Figure 5.6: An Abstracted Activity-Centric Model of Purchasing Process

Algorithm 5.3 is used to extract the execution traces of the ACT model in order to

check their consistency with the recorded traces of the ACP model. As described in the previous section, this algorithm starts with an empty trace from the *start* node of ACT model given in Figure 5.6 and traverses every sequence flow node and adds the activity nodes that found on the path to generate a process trace. As explained above, for every branch of the XOR gateway, a new trace is created and is used to add all the activity nodes flowing on that branch. These new traces are appended to the existing traces to create a complete process trace. As shown in Figure 5.6, there are three alternative paths for the process model, therefore Algorithm 5.3 generates three traces for this process model.

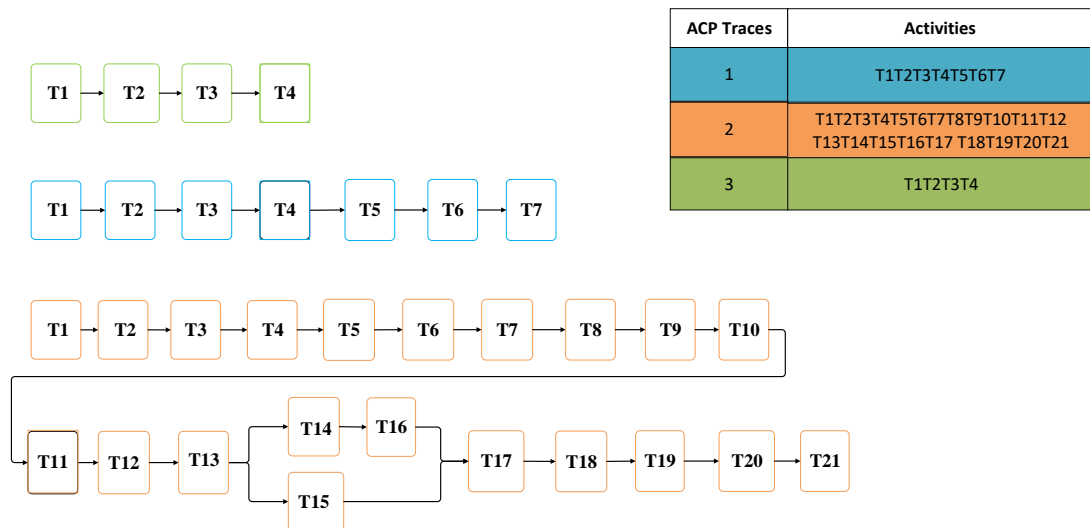


Figure 5.7: The constructed Activity-Centric Model of Purchasing Process

The extracted traces of the purchasing process model can be observed in Figure 5.7, where the three execution traces represent the three possible paths that the process model takes to complete its execution. According to trace 1, after creating the *Quote* and *PurchaseOrder*, the *Quote* is sent for approval, where it is analyzed and either *approved* or *rejected* based on that the *PurchaseOrder* is either *confirmed* or *cancelled*. The process is terminated if the *Quote* and *PurchaseOrder* artifacts are put in the states *rejected* and *cancelled*. According to trace 2, when the *Quote* and *PurchaseOrder*

artifacts are put in the states *approved* and *confirmed*, and the *PickingList* artifact is created (*init* state). In this case, the *PurchaseOrder* is *accepted* and the *PickingList* is checked for acquiring the ordered goods. The *PurchaseOrder* is *cancelled*, if the *PickingList* is *out of stock* and the process is closed. According to trace 3, when the *PickingList* is *in stock* then the *PurchaseOrder* is *filled*, and the *ShippingOrder*, *ShippingList*, *DeliveryNote* and *Invoice* are prepared and sent for shipping the ordered goods. The *Payment* is then *created* and is *approved* to clear the invoice and complete the purchasing process.

Each of the recorded traces of the purchasing process (ACP) model can be analyzed over the extracted execution traces of the constructed ACT model using Algorithm 5.4 to check their consistency. As shown in Figure 5.7, every trace of the ACP model has a matching trace in the ACT model, therefore it is clear that the constructed ACT model is consistent with the ACP model of the given purchasing process.

5.5.2 Implementation

A prototype tool was developed based on the proposed algorithms. The tool takes an XML specification of the ACP model as input. The ACP model specification is based on the extension of RuleML (Harold Boley, 2014), a standard for defining business rules. The structure of the ACP model in an XML format is shown in Figure 5.8. The prototype tool parses this specification and constructs the XML specification of the ACT model by following the construction procedure. Then, the tool extracts the execution traces of the ACT model, which are matched with the execution traces of the ACP model. For each trace of the ACP model, the tool iterates over the extracted traces of the ACT model until it finds a matching trace. When a match occurs, the corresponding trace in the activity-centric process log is marked as completed, so that the trace will not be considered in the following iteration. Here, trace matching not only includes the

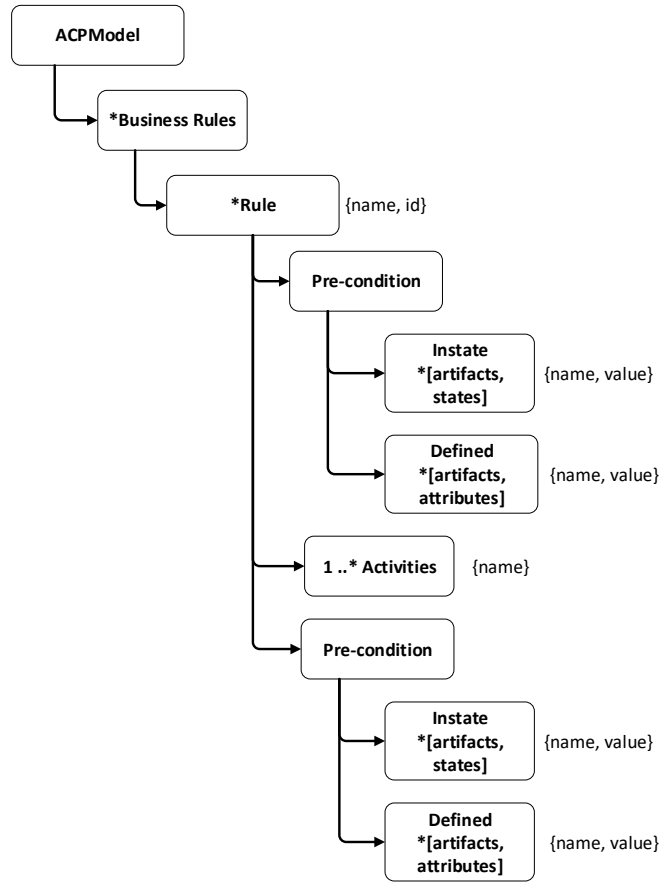


Figure 5.8: XML Format of ACP Model

matching of activities and their input and output artifacts, but also the artifact states and attributes. The prototype also presents information on the unmatched traces, for example when a trace of the ACP model is not found in the extracted traces, a message showing the missing trace in the constructed ACT model is displayed.

5.5.3 Experimental Discussion and Analysis

To evaluate the applicability and feasibility of the proposed approach, experiments have been conducted on the developed prototype by utilizing the above two motivating process models. For these test inputs, the prototype generated correct ACT models, with all traces matching the traces of the input ACP models. Based on the experiments, it is expected that the construction algorithms produce all the traces in the ACT model,

if every rule of the ACP model falls in one of the defined conditions. Therefore, when a trace of the ACP model is not found in the constructed ACT model, rewriting the original ACP model according to the specified requirements or to manually adding/updating the traces of ACT model can be useful to reconstruct the missing traces of the original process model in the resulting process model.

There are many ways to check the consistency between process models. Some of the well-known notions include the strong bi-simulation, weak bi-simulation and trace equivalence (Baier & Katoen, 2008), which can be used to compare the equivalence between process models. The proposed consistency checking method is based on the notion of trace equivalence that defines a way to draw the similarity between two models by checking if they are trace equivalent. According to this notion, two process models exhibit the trace equivalence if they can successfully perform the same set of actions in the same sequence.

Trace equivalence has two problems, in that the set of traces may be infinite and that it does not capture the moment of choice when used directly on the process models that have no similar execution semantics to check their consistency. These problems can be resolved (van der Aalst, De Medeiros & Weijters, 2006) when the behaviour of the two process models is observed from their execution logs and those behavioural traces are used to compare the two process models. Therefore, by following this notion, the proposed trace-based method aims to check the consistency between the two process models from the recorded execution traces of process models rather than directly using the process models.

5.6 Discussion and Related Work

The activity-centric and artifact-centric approaches provide different views of a business process. Activity-centric process models are used to capture the allowed execution

sequence of activities, where a flow that is not specified in such model is not allowed (De Giacomo et al., 2015). On the other hand, artifact-centric process models give freedom to execute activities as long as they meet the specified business conditions. However, these process models are less comprehensible due to their unstructured sets of business rules that often impede their adoption (Haisjackl & Zugal, 2014; Caron & Vanthienen, 2016). Therefore, this chapter proposes an approach to transform artifact-centric process models into activity-centric process models (Kunchala et al., 2020).

As discussed in Chapter 2, there are some approaches to the proposed transformation. However, most of those approaches use synchronized/unsynchronized artifact lifecycles to generate the activity-centric process models (Küster et al., 2007; Redding et al., 2008; Meyer & Weske, 2013) rather than the artifact-centric process models that contain business rules. Although there exist a few approaches to transform declarative models to imperative (activity-centric) models, they pose some limitations. For example, the approach proposed in (Prescher et al., 2014), produces duplicate tasks that lead to an increased number of execution alternatives for the resulting process models. Similarly, the approach proposed in (De Giacomo et al., 2015), does not consider the parallel states of objects.

In comparison with the above existing works, the approach presented in this chapter enables a direct transformation. Therefore, the proposed approach is useful to transform artifact-centric process models that contain business rules into activity-centric process models and also to address the aforementioned limitations.

5.7 Summary

In this chapter, the third research question (RQ3) is addressed by presenting an approach that aims to transform artifact-centric process models into activity-centric process models. In this regard, the proposed approach presents algorithms to construct an

activity-centric process model from the artifact-centric process model, and to extract the execution traces of the constructed process model in order to check the consistency of the constructed model with the base model. A case study using a purchasing process model from the supply chain domain is used to demonstrate the feasibility of the proposed approach. The prototype implementation and evaluation of the proposed algorithms is also discussed.

Chapter 6

Conclusion and Future Directions

This chapter concludes the thesis by summarizing the contributions of the research and outlining some issues for future research. Section 6.1 provides a discussion on the contributions of this thesis. Section 6.2 discusses the limitations and possible improvements of proposed research. Section 6.3 presents an overview of the future research.

6.1 Thesis Contributions

The research emphasized three challenges in supporting the transformation between the activity-centric and the artifact-centric modeling paradigms. These challenges include: i) the transformation of traditional activity-centric process models into artifact-centric process models; ii) the merging of the collaborating processes of an Inter-Organizational Business Process (IOBP) model; and iii) the construction of activity-centric process models from artifact-centric process models. In this thesis, novel approaches to address each of the aforementioned research questions are proposed. The proposed solutions are also evaluated to reveal the degree to which they can resolve the proposed problems. The main contributions of this thesis are summarized below.

6.1.1 The Synthesis Approach

Modeling business processes using artifacts and their interacting lifecycles is the core objective of the artifact-centric paradigm. In recent years, the transformation of traditional activity-centric process models into artifact-centric process models has become one of the major research interests of the business and research community. This has led to the emergence of several transformation approaches that support the semi-automatic synthesis of unsynchronized/synchronized artifact lifecycles. While a fully automated approach can facilitate the defined transformation, an automated approach is proposed and evaluated to demonstrate its feasibility and applicability.

Chapter 3 presented and discussed the synthesis approach that aims to generate synchronized artifact lifecycles from the standalone activity-centric process models, which are annotated with artifacts and states. The proposed approach is comprised of three phases: building a hierarchical tree representation of the activity-centric process model, using the process tree to generate the lifecycles of artifacts, then to refine and synchronize the generated artifact lifecycles. The thesis presented algorithms for each phase of the proposed approach. A moderately complex business case was used to demonstrate the feasibility of the proposed approach and the proposed algorithms were implemented and evaluated using a process model collection to validate the applicability and efficiency of proposed approach in the real world.

6.1.2 The Merge Approach

Artifact-centric counterparts, such as synchronized artifact lifecycles are particularly useful in the inter-organizational setting to ensure the correct execution of underlying business processes that span multiple organizations to achieve a common business goal by sharing their data resources. However, generating such artifact-centric counterparts from activity centric IOBP models is challenging compared to standalone process

models, as the behaviour of artifacts may be affected while capturing their states into the lifecycles. Therefore, this thesis proposed an approach to support the synthesis of artifact lifecycles from IOBPs and to demonstrate the feasibility and applicability of the proposed approach. The proposed approach is based on the notion of process merging that aims to combine two or more process models to generate an integrated process model by preserving the structure and behaviour of the base processes.

Chapter 4 presented and discussed an automated approach to merge the interacting processes of an activity-centric IOBP model that contains a flow of artifacts and states. The proposed approach is based on the synchronous and asynchronous interactions between the collaborating processes. Specifically, rules were proposed that define ways to combine the nodes of two or more collaborating processes. Algorithms were also presented based on these rules to support automatic process merging. The algorithms were implemented and evaluated to show their feasibility and applicability using a subset of process models from different business domains. This approach complimented the synthesis approach by extending support for the inter-organizational business process models.

6.1.3 The Construction Approach

The artifact-centric approach is useful to achieve higher flexibility due to its declarative nature. However, the drawback of declarative modelling is that it is difficult to understand the structure and flow of these process models compared to activity-centric process models, which also allow a natural mapping onto the executable models to achieve higher efficiency. Therefore, this thesis proposed an approach that uses artifact-centric process models to construct activity-centric process models.

Chapter 5 presented and discussed an automated approach to construct activity-centric process models from artifact-centric process models. The proposed approach

consists of algorithms to support the construction of activity-centric process models and to determine their consistency with the base models. A trace-based method is a part of these algorithms that extracts the execution traces of constructed models and analyzes them over the execution traces of the base model to check the consistency between both the models. The feasibility of the proposed approach was demonstrated using a process model from the supply chain domain. The proposed algorithms were implemented and evaluated using two motivating business scenarios.

6.2 Limitations and Possible Improvements

Although the above described solutions can efficiently solve the three research questions proposed in this thesis, there are some aspects to be addressed in the future. This section mainly discusses some of the possible improvements for future studies on the proposed solutions, as there are some limitations concerning their scope and applicability in the real business environment.

Firstly, the synthesis approach provides a generalized solution that can be applicable to other types of process models, such as UML activity diagrams, requiring that these models are represented with similar constructs. However, for the proposed synthesis, all the involved artifacts and states must be made explicit in the business process model, which is not always possible in a real scenario. For example, in BPMN, the data stores are used as repositories to store and retrieve data objects. Therefore, in future the proposed synthesis approach can be extended to deal with process models that have other modeling elements specific to the underlying modeling notation. It is worth mentioning that, extending the proposed synthesis approach to the process models that contain nested loops or subroutines is also one of the future research direction.

Secondly, the proposed merge is useful in two directions, one for the synthesis of artifact lifecycles from IOBPs and for merging the interacting processes of one or

more organizations during their business restructuring. However, this approach has a limitation, in that only the one-to-one interaction is allowed between the process nodes. In addition, the dataset used to evaluate the merge approach may be acceptable to demonstrate the feasibility and applicability of this approach. However, this may not be sufficient to draw conclusions on the efficiency of the proposed solution. Therefore, in future, this approach can be extended to consider more complex interactions between the processes, and it can be tested with a large dataset for evaluating the efficiency of this approach in a real scenario.

Lastly, due to the unavailability of real ACP models, a process model from the existing literature was utilized to evaluate the proposed construction approach. Although this process model is appropriate concerning the required aspects and makes a good fit for evaluating the proposed approach, it may not be sufficient to evaluate the efficiency of the proposed solution. Therefore, further tests can be proposed in future to reveal the efficiency of this solution. In addition, the proposed approach can be extended to artifact-centric IOBP models that contain a set of roles representing different participants of the collaborating business entities.

As discussed above, scope of the transformation approaches presented in this thesis is limited to structured (and semi-structured) process models that contain the most commonly used modeling constructs. Thus, these approaches can be further extended to transform process models that include other modeling constructs such as event-based gateways and loops.

6.3 Future Research Directions

This section elaborates on some of the challenges and opportunities for future research in BPM.

6.3.1 Internet-of-Things (IoT)

In the past few years, the Internet-of-Things (IoT) has emerged as the new technological evolution by equipping physical objects with sensing hardware and software, which turn them into smart objects (Meroni et al., 2018). The objective of the IoT is to facilitate the development of new applications and the improvement of existing applications (Dijkman, Sprenkels, Peeters & Janssen, 2015). Recently, numerous IoT applications have been developed and extended into major areas, including: *intelligent transportation* (Sheng-nan, Pei-pe, Jian-li & Xiao-he, 2015; Sathiyaraj & Balamurugan, 2018), *healthcare* (Sivagami, Revathy & Nithyabharathi, 2016; Manoj, Hussain & Teja, 2019), *inventory* (Caro & Sadr, 2019), *security and surveillance* (Abu, Nordin, Suboh, Yid & Ramli, 2018; Panchatcharam & Vivekanandan, 2019), *smart environments* (Hamdan, Shanableh, Zaki, Al-Ali & Shanableh, 2019) and for *waste management* (Badadhe Komal & Dahiwal Shital, 2019) to list a few. Many businesses have already implemented IoT technology, mainly in their manufacturing, retailing and transportation processes to gain some of its benefits in terms of cost reduction and improved productivity (Pundir et al., 2019). The major IoT-based applications are used in domains such as transportation and logistics to support businesses with remote monitoring and controlling of their goods throughout the supply chain process (Da Xu, He & Li, 2014; Y. Lu, Papagiannidis & Alamanos, 2018).

Janiesch et al. (2017) highlighted the mutual benefits and several research challenges in bridging the gap between the IoT and BPM. Some of these challenges are to manage the linking between the IoT data and business processes; to detect new processes from IoT data and to bridge the gap between event-based and process-based systems; and to improve conformance checking and the monitoring and execution of business processes. To resolve some of these challenges, the authors state that data-centric paradigms offer promising perspectives in linking the IoT data and processes, provide more flexibility,

and support process execution and monitoring.

Recently, Meroni et al. (2017, 2018) proposed to use the artifact-centric approach for monitoring the compliant execution of IOBPs. Based on the artifacts, a decentralized solution that enables the physical objects participating in the collaboration to monitor their own business conditions and activities that invoke them. However, the privacy and security aspects of organizations participating in the collaboration still need to be resolved in future.

6.3.2 Block Chain

Block Chain is an emerging technology that is expanding its applications to diverse fields, including: *finance* (Nofer, Gomber, Hinz & Schiereck, 2017), *food safety* (Tian, 2017), *trade logistics* (Hackius & Petersen, 2017), *healthcare* (Mettler, 2016), *education* (Grech & Camilleri, 2017; Turkanović, Hölbl, Košič, Heričko & Kamišalić, 2018) and *agriculture* (Staples et al., 2017) to support a broad range of business domains and individuals. Shared ledger is a commonly known block chain technology that enables collaboration among diverse stakeholders situated across multiple organizations. In this regard, smart contracts, i.e., programs that are event-driven and data-centric, are used to guide such collaboration.

Recently, Hull et.al (2016; 2017) investigated the suitability of the artifact-centric paradigm and the notion of artifact-centric services interoperation (ACSI) hubs (Hull et al., 2009) for a shared ledger business process collaboration. The authors demonstrated the potential advantages of the artifact-centric paradigm in providing a robust basis for shared ledger business collaboration and highlighted several challenges to adopting this paradigm in block chain technology. These challenges include: Model abstractions to design a coherent business collaboration language that supports a more procedural

and declarative style to cover the use cases from different industry sectors; View-based intuitive meta-models to specify access rights to ensure the privacy of data and processing steps in a business collaboration and interfaces for supporting the interaction between smart contracts; Transformation approaches that support the conversion from existing legacy collaborating business processes into block chain-enabled processes; Extending existing automatic artifact-centric verification techniques to the block chain context; Realization methods to implement block chain based business collaboration, and the tools that support the discovery of smart contracts based on artifacts for checking the compatibility and for reasoning and testing their behaviours resulting from their interactions.

Mendling et al. (2018) also proposed several new challenges and opportunities for future research in block chain technology for BPM. These challenges include: The development of new execution and monitoring systems and to devise new analysis and engineering methods for business processes based on block chain technology; To redesign the business processes to leverage the opportunities of block chains and to define appropriate methods for their evolution and adaptation; Developing techniques for the identification, discovery and analysis of relevant business processes for the adoption of block chain technology; Understanding the impact of block chains in the business innovation; Investigating culture changes towards the management and execution of business processes with the introduction of block chains and how far this technology has come in comparison to the adoption of other technologies.

In summary, this thesis aimed to facilitate the transformation between artifact-centric and activity-centric process models. In this regard, automatic approaches were proposed to address the aforementioned research questions. The feasibility and applicability of these approaches were also demonstrated using a set of process models from the BPMAI process model repository and existing literature. This thesis also established the sufficient conditions under which the proposed transformation is possible. Although

a comprehensive evaluation is presented in this thesis, in the future an evaluation using a large set of real process models can be useful to reveal the efficiency of proposed approaches in practice.

References

- Abiteboul, S., Benjelloun, O. & Milo, T. (2008). The active xml project: an overview. *The VLDB Journal—The International Journal on Very Large Data Bases*, 17(5), 1019–1040.
- Abiteboul, S., Bourhis, P., Galland, A. & Marinoiu, B. (2009). The axml artifact model. In *2009 16th international symposium on temporal representation and reasoning* (pp. 11–17).
- Abiteboul, S., Bourhis, P., Marinoiu, B. & Galland, A. (2010). Axart: enabling collaborative work with axml artifacts. *Proceedings of the VLDB Endowment*, 3(1-2), 1553–1556.
- Abiteboul, S., Segoufin, L. & Vianu, V. (2009a). Modeling and verifying active xml artifacts. *IEEE Data Eng. Bull.*
- Abiteboul, S., Segoufin, L. & Vianu, V. (2009b). Static analysis of active xml systems. *ACM Transactions on Database Systems (TODS)*, 34(4), 23.
- Abu, M. A., Nordin, S. F., Suboh, M. Z., Yid, M. S. M. & Ramli, A. F. (2018). Design and development of home security systems based on internet of things via favoriot platform. *International Journal of Applied Engineering Research*, 13(2), 1253–1260.
- Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., . . . others (2003). *Business process execution language for web services*. version.
- Arkin, A., Askary, S., Bloch, B., Curbera, F., Golland, Y., Kartha, N., . . . Yiu, A. (2005). Web services business process execution language version 2.0. *Working Draft. WS-BPEL TC OASIS*.
- Assy, N., Chan, N. N. & Gaaloul, W. (2013). Assisting business process design with configurable process fragments. In *2013 ieee international conference on services computing* (pp. 535–542).
- Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Naur, P., . . . others (1960). Report on the algorithmic language algol 60. *Numerische Mathematik.*, 2(1), 106–136.
- Badadhe Komal, R. & Dahiwal Shital, S. (2019). Iot based intelligent system for wastemanagement. *Waste management*.
- Bagheri Hariri, B., Calvanese, D., De Giacomo, G., Deutsch, A. & Montali, M. (2013). Verification of relational data-centric dynamic systems with external services. In *Proceedings of the 32nd acm sigmod-sigact-sigai symposium on principles of database systems* (pp. 163–174).

- Baier, C. & Katoen, J.-P. (2008). *Principles of model checking*. MIT press.
- Barcelona, M., García-Borgoñón, L., Escalona, M. & Ramos, I. (2018). Cbg-framework: A bottom-up model-based approach for collaborative business process management. *Computers in Industry*, 102, 1–13.
- Barker, A., Walton, C. D. & Robertson, D. (2009). Choreographing web services. *IEEE Transactions on Services Computing*, 2(2), 152–166.
- Belardinelli, F., Lomuscio, A. & Patrizi, F. (2011). Verification of deployed artifact systems via data abstraction. In *International conference on service-oriented computing* (pp. 142–156).
- Belardinelli, F., Lomuscio, A. & Patrizi, F. (2012a). An abstraction technique for the verification of artifact-centric systems. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.
- Belardinelli, F., Lomuscio, A. & Patrizi, F. (2012b). Verification of gsm-based artifact-centric systems through finite abstraction. In *International conference on service-oriented computing* (pp. 17–31).
- Bhattacharya, K., Caswell, N. S., Kumaran, S., Nigam, A. & Wu, F. Y. (2007). Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems Journal*, 46(4), 703–721.
- Bhattacharya, K., Gereade, C., Hull, R., Liu, R. & Su, J. (2007). Towards formal analysis of artifact-centric business process models. In *International conference on business process management* (pp. 288–304).
- Bhattacharya, K., Hull, R. & Su, J. (2009). A data-centric design methodology for business processes. In *Handbook of research on business process modeling* (pp. 503–531). IGI Global.
- Boaz, D., Heath, T., Gupta, M., Limonad, L., Sun, Y., Hull, R. & Vaculin, R. (2014). The acsi hub: A data-centric environment for service interoperation. In *Bpm (demos)* (p. 11).
- Bouchbout, K. & Alimazighi, Z. (2011). Inter-organizational business processes modelling framework. In *Adbis* (2) (pp. 45–54).
- Bulanov, P., Lazovik, A. & Aiello, M. (2011). Business process customization using process merging techniques. In *Service-oriented computing and applications (soca), 2011 ieee international conference on* (pp. 1–4).
- Cabanillas, C., Resinas, M., Ruiz-Cortés, A. & Awad, A. (2011). Automatic generation of a data-centered view of business processes. In (pp. 352–366). London, UK: Springer.
- Cangialosi, P., De Giacomo, G., De Masellis, R. & Rosati, R. (2010). Conjunctive artifact-centric services. In *International conference on service-oriented computing* (pp. 318–333).
- Cardoso, J. (2005). Control-flow complexity measurement of processes and weyuker's properties. In *6th international enformatika conference* (Vol. 8, pp. 213–218).
- Caro, F. & Sadr, R. (2019). The internet of things (iot) in retail: Bridging supply and demand. *Business Horizons*, 62(1), 47–54.
- Caron, F. & Vanthienen, J. (2016). Exploring business process modelling paradigms and design-time to run-time transitions. *Enterprise Information Systems*, 10(7),

- 790–813.
- Chao, T., Cohn, D., Flatgard, A., Hahn, S., Linehan, M., Nandi, P., . . . Wu, F. (2009). Artifact-based transformation of ibm global financing. In (pp. 261–277). Ulm, Germany: Springer.
- Chebbi, I., Dustdar, S. & Tata, S. (2006). The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering*, 56(2), 139–173.
- Chinnici, R., Haas, H., Lewis, A. A., Moreau, J.-J., Orchard, D. & Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 2: Adjuncts. *W3C Recommendation*, 6.
- Chinosi, M. & Trombetta, A. (2012). Bpmn: An introduction to the standard. *Computer Standards & Interfaces.*, 34(1), 124–134.
- Chiu, D. K., Cheung, S.-C., Till, S., Karlapalem, K., Li, Q. & Kafeza, E. (2004). Workflow view driven cross-organizational interoperability in a web service environment. *Information Technology and Management*, 5(3-4), 221–250.
- Chiu, D. K., Karlapalem, K., Li, Q. & Kafeza, E. (2002). Workflow view based e-contracts in a cross-organizational e-services environment. *Distributed and parallel databases*, 12(2-3), 193–216.
- Clarke, E. M. (1999). Model checking.
- Clarke, E. M. & Emerson, E. A. (1981). Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on logic of programs* (pp. 52–71).
- Cohn, D., Dhoolia, P., Heath, F., Pinel, F. & Vergo, J. (2008). Siena: From powerpoint to web app in 5 minutes. In *International conference on service-oriented computing* (pp. 722–723).
- Cohn, D. & Hull, R. (2009). Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32(3), 3–9.
- Damaggio, E., Deutsch, A., Hull, R. & Vianu, V. (2011). Automatic verification of data-centric business processes. In *International conference on business process management* (pp. 3–16).
- Damaggio, E., Hull, R. & VaculíN, R. (2013). On the equivalence of incremental and fixpoint semantics for business artifacts with guard–stage–milestone lifecycles. *Information Systems*, 38(4), 561–584.
- Davidson, J. D. & Mordani, R. (2000). Java api for xml processing.
- Da Xu, L., He, W. & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4), 2233–2243.
- Decker, G. & Barros, A. (2007). Interaction modeling using bpmn. In *International conference on business process management* (pp. 208–219).
- Decker, G., Kopp, O., Leymann, F. & Weske, M. (2007). Bpel4chor: Extending bpm for modeling choreographies. In *Ieee international conference on web services (icws 2007)* (pp. 296–303).
- Decker, G. & Weske, M. (2007). Local enforceability in interaction petri nets. In *International conference on business process management* (pp. 305–319).

- Decker, G. & Weske, M. (2011). Interaction-centric modeling of process choreographies. *Information Systems*, 36(2), 292–312.
- De Giacomo, G., De Masellis, R. & Rosati, R. (2012). Verification of conjunctive artifact-centric services. *International Journal of Cooperative Information Systems*, 21(02), 111–139.
- De Giacomo, G., Dumas, M., Maggi, F. M. & Montali, M. (2015). Declarative process modeling in bpmn. In *International conference on advanced information systems engineering* (pp. 84–100).
- de Leoni, M., Maggi, F. M. & van der Aalst, W. M. (2015). An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data. *Information Systems*, 47, 258–277.
- Derguech, W. & Bhiri, S. (2011). An automation support for creating configurable process models. In *International conference on web information systems engineering* (pp. 199–212).
- Derguech, W., Bhiri, S. & Curry, E. (2017). Designing business capability-aware configurable process models. *Information Systems*, 72, 77–94.
- Deutsch, A., Hull, R., Li, Y. & Vianu, V. (2018). Automatic verification of database-centric systems. *ACM SIGLOG News*, 5(2), 37–56.
- Deutsch, A., Hull, R., Patrizi, F. & Vianu, V. (2009). Automatic verification of data-centric business processes. In *Proceedings of the 12th international conference on database theory* (pp. 252–267).
- Deutsch, A., Hull, R. & Vianu, V. (2014). Automatic verification of database-centric systems. *ACM SIGMOD Record*, 43(3), 5–17.
- Deutsch, A., Li, Y. & Vianu, V. (2016). Verification of hierarchical artifact systems. In *Proceedings of the 35th acm sigmod-sigact-sigai symposium on principles of database systems* (pp. 179–194).
- Dijkman, R. M., Dumas, M. & Ouyang, C. (2008). Semantics and analysis of business process models in bpmn. *Information and Software technology*, 50(12), 1281–1294.
- Dijkman, R. M., Sprenkels, B., Peeters, T. & Janssen, A. (2015). Business models for the internet of things. *International Journal of Information Management*, 35(6), 672–678.
- Dumas, M. (2011). On the convergence of data and process engineering. In *East european conference on advances in databases and information systems* (pp. 19–26).
- Dumas, M. & Ter Hofstede, A. H. (2001). Uml activity diagrams as a workflow specification language. In *International conference on the unified modeling language* (pp. 76–90).
- Emerson, E. A. (1997). Model checking and the mu-calculus. *DIMACS series in discrete mathematics*, 31, 185–214.
- Eshuis, R. (2006). Symbolic model checking of uml activity diagrams. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1), 1–38.
- Eshuis, R. & Grefen, P. (2008). Constructing customized process views. *Data & Knowledge Engineering*, 64(2), 419–438.

- Eshuis, R. & Van Gorp, P. (2012). Synthesizing object life cycles from business process models. In *International conference on conceptual modeling* (pp. 307–320).
- Eshuis, R. & Van Gorp, P. (2016). Synthesizing object life cycles from business process models. *Software & Systems Modeling*, 15(1), 281–302.
- Eshuis, R., Vonk, J. & Grefen, P. (2011). Transactional process views. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 119–136).
- Fagin, R., Kolaitis, P. G., Miller, R. J. & Popa, L. (2005). Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1), 89–124.
- Fan, B., Li, Y., Liu, S. & Zhang, Y. (2015). Run jta in jtang: Modeling in artifact-centric model and running in activity-centric environment. In *Asia-pacific conference on business process management* (pp. 83–97).
- Ferguson, D. (2001). Ibm web services: Technical and product architecture and roadmap. *IBM Whitepaper*; May.
- Friedow, C., Völker, M. & Hewelt, M. (2018). Integrating iot devices into business processes. In *International conference on advanced information systems engineering* (pp. 265–277).
- Fritz, C., Hull, R. & Su, J. (2009). Automatic construction of simple artifact-based business processes. In *Proceedings of the 12th international conference on database theory* (pp. 225–238).
- Gerede, C. E., Bhattacharya, K. & Su, J. (2007). Static analysis of business artifact-centric operational models. In *Ieee international conference on service-oriented computing and applications (soca'07)* (pp. 133–140).
- Gerede, C. E. & Su, J. (2007). Specification and verification of artifact behaviors in business process models. In *International conference on service-oriented computing* (pp. 181–192).
- Goedertier, S., Vanthienen, J. & Caron, F. (2015). Declarative business process modelling: principles and modelling languages. *Enterprise Information Systems*, 9(2), 161–185.
- Gonzalez, P., Griesmayer, A. & Lomuscio, A. (2012). Verifying gsm-based business artifacts. In *2012 ieee 19th international conference on web services* (pp. 25–32).
- Gonzalez, P., Griesmayer, A. & Lomuscio, A. (2013). Model checking gsm-based multi-agent systems. In *International conference on service-oriented computing* (pp. 54–68).
- Gottschalk, F., van der Aalst, W. M. & Jansen-Vullers, M. H. (2008). Merging event-driven process chains. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 418–426).
- Goyal, V. & Jain, A. (2011). Service-oriented architecture & its concept unleashed. *Advances in Modeling, Optimization and Computing*, 967.
- Grech, A. & Camilleri, A. F. (2017). *Blockchain in education*. Luxembourg: Publications Office of the European Union.
- Hackius, N. & Petersen, M. (2017). Blockchain in logistics and supply chain: trick or treat? In *Proceedings of the hamburg international conference of logistics (hicl)* (pp. 3–18).

- Haisjackl, C. & Zugal, S. (2014). Investigating differences between graphical and textual declarative process models. In *International conference on advanced information systems engineering* (pp. 194–206).
- Hamdan, O., Shanableh, H., Zaki, I., Al-Ali, A. & Shanableh, T. (2019). Iot-based interactive dual mode smart home automation. In *2019 ieee international conference on consumer electronics (icce)* (pp. 1–2).
- Hariri, B. B., Calvanese, D., De Giacomo, G., De Masellis, R. & Felli, P. (2011). Foundations of relational artifacts verification. In *International conference on business process management* (pp. 379–395).
- Hariri, B. B., Calvanese, D., Montali, M., Santoso, A. & Solomakhin, D. (2013). Verification of semantically-enhanced artifact systems. In *International conference on service-oriented computing* (pp. 600–607).
- Harold Boley, A. P. S. T. B. G. N. B. G. G. F. O. D. H., Tara Athan. (2014). *Specification of deliberation ruleml 1.0*. Retrieved from <http://ruleml.org/1.0/>
- Heath, F. T., Boaz, D., Gupta, M., Vaculín, R., Sun, Y., Hull, R. & Limonad, L. (2013). Barcelona: A design and runtime environment for declarative artifact-centric bpm. In *International conference on service-oriented computing* (pp. 705–709).
- Huang, Y., Li, W., Liang, Z., Xue, Y. & Wang, X. (2018). Efficient business process consolidation: combining topic features with structure matching. *Soft Computing*, 22(2), 645–657.
- Hull, R. (2008). Artifact-centric business process models: Brief survey of research results and challenges. *On the Move to Meaningful Internet Systems (OTM)*., 1152–1163.
- Hull, R. (2011). Towards flexible service interoperation using business artifacts. In *2011 ieee 15th international enterprise distributed object computing conference* (pp. 20–21).
- Hull, R. (2017). Blockchain: Distributed event-based processing in a data-centric world. In *Proceedings of the 11th acm international conference on distributed and event-based systems* (pp. 2–4).
- Hull, R., Batra, V. S., Chen, Y.-M., Deutsch, A., Heath III, F. F. T. & Vianu, V. (2016). Towards a shared ledger business collaboration language based on data-aware processes. In *International conference on service-oriented computing* (pp. 18–36).
- Hull, R., Damaggio, E., De Masellis, R., Fournier, F., Gupta, M., Heath III, F. T., ... others (2011). Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In *Proceedings of the 5th acm international conference on distributed event-based system* (pp. 51–62).
- Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath, F. T., Hobson, S., ... others (2010). Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In *International workshop on web services and formal methods* (pp. 1–24).
- Hull, R., Narendra, N. C. & Nigam, A. (2009). Facilitating workflow interoperation using artifact-centric hubs. In *Service-oriented computing* (pp. 1–18). Springer.
- Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C.,

- ... others (2017). The internet-of-things meets business process management: mutual benefits and challenges. *arXiv preprint arXiv:1709.03628*.
- Jard, C. & Jeron, T. (1989). On-line model-checking for finite linear temporal logic specifications. In *International conference on computer aided verification* (pp. 189–196).
- Jensen, K. & Kristensen, L. M. (2009). *Coloured petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media.
- Jiang, P., Shao, X., Gao, L., Qiu, H. & Li, P. (2010). A process-view approach for cross-organizational workflows management. *Advanced Engineering Informatics*, 24(2), 229–240.
- Johnson, R., Pearson, D. & Pingali, K. (June, 1994). The program structure tree: Computing control regions in linear time. In (pp. 171–185). Florida, USA: ACM.
- Jonnnavithula, L., Antunes, P., Cranefield, J. & Pino, J. A. (2015). Organisational issues in modelling business processes: An activity-based inventory and directions for research. In *Pacis* (p. 184).
- Kakas, A. C., Kowalski, R. A. & Toni, F. (1992). Abductive logic programming. *Journal of logic and computation*, 2(6), 719–770.
- Kang, G., Yang, L. & Zhang, L. (2019). Verification of behavioral soundness for artifact-centric business process model with synchronizations. *Future Generation Computer Systems*.
- Kavantzas, N., Burdett, D., Ritzinger, G. & Lafon, Y. (2005). *Web services choreography description language version 1.0, w3c candidate recommendation* (Tech. Rep.). Technical report, November 2005. <http://www.w3.org/TR/ws-cdl-10>.
- Keller, G., Scheer, A.-W. & Nüttgens, M. (1992). *Semantische prozeßmodellierung auf der grundlage" ereignisgesteuerter prozeßketten (epk)"*. Inst. für Wirtschaftsinformatik.
- Kiepuszewski, B., Ter Hofstede, A. H. M. & Bussler, C. J. (2000). On structured workflow modelling. In *International conference on advanced information systems engineering* (pp. 431–445).
- Kindler, E. (2004). On the semantics of epcs: A framework for resolving the vicious circle. In *International conference on business process management* (pp. 82–97).
- Kökörcený, M. & Kovář, V. (2015). Building enterprise applications using unicorn universe services. In *Service-oriented computing-icsoc 2014 workshops* (pp. 3–5).
- Koutsos, A. & Vianu, V. (2017). Process-centric views of data-driven business artifacts. *Journal of Computer and System Sciences*, 86, 82–107.
- Kovář, V., Beránek, M. & Feuerlicht, G. (2017). Modelling enterprise applications using business artifacts. In *Iceis 2017-proceedings of the 19th international conference on enterprise information systems*.
- Kucukoguz, E. & Su, J. (2010). On lifecycle constraints of artifact-centric workflows. In *International workshop on web services and formal methods* (pp. 71–85).
- Kumaran, S., Liu, R. & Wu, F. Y. (2008). On the duality of information-centric and activity-centric models of business processes. In *International conference on*

- advanced information systems engineering* (pp. 32–47).
- Kumaran, S., Nandi, P., Heath, T., Bhaskaran, K. & Das, R. (2003). Adoc-oriented programming. In *2003 symposium on applications and the internet, 2003. proceedings.* (pp. 334–341).
- Kumaran, S. B., Liu, R. & Wu, F. Y.-F. (2012, December 25). *Automatic generation of executable components from business process models.* Google Patents. (US Patent 8,340,999)
- Kunchala, J., Yu, J., Sheng, Q. Z., Han, Y. & Yongchareon, S. (2015). Synthesis of artifact lifecycles from activity-centric process models. In *Enterprise distributed object computing conference (edoc), 2015 ieee 19th international* (pp. 29–37).
- Kunchala, J., Yu, J. & Yongchareon, S. (2014). A survey on approaches to modeling artifact-centric business processes. In *International conference on web information systems engineering* (pp. 117–132).
- Kunchala, J., Yu, J., Yongchareon, S. & Han, Y. (2017). Towards merging collaborating processes for artifact lifecycle synthesis. In *Proceedings of the australasian computer science week multiconference* (p. 50).
- Kunchala, J., Yu, J., Yongchareon, S. & Liu, C. (2019). An approach to merge collaborating processes of an inter-organizational business process for artifact lifecycle synthesis. *Computing*, 1–26.
- Kunchala, J., Yu, J., Yongchareon, S. & Wang, G. (2020). Trace-based approach for consistent construction of activity-centric process models from data-centric process models. In *Australasian database conference* (pp. 42–54).
- Kunze, M., Berger, P., Weske, M., Lohmann, N. & Moser, S. (2012). Bpm academic initiative-fostering empirical research. In (pp. 1–5). Tallinn, Estonia: CEUR-WS.org.
- Künzle, V. & Reichert, M. (2011). Philharmonicflows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4), 205–244.
- Küster, J. M., Gerth, C., Förster, A. & Engels, G. (2008). Detecting and resolving process model differences in the absence of a change log. In (pp. 244–260). Milan, Italy: Springer.
- Küster, J. M., Ryndina, K. & Gall, H. C. (2007). Generation of business process models for object life cycle compliance. In (pp. 165–181). Brisbane, Australia: Springer.
- La Rosa, M., Dumas, M., Uba, R. & Dijkman, R. (2010). Merging business process models. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 96–113).
- La Rosa, M., Dumas, M., Uba, R. & Dijkman, R. (2013). Business process model merging: An approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 22(2), 11.
- Lei, J., Bai, R., Guo, L. & Zhang, L. (2016). Towards a scalable framework for artifact-centric business process management systems. In *International conference on web information systems engineering* (pp. 309–323).
- Leymann, F. et al. (2001). Web services flow language (wsfl 1.0).
- Li, D. & Wu, Q. (2011). Translating artifact-based business process model to bpm. In

- International conference on computer science, environment, ecoinformatics, and education* (pp. 482–489).
- Li, G., de Carvalho, R. M. & van der Aalst, W. M. (2017). Automatic discovery of object-centric behavioral constraint models. In *International conference on business information systems* (pp. 43–58).
- Li, Y., Deutsch, A. & Vianu, V. (2017a). Spinart: A spin-based verifier for artifact systems. *arXiv preprint arXiv:1705.09427*.
- Li, Y., Deutsch, A. & Vianu, V. (2017b). A spin-based verifier for artifact systems. *arXiv preprint arXiv:1705.09427*.
- Limonad, L., Boaz, D., Hull, R., Vaculin, R. & Heath, F. (2012). A generic business artifacts based authorization framework for cross-enterprise collaboration. In *2012 annual srii global conference* (pp. 70–79).
- Lin, D. (2007). Compatibility analysis of local process views in interorganizational-workflow. In *The 9th ieee international conference on e-commerce technology and the 4th ieee international conference on enterprise computing, e-commerce and e-services (cec-eee 2007)* (pp. 149–156).
- Liu, D. R. & Shen, M. (2003). Workflow modeling for virtual processes: an order-preserving process-view approach. *Information Systems*, 28(6), 505–532.
- Liu, D. R. & Shen, M. (2004). Business-to-business workflow interoperation based on process-views. *Decision Support Systems*, 38(3), 399–419.
- Liu, G., Liu, X., Qin, H., Su, J., Yan, Z. & Zhang, L. (2009). Automated realization of business workflow specification. In *Service-oriented computing. icsoc/service-wave 2009 workshops* (pp. 96–108).
- Liu, R., Bhattacharya, K. & Wu, F. Y. (2007). Modeling business contexture and behavior using business artifacts. In *International conference on advanced information systems engineering* (pp. 324–339).
- Liu, R., Wu, F. Y. & Kumaran, S. (2010). Transforming activity-centric business process models into information-centric models for soa solutions. *Journal of Database Management (JDM)*, 21(4), 14–34.
- Lohmann, N. & Nyolt, M. (2011). Artifact-centric modeling using bpmn. In *International conference on service-oriented computing* (pp. 54–65).
- Lohmann, N. & Wolf, K. (2010). Artifact-centric choreographies. In *International conference on service-oriented computing* (pp. 32–46).
- Ltd, V. P. I. (2002). *Visual paradigm*. Retrieved from <https://www.visual-paradigm.com>
- Lu, X., Nagelkerke, M., van de Wiel, D. & Fahland, D. (2015). Discovering interacting artifacts from erp systems. *IEEE Transactions on Services Computing*, 8(6), 861–873.
- Lu, Y., Papagiannidis, S. & Alamanos, E. (2018). Internet of things: A systematic review of the business literature from the user and organisational perspectives. *Technological Forecasting and Social Change*, 136, 285–297.
- Luckham, D. C., Park, D. M. R. & Paterson, M. S. (1970). On formalised computer programs. *Journal of Computer and System Sciences*, 4(3), 220–249.

- Maamar, Z., Badr, Y. & Narendra, N. C. (2010). Business artifacts discovery and modeling. In *Service-oriented computing - 8th international conference, ICSOC 2010, san francisco, ca, usa, december 7-10, 2010. proceedings* (pp. 542–550).
- MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F., Metz, R. & Hamilton, B. A. (2006). Reference model for service oriented architecture 1.0. *OASIS standard*, 12, 18.
- Manoj, A. S., Hussain, M. A. & Teja, P. S. (2019). Patient health monitoring using iot. In *Mobile health applications for quality healthcare delivery* (pp. 30–45). IGI Global.
- Marrella, A., Mecella, M., Russo, A., Steinau, S., Andrews, K. & Reichert, M. (2015). Data in business process models, a preliminary empirical study (short paper). In *2015 ieee 8th international conference on service-oriented computing and applications (soca)* (pp. 116–122).
- Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., ... others (2004). Owl-s: Semantic markup for web services. *W3C member submission*, 22(4).
- Mendling, J. & Hafner, M. (2008). From ws-cdl choreography to bpm process orchestration. *Journal of Enterprise Information Management*, 21(5), 525–542.
- Mendling, J. & Simon, C. (2006). Business process design by view integration. In *International conference on business process management* (pp. 55–64).
- Mendling, J., Weber, I., Aalst, W. V. D., Brocke, J. V., Cabanillas, C., Daniel, F., ... others (2018). Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 9(1), 4.
- Meroni, G., Baresi, L., Montali, M. & Plebani, P. (2018). Multi-party business process compliance monitoring through iot-enabled artifacts. *Information Systems.*, 73, 61–78.
- Meroni, G., Di Ciccio, C., Mendling, J. et al. (2017). Artifact-driven process monitoring: dynamically binding real-world objects to running processes. In *Caise 2017 forum* (pp. 105–112).
- Mettler, M. (2016). Blockchain technology in healthcare: The revolution starts here. In *2016 ieee 18th international conference on e-health networking, applications and services (healthcom)* (pp. 1–3).
- Meyer, A., Smirnov, S. & Weske, M. (2011). *Data in business processes* (No. 50). Universitätsverlag Potsdam.
- Meyer, A. & Weske, M. (2013). Activity-centric and artifact-centric process model roundtrip. In (pp. 167–181). Springer.
- Meyer, A. & Weske, M. (September, 2013). Extracting data objects and their states from process models. In (pp. 27–36). Vancouver, Canada: IEEE.
- Mikk, E., Lakhnechi, Y. & Siegel, M. (December, 1997). Hierarchical automata as model for statecharts. In (pp. 181–196). Kathmandu, Nepal: Springer.
- Milner, R. (1999). *Communicating and mobile systems: the pi calculus*. Cambridge university press.

- Milner, R., Parrow, J. & Walker, D. (1992). A calculus of mobile processes, i. *Information and computation*, 100(1), 1–40.
- Montali, M., Pesic, M., van der Aalst, W. M., Chesani, F., Mello, P. & Storari, S. (2010). Declarative specification and verification of service choreographies. *ACM Transactions on the Web (TWEB)*, 4(1), 3.
- Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. et al. (2009). Owl 2 web ontology language profiles. *W3C recommendation*, 27, 61.
- Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 541–580.
- Nandi, P. & Kumaran, S. (2005). Adaptive business objects-a new component model for business integration. In *Iceis (3)* (pp. 179–188).
- Natschläger, C. (August, 2011). Deontic bpmn. In (pp. 264–278). Toulouse, France: Springer.
- Ngamakeur, K., Yongchareon, S. & Liu, C. (2012). A framework for realizing artifact-centric business processes in service-oriented architecture. In *International conference on database systems for advanced applications* (pp. 63–78).
- Nigam, A. & Caswell, N. S. (2003). Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3), 428–445.
- Nofer, M., Gomber, P., Hinz, O. & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, 59(3), 183–187.
- Nooijen, E. H., van Dongen, B. F. & Fahland, D. (2012). Automatic discovery of data-centric and artifact-centric processes. In *International conference on business process management* (pp. 316–327).
- Norta, A., Grefen, P. & Narendra, N. C. (2014). A reference architecture for managing dynamic inter-organizational business processes. *Data & Knowledge Engineering*, 91, 52–89.
- OMG, B. P. M. N. (2006). Version 1.0. *OMG Final Adopted Specification, Object Management Group*, 190.
- OMG, O. (2007). Unified modeling language. *Superstructure*.
- Orlowska, W. S. M. E. (1997). On correctness issues in conceptual modeling of workflows. In *Proceedings of the 5th european conference on information systems [c]*.
- Ouyang, C., Dumas, M., Breutel, S. & ter Hofstede, A. (2006). Translating standard process models to bpel. In *International conference on advanced information systems engineering* (pp. 417–432).
- Ouyang, C., Dumas, M., Ter Hofstede, A. H. & Van der Aalst, W. M. (2006). From bpmn process models to bpel web services. In *2006 ieee international conference on web services (icws'06)* (pp. 285–292).
- Ouyang, C., van der Aalst, W. M., Dumas, M. & Ter Hofstede, A. H. (2006). Translating bpmn to bpel.
- Ouyang, C., Verbeek, E., Van Der Aalst, W. M., Breutel, S., Dumas, M. & Ter Hofstede, A. H. (2007). Formal semantics and analysis of control flow in ws-bpel. *Science of computer programming*, 67(2-3), 162–198.

- Panchatcharam, P. & Vivekanandan, S. (2019). Internet of things (iot) in healthcare—smart health and surveillance, architectures, security analysis and data transfer: A review. *International Journal of Software Innovation (IJSI)*, 7(2), 21–40.
- Peltz, C. (2003). Web services orchestration and choreography. *Computer*(10), 46–52.
- Pesic, M., Schonenberg, H. & Van der Aalst, W. M. (2007). Declare: Full support for loosely-structured processes. In *11th ieee international enterprise distributed object computing conference (edoc 2007)* (pp. 287–287).
- Petrasch, R. & Hentschke, R. (2015). Towards an internet-of-things-aware process modeling method. In *2nd manag. innov. technol. int. conf. towar* (pp. 168–172).
- Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J. & Reijers, H. A. (2011). Imperative versus declarative process modeling languages: An empirical investigation. In *International conference on business process management* (pp. 383–394).
- Polyvyanyy, A., García-Bañuelos, L. & Dumas, M. (2010). Structuring acyclic process models. In (pp. 276–293). New Jersey, USA: Springer.
- Polyvyanyy, A., García-Bañuelos, L. & Dumas, M. (2012). Structuring acyclic process models. *Information Systems*, 37(6), 518–538.
- Polyvyanyy, A., Smirnov, S. & Weske, M. (2009). On application of structural decomposition for process model abstraction. In (pp. 110–122). Leipzig, Germany: GI.
- Popova, V. & Dumas, M. (2012). From petri nets to guard-stage-milestone models. In *International conference on business process management* (pp. 340–351).
- Popova, V. & Dumas, M. (2013). Discovering unbounded synchronization conditions in artifact-centric process models. In *International conference on business process management* (pp. 28–40).
- Popova, V., Fahland, D. & Dumas, M. (2013). Artifact lifecycle discovery. *CoRR*, abs/1303.2554.
- Popova, V., Fahland, D. & Dumas, M. (2015). Artifact lifecycle discovery. *International Journal of Cooperative Information Systems*, 24(01), 1550001.
- Prescher, J., Di Ciccio, C. & Mendling, J. (2014). From declarative processes to imperative models. *SIMPDA*, 14, 162–173.
- Pundir, A. K., Jagannath, J. D. & Ganapathy, L. (2019). Improving supply chain visibility using iot-internet of things. In *2019 ieee 9th annual computing and communication workshop and conference (ccwc)* (pp. 0156–0162).
- Redding, G., Dumas, M., Hofstede, A. H. t. & Iordachescu, A. (2008). Generating business process models from object behavior models. *Information Systems Management*, 25(4), 319–331.
- Redding, G., Dumas, M., ter Hofstede, A. H. & Iordachescu, A. (2010). A flexible, object-centric approach for business process modelling. *Service Oriented Computing and Applications*, 4(3), 191–201.
- Reypens, C., Lievens, A. & Blazevic, V. (2016). Leveraging value in multi-stakeholder innovation networks: A process framework for value co-creation and capture. *Industrial Marketing Management*, 56, 40–50.

- Russo, A., Mecella, M., Patrizi, F. & Montali, M. (2013). Implementing and running data-centric dynamic systems. In *2013 IEEE 6th international conference on service-oriented computing and applications* (pp. 225–232).
- Ryndina, K., Küster, J. M. & Gall, H. C. (2006). Consistency of business process models and object life cycles. In (Vol. 4364, pp. 80–90). Genoa, Italy: Springer.
- Sathiyaraj, R. & Balamurugan, B. (2018). Iot based intelligent transportation system (iot-its) for global perspective: A case study. *Internet of Things and Big Data Analytics for Smart Generation*, 154, 279.
- Schönig, S., Ackermann, L., Jablonski, S. & Ermer, A. (2018). An integrated architecture for iot-aware business process execution. In *Enterprise, business-process and information systems modeling* (pp. 19–34). Springer.
- Schuldt, H., Alonso, G., Beer, C. & Schek, H.-J. (2002). Atomicity and isolation for transactional processes. *ACM Transactions on Database Systems (TODS)*, 27(1), 63–116.
- Schulz, K. A. & Orlowska, M. E. (2004). Facilitating cross-organisational workflows with a workflow view approach. *Data & Knowledge Engineering*, 51(1), 109–147.
- Schunselaar, D. M., Leopold, H., Verbeek, H., van der Aalst, W. M. & Reijers, H. A. (2014). Configuring configurable process models made easier: an automated approach. In *International conference on business process management* (pp. 105–117).
- Sheng-nan, L., Pei-pei, D., Jian-li, F. & Xiao-he, L. (2015). The implementation of intelligent transportation system based on the internet of things. *J. Chem. Pharm. Res.*, 7(3), 1074–1077.
- Siadat, S. H., Shokohyar, S. & Shafahi, S. (2019). SOA adoption factors in e-banking: An empirical analysis from the practical perspective. *IJISSS*, 11(1), 25–39.
- Sivagami, S., Revathy, D. & Nithyabharathi, L. (2016). Smart health care system implemented using iot. *International Journal of Contemporary Research in Computer Science and Technology*, 2(3).
- Solomakhin, D., Montali, M., Tessaris, S. & De Masellis, R. (2013). Verification of artifact-centric systems: Decidability and modeling issues. In *International conference on service-oriented computing* (pp. 252–266).
- Staples, M., Chen, S., Falamaki, S., Ponomarev, A., Rimba, P., Tran, A., ... Zhu, J. (2017). *Risks and opportunities for systems using blockchain and smart contracts. data61*. CSIRO), Sydney.
- Strosnider, J. K., Nandi, P., Kumaran, S., Ghosh, S. & Arsnajani, A. (2008). Model-driven synthesis of soa solutions. *IBM Systems Journal*, 47(3), 415–432.
- Sukaviriya, N., Mani, S. & Sinha, V. (2009). Reflection of a year long model-driven business and ui modeling development project. In *Ifip conference on human-computer interaction* (pp. 749–762).
- Sun, S., Kumar, A. & Yen, J. (2006). Merging workflows: A new perspective on connecting business processes. *Decision Support Systems*, 42(2), 844–858.
- Sun, Y., Xu, W. & Su, J. (2012). Declarative choreographies for artifacts. In *International conference on service-oriented computing* (pp. 420–434).

- Tian, F. (2017). A supply chain traceability system for food safety based on haccp, blockchain & internet of things. In *2017 international conference on service systems and service management* (pp. 1–6).
- Turkanović, M., Hölbl, M., Košič, K., Heričko, M. & Kamišalić, A. (2018). Eductx: A blockchain-based higher education credit platform. *IEEE access*, 6, 5112–5127.
- Van Der Aalst, W., Van Hee, K. M. & van Hee, K. (2004). *Workflow management: models, methods, and systems*. MIT press.
- Van der Aalst, W. M. (1998). The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01), 21–66.
- Van der Aalst, W. M. (2013). Business process management: a comprehensive survey. *ISRN Software Engineering*, 2013.
- Van Der Aalst, W. M. & Basten, T. (2002). Inheritance of workflows: an approach to tackling problems related to change. *Theoretical Computer Science*, 270(1-2), 125–203.
- van der Aalst, W. M., De Medeiros, A. A. & Weijters, A. (2006). Process equivalence: Comparing two process models based on observed behavior. In *International conference on business process management* (pp. 129–144).
- Van Der Aalst, W. M., Lohmann, N., Massuthe, P., Stahl, C. & Wolf, K. (2010). Multiparty contracts: Agreeing and implementing interorganizational processes. *The Computer Journal*, 53(1), 90–106.
- Van Der Aalst, W. M. & Pesic, M. (2006). Decserflow: Towards a truly declarative service flow language. In *International workshop on web services and formal methods* (pp. 1–23).
- van Der Aalst, W. M., Pesic, M. & Schonenberg, H. (2009). Declarative workflows: Balancing between flexibility and support. *Computer Science-Research and Development*, 23(2), 99–113.
- Van Der Aalst, W. M. & Ter Hofstede, A. H. (2005). Yawl: yet another workflow language. *Information systems*, 30(4), 245–275.
- Van der Aalst, W. M., van Dongen, B. F., Günther, C. W., Rozinat, A., Verbeek, E. & Weijters, T. (2009). Prom: The process mining toolkit. *BPM (Demos)*, 489(31), 2.
- van der Aalst, W. M. & Weske, M. (2001). The p2p approach to interorganizational workflows. In *International conference on advanced information systems engineering* (pp. 140–156).
- Van der Aalst, W. M., Weske, M. & Grünbauer, D. (2005). Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2), 129–162.
- van Eck, M. L., Sidorova, N. & van der Aalst, W. M. (2017). Guided interaction exploration in artifact-centric process models. In *2017 ieee 19th conference on business informatics (cbi)* (Vol. 1, pp. 109–118).
- Vanhatalo, J., Völzer, H. & Koehler, J. (2009). The refined process structure tree. *Data & Knowledge Engineering*, 68(9), 793–818.
- Vanhatalo, J., Völzer, H. & Leymann, F. (2007). Faster and more focused control-flow analysis for business process models through sese decomposition. In (pp. 43–55).

- Vienna, Austria: Springer.
- W3C. (2005). *Web services choreography description language version 1.0*. Retrieved from <https://www.w3.org/TR/ws-cdl-10/>
- Wang, J. & Kumar, A. (2005). A framework for document-driven workflow systems. In *International conference on business process management* (pp. 285–301).
- White, S. A. (2004). Introduction to bpmn. *Ibm Cooperation*, 2(0), 0.
- Wieczorek, S., Roth, A., Stefanescu, A., Kozyura, V., Charfi, A., Kraft, F. M. & Schieferdecker, I. (2009). Viewpoints for modeling choreographies in service-oriented architectures. In *2009 joint working ieee/ifip conference on software architecture & european conference on software architecture* (pp. 11–20).
- Xu, W., Su, J., Yan, Z., Yang, J. & Zhang, L. (2011). An artifact-centric approach to dynamic modification of workflow execution. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 256–273).
- Ye, C., Cheung, S.-C., Chan, W. K. & Xu, C. (2009). Atomicity analysis of service composition across organizations. *IEEE Transactions on Software Engineering*, 35(1), 2–28.
- Yongchareon, S. & Liu, C. (2010). A process view framework for artifact-centric business processes. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 26–43).
- Yongchareon, S., Liu, C. & Zhao, X. (2011). An artifact-centric view-based approach to modeling inter-organizational business processes. In *International conference on web information systems engineering* (pp. 273–281).
- Yongchareon, S., Liu, C., Zhao, X. & Xu, J. (2010). An artifact-centric approach to generating web-based business process driven user interfaces. In *International conference on web information systems engineering* (pp. 419–427).
- Yongchareon, S., Liu, C., Zhao, X., Yu, J., Ngamakeur, K. & Xu, J. (2018). Deriving user interface flow models for artifact-centric business processes. *Computers in Industry*, 96, 66–85.
- Yongchareon, S., Ngamakeur, K., Liu, C., Chaisiri, S. & Yu, J. (2014). A workflow execution platform for collaborative artifact-centric business processes. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 639–643).
- Yongchareon, S., Yu, J., Zhao, X. et al. (2015). A view framework for modeling and change validation of artifact-centric inter-organizational business processes. *Information systems.*, 47, 51–81.
- Zafar, I., Azam, F., Anwar, M. W., Butt, W. H., Maqbool, B. & Nazir, A. K. (2018). Business process models to web services generation: A systematic literature review. In *2018 ieee 9th annual information technology, electronics and mobile communication conference (iemcon)* (pp. 789–794).
- Zaha, J. M., Barros, A., Dumas, M. & ter Hofstede, A. (2006). Let's dance: A language for service behavior modeling. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 145–162).
- Zemni, M. A., Mammar, A. & Hadj-Alouane, N. B. (2016). An automated approach for merging business process fragments. *Computers in Industry*, 82, 104–118.

- Zhao, D., Liu, G., Jiang, Y., Gao, F. & Wang, Y. (2011). The execution and detection of artifact-centric business process. In *2011 ieee international conference on computer science and automation engineering* (Vol. 4, pp. 491–495).
- Zhao, D., Liu, G., Wang, Y., Gao, F., Li, H. & Zhang, D. (2011). A-stein: A prototype for artifact-centric business process management systems. In *2011 international conference on business management and electronic information* (Vol. 1, pp. 247–250).
- Zhao, X. & Liu, C. (2006). Tracking over collaborative business processes. In *International conference on business process management* (pp. 33–48).
- Zhao, X., Liu, C., Sadiq, W. & Kowalkiewicz, M. (2008). Process view derivation and composition in a dynamic collaboration environment. In *Otm confederated international conferences" on the move to meaningful internet systems"* (pp. 82–99).
- Zhao, X., Liu, C., Sadiq, W., Kowalkiewicz, M. & Yongchareon, S. (2009). Ws-bpel business process abstraction and concretisation. In *International conference on database systems for advanced applications* (pp. 555–569).
- Zhao, X., Liu, C., Sadiq, W., Kowalkiewicz, M. & Yongchareon, S. (2011). Implementing process views in the web service environment. *World Wide Web*, 14(1), 27–52.
- Zhao, X., Liu, C. & Yang, Y. (2005). An organisational perspective on collaborative business processes. In *International conference on business process management* (pp. 17–31).
- Zhao, X., Su, J., Yang, H. & Qiu, Z. (2009). Enforcing constraints on life cycles of business artifacts. In *2009 third ieee international symposium on theoretical aspects of software engineering* (pp. 111–118).

Appendix A

Appendix

This section presents the complete recruitment process model, a fragment of its process tree, the lifecycle of position artifact of this process model and the XML specification of BPMN process model related to Chapter 3. The complete AirTravel process model related to Chapter 4, and the constructed BPMN model of the ACP purchasing process model related to Chapter 5 are also presented.

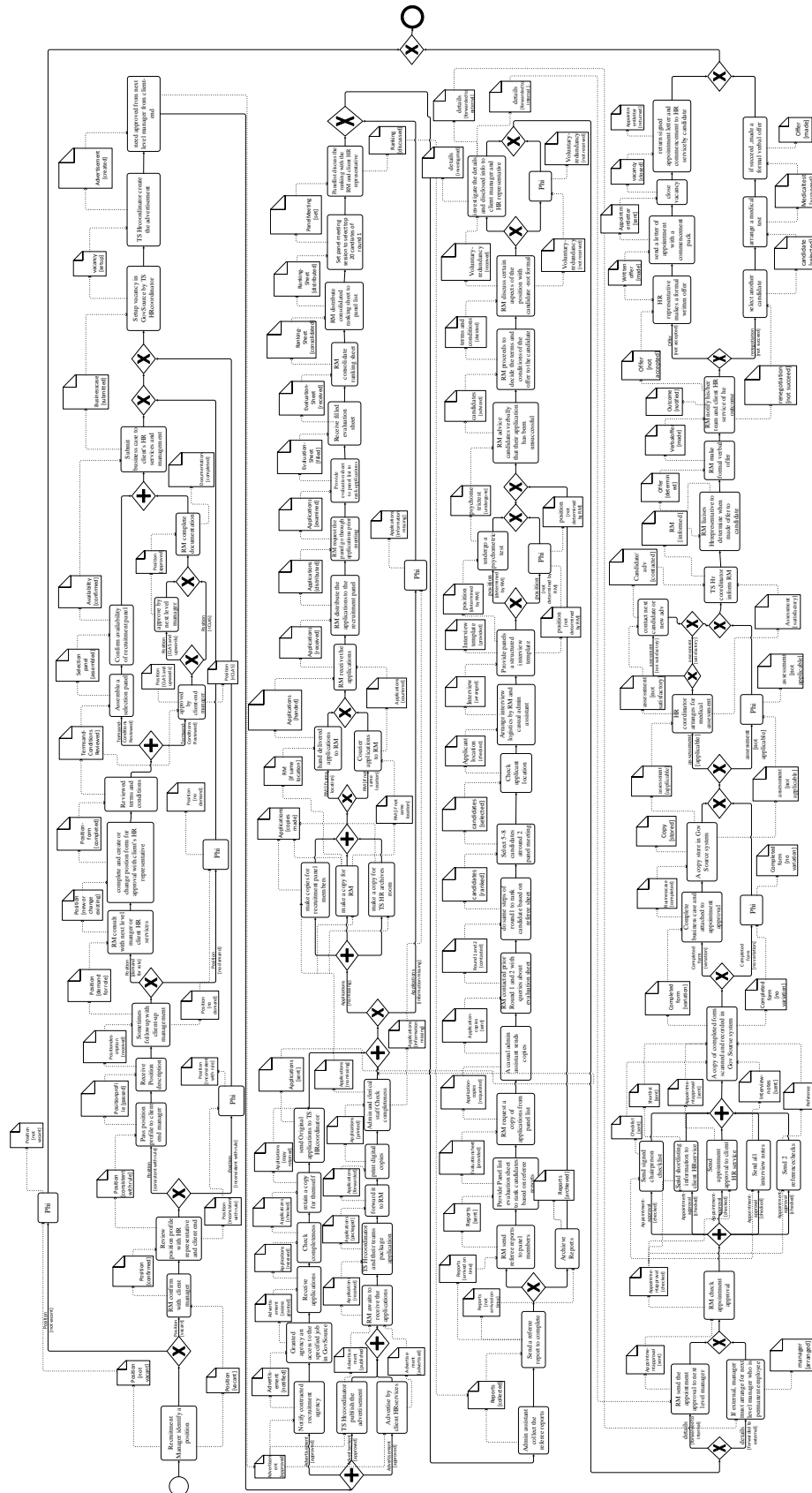


Figure A.1: Complete Recruitment process model

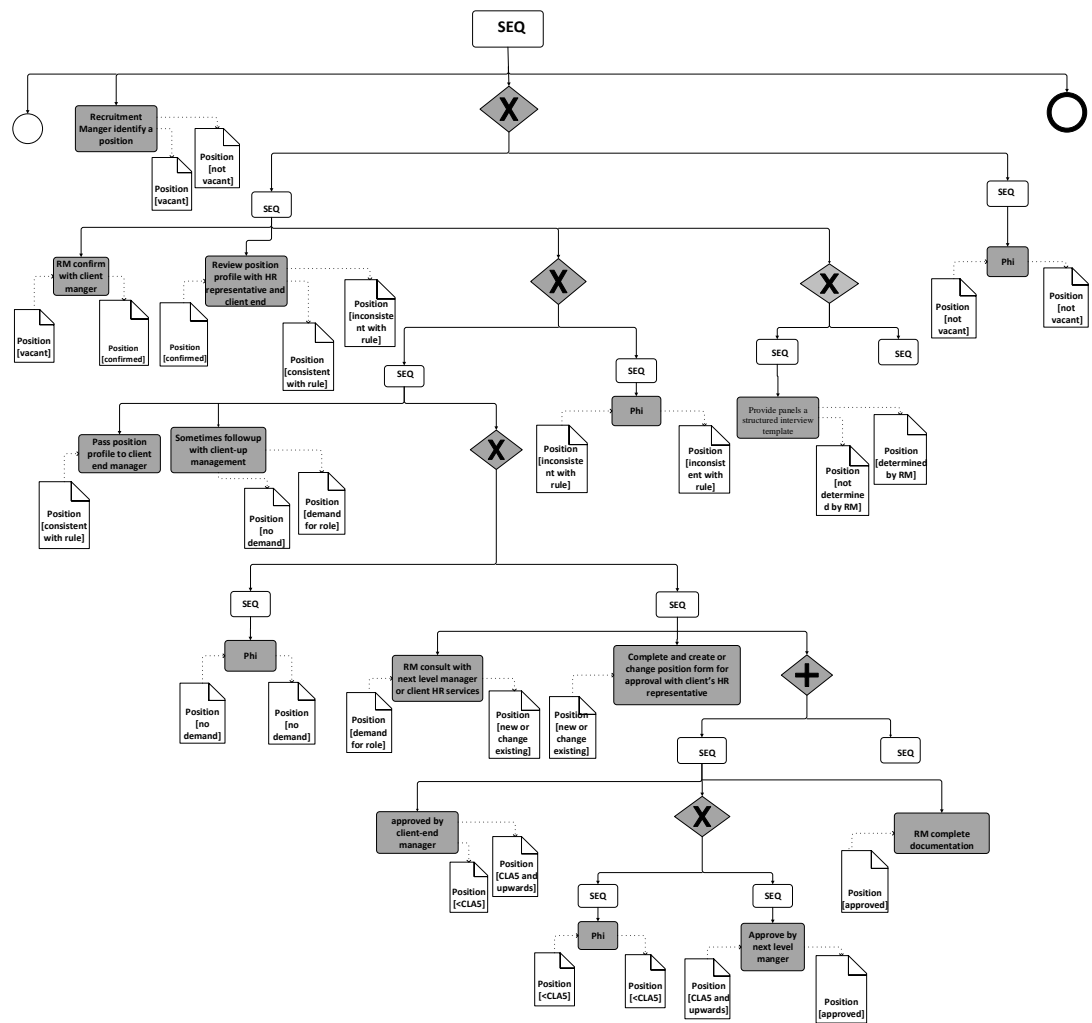


Figure A.2: Process tree fragment of Recruitment process model

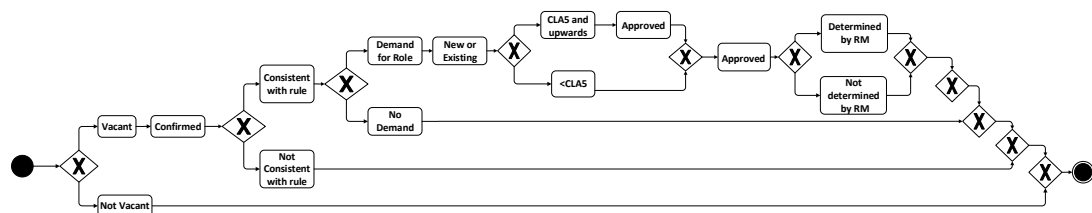


Figure A.3: Lifecycle of Position artifact


```

<?xml version="1.0"?>
<process>
  <artifacts>
    <artifact name="Order" id="_1"/>
    <artifact name="Products" id="_2"/>
    <artifact name="Invoice" id="_3"/>
  </artifacts>
  <processdef>
    <startEvent name="init"/>
    <task name="Receive Order">
      <inputartifact id="_1" state="init"/>
      <outputartifact id="_1" state="received"/>
    </task>
    <task name="Analyze Order">
      <inputartifact id="_1" state="received"/>
      <outputartifact>
        <art1 id="_1" state="rejected"/>
        <art2 id="_1" state="confirmed"/>
      </outputartifact>
    </task>
    <gateway name="xorsplit" id="G_1">
      <branch>
        <task name="Check Stock">
          <inputartifact>
            <art1 id="_2" state="init"/>
            <art2 id="_1" state="confirmed"/>
          </inputartifact>
          <outputartifact>
            <art1 id="_1" state="checked"/>
            <art2 id="_2" state="not in stock"/>
            <art3 id="_2" state="in stock"/>
          </outputartifact>
        </task>
        <gateway name="xorsplit" id="G_2">
          <inputartifact>
            <art1 id="_2" state="not in stock"/>
            <art2 id="_2" state="in stock"/>
          </inputartifact>

```

Figure A.4: XML Specification of BPMN Process Model



Figure A.5: AirTravel Process Model

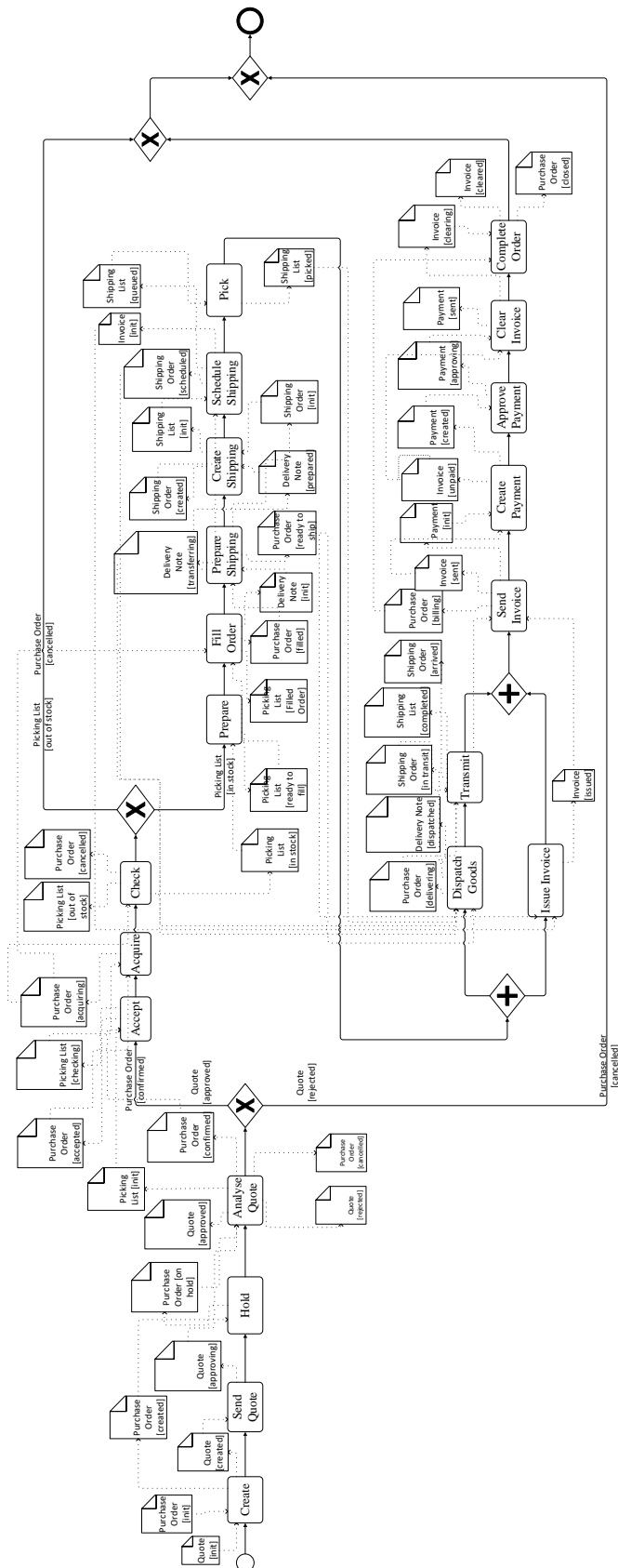


Figure A.6: The constructed Activity-Centric Model of Purchasing Process