Hybrid Composition of Microservices: A Metrics-based Analysis

By

Razibul Hasan

Supervised By

Prof. Morad Benyoucef

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the

Master of Science in Digital Transformation and Innovation

University of Ottawa

Abstract

"Microservices" is an architectural and organizational style in software design and development in which there is a composition mechanism for independent microservices to call, communicate, and message each other within an application. The microservices composition approach makes design easier to scale, faster to develop, and can accelerate the introduction of new features into applications. To satisfy business requirements, selecting the proper composition style is important for software development; otherwise, application development may fail.

The objective of this research is to investigate the hybrid method for composing microservices and compare it with other composition approaches (choreography and orchestration), using quality metrics from the software engineering and business process modeling literature. More precisely, we make use of coupling, cohesion, and scalability metrics to analyze BPMN models representing e-commerce scenarios modeled as microservice compositions.

This thesis follows the five steps: research problem identification and objectives, requirement analysis and system design, model design and development, model testing and deployment, and evaluate our BPMN models representing microservice compositions. We develop multiple BPMN workflows as artifacts to analyze choreography, orchestration, and hybrid styles for the microservices composition of e-commerce scenarios. We propose several hybrid models by integrating orchestrations and choreographies in the same workflow.

We created a series of small, mid-sized, and end-to-end workflows of e-commerce scenarios. At the tool level, we use the Camunda Modeler, Camunda Platform 8 (as the automation process engine), and Amazon Web Services (AWS) to design, develop, and deploy our models.

Finally, we use our calculations of the coupling, cohesion, and scalability measures to reveal an understanding of modeling microservices choreography, orchestration, and hybrid approaches, and we discuss when to use a specific approach for microservices composition. We have found from the evaluation that our proposed models are less tightly coupled compared to those modeled using orchestration and choreography. However, we also discovered that the orchestration style offers better scalability and a lower ratio of coupling and cohesion compared to choreography and hybrid approaches.

Acknowledgments

First, I would like to thank God for giving me strength and an opportunity to achieve my academic goal. I would also like to express my sincere gratitude to several people without whom I would not have been able to complete this thesis.

I would like to express my special thanks and most profound appreciation to my supervisor, Professor Morad Benyoucef, for his valuable guidance, efforts, and encouragement throughout the research. I have always enjoyed our friendly relationship and have always valued his advice. My academic journey and research were possible only through his effective guidance and direction, while at the same time giving me the freedom to explore on my own. Thank you for your advice on my research and my life.

I am grateful to Professor Daniel Amyot and Professor Hussein Al Osman for their valuable time, encouragement, feedback, and guidance for the past years for this thesis. I am thankful to Professor Umar Ruhi for his comments and feedback on this thesis proposal. I also offer thanks to the microservices online community for their support and collaboration.

Most importantly, this journey would not have been possible without the sacrifice and patience of my entire family, to whom I offer thanks for giving me strength and constant inspiration.

Finally, I want to express my gratitude to my fellow lab members for always being there, when needed, with support and assistance related to my research and my personal life, their motivation, friendship, and support were especially important during the time of the pandemic.

Table of Contents

ABSTRACT .... **ii**
ACKNOWLEDGMENTS .... **iii**
TABLE OF CONTENTS .... **iv**
LIST OF TABLES .... **vii**
LIST OF FIGURES .... **viii**
LIST OF ACRONYMS .... **ix**
CHAPTER 1. INTRODUCTION **1**
1.1 WHAT IS SERVICE-ORIENTED ARCHITECTURE? 2
1.2 WHY MICROSERVICE ARCHITECTURE (MSA)? 2
1.3 MICROSERVICES COMPOSITION 3
1.4 WHY THE E-COMMERCE DOMAIN? 3
1.5 PROBLEM STATEMENT 4
1.6 RESEARCH OBJECTIVES 6
1.7 RESEARCH QUESTIONS 6
1.8 RESEARCH CONTRIBUTIONS 6
1.9 ORGANIZATION OF THESIS 7
CHAPTER 2. RESEARCH DESIGN AND METHODOLOGY **8**
2.1 RESEARCH METHODOLOGY 8
2.2 RESEARCH DESIGN AND METHODS 10
CHAPTER 3. A LITERATURE REVIEW ON MICROSERVICES COMPOSITION **12**
3. LITERATURE REVIEW 12
3.1 SEARCH STRATEGY 12
3.1.1 Source Databases *13*
3.1.2 Search Query *13*
3.1.3 Exclusion and Inclusion Criteria *13*
3.1.4 Final Selection of Peer-reviewed Articles *13*
3. 2 MICROSERVICES COMPOSITION APPROACHES 15
3.3 DATA COLLECTION FOR RQ1 AND RQ2 16
3.4 SYNTHESIS 24
3.5 DATA RESULTS 25
3.5.1 Advantages and Limitation of Microservices Orchestration, Choreography, and Hybrid (RQ1): *25*

3.5.2 Hybrid Techniques for Microservices Composition (RQ2): *27*
3.6 DISCUSSION 27
3.7 SUMMARY 28
CHAPTER 4. BPMN MODELS OF MICROSERVICES 30
4.1 CREATION OF BPMN MODELS USING CAMUNDA MODELER 30
4.1.2 Why have we opted for the Camunda Platform? **..** *30*
4.2 BPMN ELEMENTS IN CAMUNDA MODELER 31
4.3 E-COMMERCE SCENARIOS FOR BPMN MODELS 31
4.4 DESIGNING CHOREOGRAPHY-BASED BPMN MODELS 32
4.5 DESIGNING ORCHESTRATION-BASED BPMN MODELS 33
4.6 DESIGNING HYBRID BPMN MODELS 33
4.7 SMALL-SIZED BPMN MODELS 33
4.7.1 Checkout Process *34*
4.7.2 User Authentication *35*
4.7.3 Shipment *36*
4.7.4 Payment Process *38*
4.7.5 Order Process *39*
4.8 MID-SIZED BPMN MODELS 40
4.9 END-TO-END BPMN MODELS 43
4.9.1 Strategy for Developing Hybrid Composition *49*
4.10 DEPLOYMENT OF BPMN MODELS 50
CHAPTER 5. Quality Metrics in Software Engineering and Evaluation of Microservices BPMN Models 52
5.1 INTRODUCTION 52
5.2 QUALITY METRICS IN SOFTWARE PROGRAMS AND BUSINESS PROCESS MODELING 53
5.3 ADAPTATION OF METRICS IN SOFTWARE PROGRAMS AND BUSINESS PROCESS MODELING 54
5.4 WHY COUPLING, COHESION, AND SCALABILITY FOR EVALUATING MODELS? 55
5.5 COUPLING METRICS ADAPTATION 55
5.5.1 What are Imported and Exported Coupling in a Process? *56*
5.5.2 Coupling Metrics Measurement for BPMN Models *56*
5.5.3 Measurement of Imported and Exported Coupling in a Process *57*
5.5.4 Calculating Process Coupling *58*
5.6 COHESION METRICS ADAPTATION 59

5.6.1 Cohesion Metrics Measurement for BPMN Models ... *59*
5.7 COUPLING/COHESION RATIO ... 60
5.8 THE CONTROL FLOW COMPLEXITY (CFC) ... 61
5.9 SCALABILITY METRICS IN BUSINESS PROCESS MODELING ... 61
5.9.1 Scale Value Comparison ... *63*
5.9.2 Scalability Measurement for BPMN Models ... *63*
5.9.3 Calculating the Structural Aspect ... *64*
5.9.4 Calculating the Behavioral Aspect ... *64*
5.9.5 Simulated Average Value of Similarity ... *65*
5.9.6 Determine the Control Flow Complexity and the Scale Value ... *65*
5.9.7 Business Process Models Scale Value Comparison ... *66*
5.9.8 Comparison of the Scalability Metric Results ... *66*
5.10 DISCUSSION ... 67
CHAPTER 6. CONCLUSIONS AND FUTURE WORK ... 69
6.1 DISCUSSION ... 69
6.2 CONCLUSION ... 71
6.3 RESEARCH CONTRIBUTIONS ... 72
6.4 LIMITATIONS AND FUTURE WORK ... 73
REFERENCES ... 75
APPENDICES ... 80

List of Tables

Table 1: Research Gap........ 5
Table 2: Addressed Research Questions, Methods use, and Output 11
Table 3: List of Publications Selected for Microservices Composition 16
Table 4: Microservices Composition Styles Advantages and Limitations 19
Table 5: Similarities in Software Designs and Business Process Modeling (Vanderfeesten et al., 2007)... 54
Table 6: Matching between Object Oriented Software and BPMN (Khlif et al., 2009) 54
Table 7: Comparison of ICP and ECP Metrics Results for Composition Models 57
Table 8: Process Model and Process Coupling (cp) 58
Table 9: Process models and Process Cohesion (ch) 60
Table 10: Process models and Coupling Vs Cohesion Ratio........ 60
Table 11: Elements, Transition Adjacency Relations, and Control Flow Complexity........ 64
Table 12: Results of the Structural Similarity Calculation with Jaccard Coefficient 64
Table 13: Results of the Behavioral Similarity Calculation with Jaccard Coefficient 65
Table 14: Results of the average value of Structural and Behavioral similarity 65
Table 15: Control Flow Complexity and the Scale Value 66
Table 16: Business Process Models Scale Value Comparison 66
Table 17: Business Process Models Scalability Comparison 67

List of Figures

Figure 1. The five steps of our research 8
Figure 2. Papers selection process PRISMA guidelines (Moher et al., 2009) 14
Figure 3. The Chronological Overview of the Selected Papers 15
Figure 4. Mind Map for Microservices Compositions 25
Figure 5. Choreography Model for Checkout Process 34
Figure 6. Orchestration Model for Checkout Process 34
Figure 7. Choreography Model for User Authentication 35
Figure 8. Orchestration Model for User Authentication 36
Figure 9. Choreography Model for Shipment Process 37
Figure 10. Orchestration Model for Shipment 38
Figure 11. Choreography Model for Payment Process 39
Figure 12. Orchestration Model for Order Process 39
Figure 13. Choreography Model for Checkout-User Authentication Process 40
Figure 14. Orchestration Model for Checkout-User Authentication Process 41
Figure 15. Choreography Model for Shipment and Payment Process 42
Figure 16. Hybrid Model with one Orchestration and one Choreography 43
Figure 17. Choreography Model for end-to-end e-Commerce Scenarios 44
Figure 18. Orchestration Model for end-to-end e-Commerce Scenarios 45
Figure 19. Hybrid (A) Model with one Orchestration and three Choreographies 46
Figure 20. Hybrid (B) Model with two Orchestrations and two Choreographies 47
Figure 21. Hybrid (C) Model with three Orchestrations and one Choreography 48

List of Acronyms

Acronym	Definition
AMQP	Advanced Messaging Queuing Protocol
API	Application Programming Interface
AWS	Amazon Web Services
B2B	Business-to-Business
BPMN	Business Process Modeling Notation
CFC	Control-Flow Complexity
CH	Cohesion
CP	Coupling
CPSS	Cyber-Physical-Social System
DSL	Domain Specific Language
E-COMMERCE	Electronic Commerce
EC	Exported Coupling
EC2	Elastic Compute Cloud
ECP	Exported Coupling of a Process
EKS	Elastic Kubernetes Service
IC	Imported Coupling
ICP	Imported Coupling of a Process
IEEE	Institute of Electrical and Electronics Engineers
JVM	Java Virtual Machine

MSA	Microservice Architecture
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
REST	Representational State Transfer
REST API	Representational State Transfer Application Programming Interface
RQ	Research Question
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TAR	Transition Adjacency Relations
UML	Unified Modeling Language
WSDL	Web Services Description Language
XML	Extensible Markup Language
XOR	Exclusive Gateway

Chapter 1. Introduction

Microservices represent a new trend and a novel variant of service-oriented architecture (SOA) that emphasizes the design and development of scalable software that is also highly maintainable. Microservices constitute an architectural approach that structures applications as a collection of small, autonomous services, designed around a certain business logic (Asri et al., 2022; Davide Taibi, Lenarduzzi, & Pahl, 2018). It represents a software design style in which complex applications are composed of small, independent collections of services that communicate among each other. Microservices are autonomous services each of which performs independently where services are developed, deployed, tested, upgraded, and scaled around a business capability (Lewis & Fowler, 2014; Nadeem & Malik, 2022; Volynsky et al., 2022). Thus, Microservice Architecture (MSA) is an architectural paradigm that ties together many highly cohesive but loosely coupled services (Dragoni et al., 2017; Müller, 2019; Rademacher et al., 2018).

MSA performs independently and communicates via messaging and Representational State Transfer Application Programming Interface (REST API), whereas SOAs are more flexible in terms of utilizing various messaging protocols, such as Simple Object Access Protocol (SOAP), and Advanced Messaging Queuing Protocol (AMQP) for communication. MSA can be built using a wide range of technologies and languages, based upon a few basic principles: bounded context, size, and small services being autonomous of each other (Hasselbring & Steinacker, 2017). Microservices in such an architecture are especially well-suited for building, testing, deploying, operating, and monitoring software (Savchenko et al., 2018). The key principles of MSA include structural automation, business capability, decentralized control, and design for failure (Butzin, Golatowski, & Timmermann, 2016; Lewis & Fowler, 2014; Pahl & Jamshidi, 2016). Some architectural styles have been proposed and used to design microservice-based applications, which have been developed, tested, deployed, and operated by small teams (Richardson, 2020).

Since microservices composition is a style that combines lightly coupled services together to deliver a solution for business requirements, each service performs its corresponding task for ensuring the proper functioning of the entire application. As such, it becomes crucially important to define a communication mechanism between and among the microservices for the application(s). Any medium to large-sized organization usually will have thousands of distributed microservices across its business applications landscape (A. H. Filho et al., 2021; Taibi et al., 2018). To offer valuable services to users, while minimizing coupling and reducing complexity among the microservices, effective control becomes an important factor for such distributed microservices. They must be composed in a way that satisfies the business requirements (Ortiz, Torres, & Valderas, 2020).

The main goal of this thesis is to investigate a hybrid composition style for microservices by adapting the good aspects of choreography and orchestration and assessing our resulting model's using coupling, cohesion, and scalability metrics. We use e-commerce applications as our research domain; hence our models represent common e-commerce scenarios.

1.1 What is Service-Oriented Architecture?

Service-oriented architecture (SOA) is an architectural style in software engineering; it defines a way for software architectures to leverage components, called services, to create applications and SOAP, AMQP, and other protocols as communication protocols ("What Is Service-Oriented Architecture?" n.d.). It offers several benefits over monolithic applications, such as faster time to market, maintenance efficiency, and better adaptability.

1.2 Why Microservice Architecture (MSA)?

Microservice architecture creates interesting opportunities for software development. MSA enforces explicit requirements on service granularity for exactly one business capability in an application (de Castro & Rigo, 2023; Rademacher et al., 2018). The main advantages of microservices are their scalability in collaborative distributed scenarios, as well as the enhanced possibilities for service development, testing, and operation (Alpers et al., 2015; Nebel et al., 2021; Sorgalla et al., 2018; Rademacher et al., 2019). Microservices-based application scaling is easier than for a monolithic application (Davide et al., 2017). The core concepts are that the microservices are loosely coupled with other services and have high reusability. The reusability of an application defines the utilization of identical microservices to provide solutions to different tasks across the system. The loosely coupled features allow microservices to solve the complexity of any application by using its reusability (Artamonov & Sukhodolov, 2018). Service-oriented architecture promotes sharing concepts as much as possible. Microservice architecture, in contrast, is designed based on the principle of minimizing the sharing of components as much as possible (Mark, 2016).

1.3 Microservices Composition

Microservices composition is a technique that defines how multiple microservices in an application collaborate with each other in order to accomplish a business goal (Ali et al., 2022). In general, microservices use mainly two different composition approaches: orchestration (centralized) and choreography (decentralized) control techniques. In communicating between services, orchestration is synchronous, whereas choreography is asynchronous by nature (Asri et al., 2022; Bigheti et al., 2019; Valderas et al., 2020). Also, in an orchestration-based approach, all communication is performed within the system where the orchestrator (i.e., central controller) is responsible for it. All requests and responses are managed by the orchestrator using a request/response message mechanism for communication. Alternatively, choreography distributes its control mechanism, primarily as a listen/react for communication.

1.4 Why the e-Commerce Domain?

There is significant growth in the popularity of microservices in software development, and many tech giants as well as e-commerce companies have adopted microservice architectures to build applications for their business, e.g., Uber, Amazon, eBay, Apple, otto.de, Airbnb, Netflix, PayPal, LinkedIn, Twitter and many others (Hasselbring & Steinacker, 2017; Taibi et al., 2018; Karwatka et al., 2018; Kappagantula, 2019).

The first reason to select an e-commerce domain for this research is to help us investigate the appropriate model of microservice architecture for delivering critical new features in e-commerce applications and a wide range of business functionalities. E-commerce is one of the software industries that can benefit extensively from microservice architectures since e-commerce applications are built on multiple services that address user needs. In this digital era, consumer demands are dynamic (i.e., anywhere, anytime, and on any service). Microservices are transforming e-commerce application development by allowing hundreds of teams to seamlessly work in parallel; new features or versions of services can be released within hours rather than months (Goetsch, 2017). The second reason is that we have identified examples in the literature related to the e-commerce application domain, which leads us to believe it would be helpful to compare our own scenarios with other scenarios identified in the literature.

Moreover, the literature discusses the breakdown of monolithic e-commerce applications into scalable microservices-based applications, demonstrating how both registered and anonymous users browse through many product categories and product detail pages, and how microservices serve those different requests (Christudas, 2019; Hasselbring & Steinacker, 2017; Koyya & Muthukumar, 2022). In addition, re-architecting the monolithic, integration techniques (i.e., continuous integration and continuous delivery), deployment of microservices, and data management, as well as the pros and cons of microservices for e-commerce are presented by Karwatka et al. (2018). The literature describes how to use microservices and modeling for placing an order in an online shop (Valderas et al., 2020; Ortiz et al., 2020). Distributed transactions in a microservice architecture and saga pattern of e-commerce applications are defined (Rudrabhatla, 2018). A business-to-business (B2B) e-commerce microservice-based architecture consisting of a business layer, service, data access layer, and data management layer is described (Wu, Ding, & Hou, 2019).

E-commerce applications have increased in recent years as a way for organizations to promote their business and increase profits. By implementing microservices, they can overcome the problem of service scalability and reusability when building e-commerce applications (Asri et al., 2022; Asrowardi, Putra, & Subyantoro, 2020; de Castro & Rigo, 2023; Shuo et al., 2023).

Finally, selecting e-commerce applications as a domain in this research provides us with a complex e-commerce application structure, with the need to modify components frequently while sustaining usability. By using e-commerce applications, we are able to implement and study scenarios, starting from simple to end-to-end, and look at them from the points of view of both the enterprise and the consumer.

1.5 Problem Statement

Although MSA is being increasingly used for software development, it is still challenging for organizations to execute the appropriate approach to meet requirements for microservices composition. With regards to current knowledge and implementation gaps, we have found that most of the literature focuses on developing an architecture based on microservices via either a choreography or an orchestration-based composition approach. However, some studies propose their own language for service composition in MSA in facing these challenges.

We have found several papers discussing open-source modeling tools for microservices orchestration, but they do not compare composition styles from a design point of view (Cao, 2020;

Koschel & Schulze, 2022; Rademacher et al., 2018; Hamidehkhan & Wurster, 2019; Rademacher et al., 2020; Rücker, 2019; Shuo et al., 2023; Valderas et al., 2020).

Concerning empirical research in the microservice composition domain, there are only a few relevant contributions to hybrid composition. Just as surprising, we have found only a few studies that focus on microservices coupling and cohesion, and these studies do not look at how communication between microservices affects how they work together or how well they can handle growth. Furthermore, no one has evaluated these concepts in either theoretical or industrial-level adaptation (Valderas et al., 2020). To the best of our knowledge, we identify the following research gaps from the literature:

Table 1: Research Gap

Related research	**Research gap**	**Description of the gap from the related works**
Hybrid composition	Previous works are generally selective in choosing choreography or orchestration for microservices architecture composition. Less attention has been given to hybrid composition in prior research. From previous studies, it is unclear in which context and when to use the hybrid approach for microservices composition.	The EUCalipTool is a mobile application targeted to end-users without programming skills and allows them to easily compose services. By using EUCalipTool, however, it is difficult to do industrial-level hybrid microservice composition due to a lack of scalability features that requires composing thousands of services for a large organization. EUCalipTool acts as an orchestrator between BPMN models and users (Valderas et al., 2020). A hybrid pattern used for communication between microservices and SOA is considered by Taibi et al. for migration purposes (Taibi et al., 2018); the hybrid pattern brings together the service registry and API gateway, and it swaps out the API gateway with the message bus as introduced by Kazanaviˇcius & Maˇzeika (2023); however, we did not find any indication on whether or not this presents a hybrid composition approach for microservice architecture. One simple way to use a hybrid approach, combine choreography and orchestration and presented as basic diagrams without justification described by Singhal et al. (2019a).

		Although the above-mentioned studies touch on the technical aspects of the hybrid method of MSA-based systems and their adaptation in the industry, the unique features of the hybrid composition approach remain largely unexplored.

1.6 Research Objectives

This research has the following objectives:

1. **achieve a good understanding of choreography, orchestration, and hybrid compositions of microservices and the differences between them,**
2. **build a comprehensive understanding of the hybrid composition of microservices taking advantage of the benefits of both the choreography and orchestration approaches and how and when to use the hybrid composition in applications; and**
3. **design and implement hybrid models for microservices and analyze them based on metrics.**

1.7 Research Questions

We aim to address the research gap we have identified in the previous section. The foremost goal of our research is to design a hybrid method for composing microservices, taking advantage of the orchestration and choreography approaches; and to analyze resulting models. The research aims to focus on the following questions:

- **RQ1: What are the advantages and disadvantages of microservices orchestration, choreography, and hybrid composition?**
- **RQ2: In what situation should we use the hybrid techniques for microservices composition?**
- **RQ3: How does hybrid composition compare to choreography and orchestration in terms of coupling, cohesion, scaling, and complexity for e-commerce systems?**

1.8 Research Contributions

The study aims to enhance the understanding of e-commerce systems development practices by examining the most effective microservice composition style (orchestration, choreography, and hybrid) for developing e-commerce applications. To achieve this, we create BPMN models that implement different e-commerce scenarios using choreography, orchestration, and hybrid approaches. We then evaluate and compare these models using coupling, cohesion, and scalability metrics.

This study provides researchers and practitioners with guidelines and strategies for creating hybrid models to integrate microservices, helping them choose the right composition tools. We have made our models, artifacts, and codes accessible online for researchers and practitioners to use. These resources can be reused and applied to recreate the models as needed.

Furthermore, this thesis makes an academic contribution by addressing the limited research on assessing and comparing the microservice composition hybrid style. It introduces a novel approach of hybrid composition by combining choreography and orchestration, utilizing multiple orchestrators within each model. The models and analysis presented in this study can be valuable to a wide range of professionals, including enterprise users, business analysts, and software designers. The outcomes of our research will benefit both academic researchers seeking knowledge on microservice composition and practitioners involved in software development across many industries.

1.9 Organization of Thesis

This thesis is structured as follows: the second chapter presents the Research Design and Methodology which we used in this research. The third chapter focuses on a review of microservices composition styles in the literature and their benefits and limitations. In the fourth chapter, we present three categories of designed models of microservices for e-commerce scenarios using BPMN. The fifth chapter focuses on the quality metrics identified in the areas of software engineering and business process modeling. In this chapter, we also analyze and compare the developed BPMN models using various metrics, including coupling, cohesion, and scalability. Finally, the sixth chapter presents our conclusions and outlines areas for future work.

Chapter 2. Research Design and Methodology

This Chapter discusses the process of planning and conducting the research, including the steps and methods, as well as the overall research plan.

2.1 Research Methodology

We follow five steps in our research. These steps are described in in **Figure 1**.

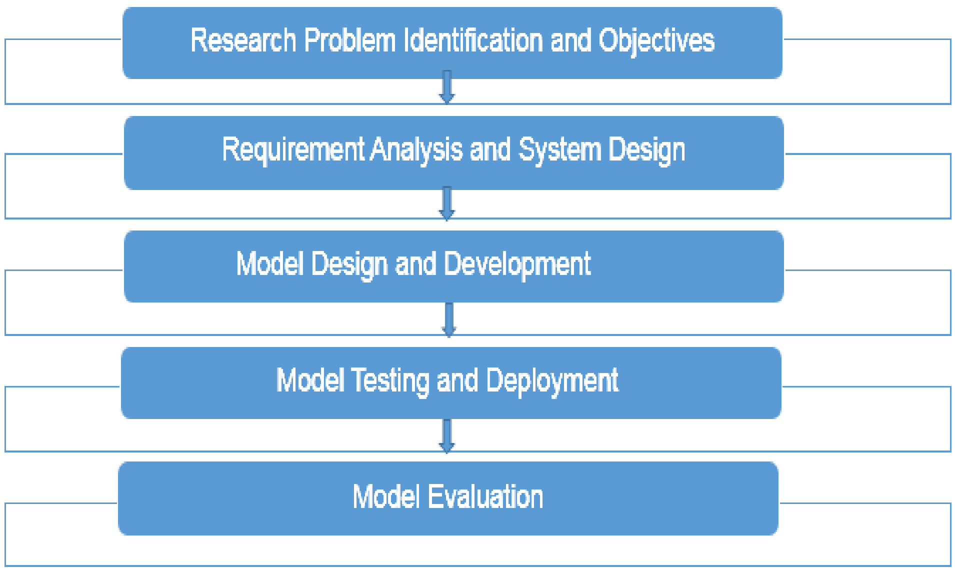


Figure 1. The five steps of our research

1. **Research Problem Identification and Objectives:**
   First, we identify the research problem and research questions, comparing advantages (scalability, coupling, and cohesion) of microservices composition: hybrid vs. choreography and orchestration in the development of e-Commerce applications, based on recent publications. Our research objectives are focused on designing, evaluating, and comparing composition styles of microservices, using BPMN models in the e-Commerce domain. The lack of attention to the hybrid approach and its benefits is the main motivation for these research questions.
2. **Requirement Analysis and System Design:**
   We analyse our main requirements based on the research objectives and focusing on system design selecting composition tools for our models considering composition styles of

microservices. To understand the requirements more clearly, we initially figured out the main ideas for microservices composition. These include different ways of designing systems and their characteristics, how microservices work together (using orchestration, choreography, and hybrid methods). Then, we performed a thorough review of the existing literature, known as a Systematic Literature Review (SLR). Insights will guide developers in deciding on model design tools, and where and when to apply which composition approach to satisfy their business requirements.

3. **Model Design and Development:**

   Next, we design and develop e-Commerce applications based on the business logic of e-commerce scenarios, combining both the choreography and orchestration approaches into a single workflow to propose a hybrid composition style for the microservices. To this end, we follow these steps: a) Study multiple e-Commerce websites; b) Identify, document, and classify e-Commerce scenarios into three categories (small-sized, mid-sized, and end-to-end); c) Use BPMN 2.0 to model choreography and orchestration for each scenario from the 3 categories. We use choreography and orchestration techniques to create models for small scenarios, the hybrid style requires a more complicated scenario to design BPMN models.

4. **Model Testing and Deployment:**

   In this fourth step, we will use the Camunda BPMN Modeler, Kubernetes, Camunda automation process engine and Amazon Web Services (AWS) to design workflows, deploy them on the cloud, and test whether they are working properly or not. When the number of microservices increases, the direct communication between microservices reduces scalability. The process engine can address this challenge by keeping composition logic and their implementation separate for these microservices (Hamidehkhan & Wurster, 2019). In our demonstration (development and deployment) process, we follow the BPMN modelling guidelines in Camunda Platform 8 to design our models within the choreographies, orchestrations, and hybrid microservices, representing the communication mechanisms among the microservices and the characteristics of each composition approach.

5. **Model Evaluation:**

Next, we evaluate our hybrid composition using quality metrics (scalability, coupling, and cohesion) found in the literature from the software engineering and workflow domains. First, we import designed models into the AWS cloud for deployment and then complete workflow instances. Second, we use quality metrics (details on the coupling, cohesion, and scalability metrics, please refer to Chapter 5) to assess and compare our models and thereby draw insights.

We went back to step 3 of the model design and development process, where we added new similarity measures and integrated the new models into the proposed e-Commerce scenario, after assessing the initial models.

2.2 Research Design and Methods

This section presents our approach to addressing the research questions identified previously.

The research design offers an overall plan for the research and the research methodology provides the systematic processes we follow. Outcomes of this type research are always an important artifact, whether it is a model, method, product, tool or process (Venable et al., 2017). We use a five phase research methodology for this research since we are seeking to produce models.

We use the inductive search approach to search related articles for microservice composition styles, and then we test models using choreography, orchestration, and hybrid approaches for artifact development which is a deductive method.

The first research question (**RQ1**) is exploratory inductive research. In inductive inquiry, we start with data collection to build a theory. To answer our research questions (**RQ1**) and achieve objectives, we conduct a literature review to generate ideas and extract a quick understanding. For example, we compared the selected studies based on composition type, advantages, and limitations of microservice composition types, and modeling tools.

The second research question (**RQ2**) is exploratory inductive research. Here, we explain in what situations developers should use the hybrid techniques and when to avoid them (P Valderas et al., 2020). To answer this question, we garner support from the literature and produce models.

Table 2: Addressed Research Questions, Methods use, and Output

Research questions	Methods used to answer	Output
RQ1	Literature Review	Theory building, concepts, models, modeling tools, pros and cons of choreography, orchestration, and hybrid; design the initial models
RQ2	Literature Review	Modify the models and concepts, provide guidelines
RQ3	Model Design and Development, Model Testing and Deployment, Evaluation	Existing models testing, creating new models, proposing hybrid models, validating models with quality metrics

The third research question (**RQ3**) is descriptive deductive research, and we want to answer this question by testing models (deductive) and designing models from the literature by applying the research methods. We identify concepts (combining choreography and orchestration) for composing a hybrid approach within a single workflow (Singhal et al., 2019a). We model and then validate those models with the help of quality metrics from the literature (Vanderfeesten et al., 2007). This is the cause-and-effect type part of our research.

Chapter 3. A Literature Review on Microservices Composition

This chapter provides a literature review of microservices composition styles. It aims to identify, summarize, and compare current composition approaches. The literature review contributes to this thesis by providing an overview of previously published, relevant, peer-reviewed papers on our research topic.

3. Literature Review

3.1 Search Strategy

As a search strategy, our exploratory search is based on the research questions of our thesis. There are three key concepts: microservices, architecture, and composition.

First, the keywords of each concept are extracted based on these three concepts. In the next step, we add the synonyms for each keyword by applying related background information and brainstorming. These synonyms (micro-service, micro service; orchestration, choreography, hybrid; architecting, model, design) of the three key concepts are used to form a preliminary Boolean logic search query that is entered into specific databases. A total of four bibliographic databases are used.

For consistency, the search query is applied to the title, keywords, and abstracts of papers in all databases in this literature review. The preliminary automatic search query is entered into different databases: Web of Science, Scopus, ACM Digital Library and IEEE Xplore. A pearl growing strategy is executed to enrich the search query after a few main papers are found (Francesco et al., 2019).

In order to avoid missing a number of relevant papers during the search, snowballing is used. Backward snowballing focuses on all the references of each considered study, whereas for the forward snowballing, Google Scholar is used to obtain those papers citing the current one.

The main 3 core concepts are microservices, architecture, and composition, which we use to generate synonyms. Each query line contains synonyms for each key concept.

We have found that wildcards have limitations in the IEEE Xplore Digital Library database search: the maximum number of wildcards is limited to five. Therefore, we use alternate search strings for these keywords “workflow OR workflows OR design OR designs”, and the rest of the keywords remain the same for this specific database search.

3.1.1 Source Databases

The concept of microservices architecture is technical and organized around business capabilities. Therefore, we use IEEE Xplore and ACM Digital Library in the field of engineering and computer science-related publications. The Scopus and Web of Science databases, in general, cover all the areas of publications, including business, management, and health science.

3.1.2 Search Query

The finalized search query is the following:

(microservice* OR micro-service* OR "micro service") AND (choreography OR orchestration OR hybrid) AND (composition* OR architect* OR design* OR model*)

3.1.3 Exclusion and Inclusion Criteria

Exclusion: in our search we did not consider non-peer-reviewed articles. We also excluded papers that were not written in the English language; papers about definition-based microservices but which do not focus on composition or architectural solutions, and papers that were published in 2013 or before, since the concept of microservice architectures first appeared in the year 2014.

Inclusion: important papers on microservices architectural styles were identified using backward snowballing of seminal papers.

3.1.4 Final Selection of Peer-reviewed Articles

By applying the search query to all the above databases and screening papers, using Covidence online tools, we found total of 594 unique results (after removing duplicate articles shown in **Figure 2**). Based on the search query, with inclusion and exclusion criteria applied to the title and abstract, and screening out any irrelevant results, 65 important papers were selected. The next step was full-text screening and quality assessment, by which 26 papers were selected for consideration in the literature review.

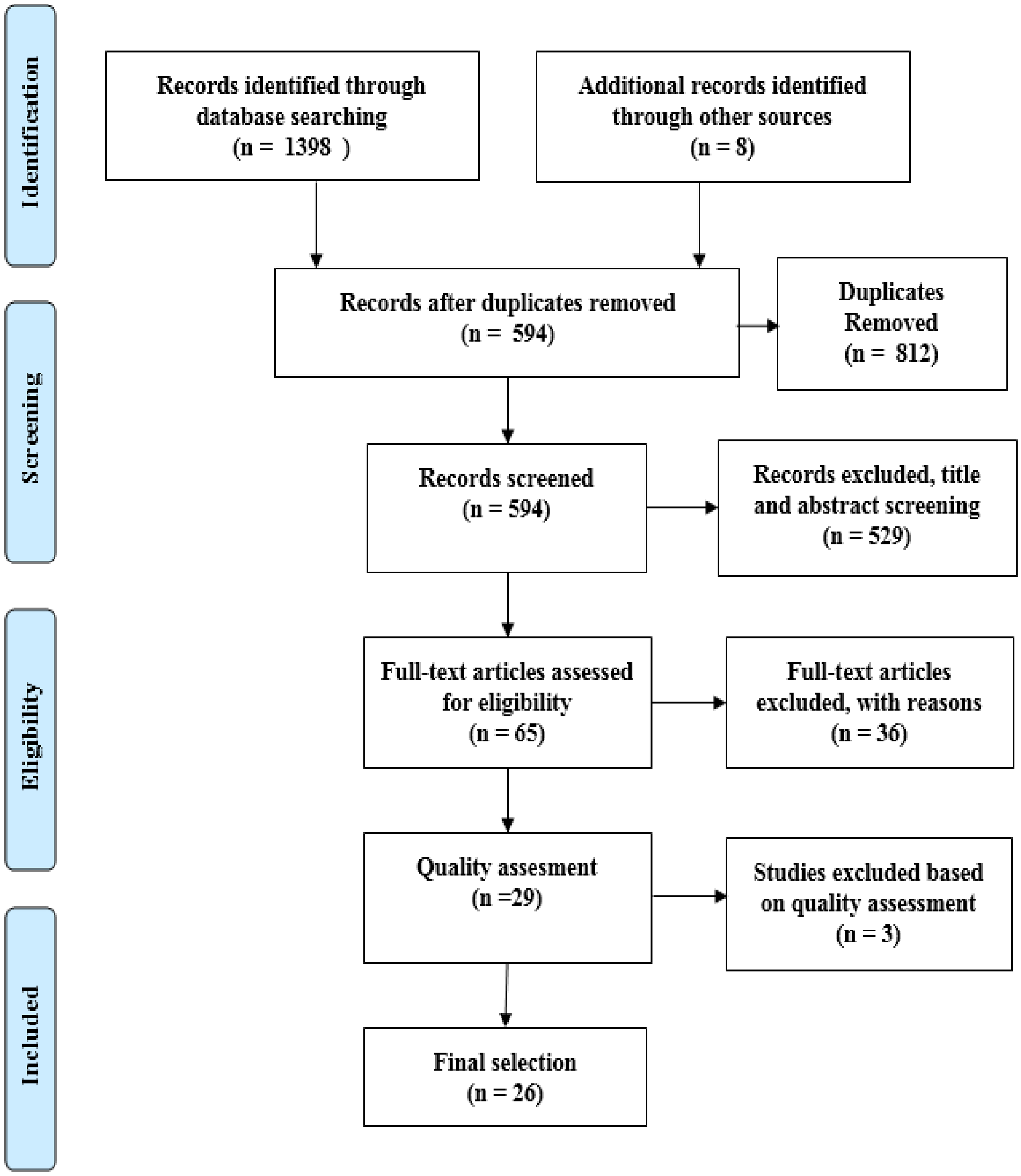


Figure 2. Papers selection process PRISMA guidelines (Moher et al., 2009)

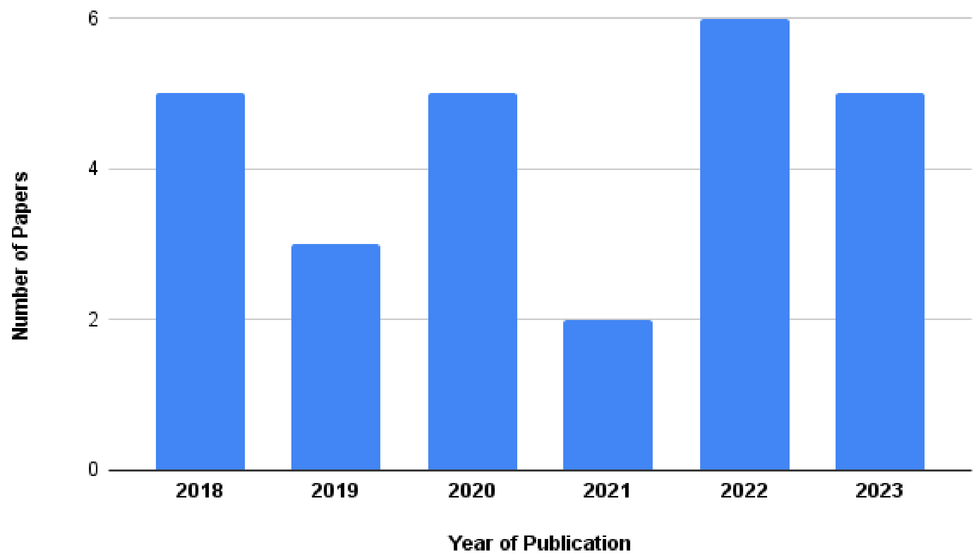


Figure 3. The Chronological Overview of the Selected Papers

Figure 3 shows our selection of 26 peer reviewed papers published up to the mid of April 2023. We selected 5 articles from the year 2018 and a total of 6 articles from year 2022, while we identified 2 paper from the year 2021.

3. 2 Microservices Composition Approaches

Microservice composition is a process that is used to connect numerous microservices in an application or scenario, to collaborate in a business application, and achieve the business requirements of the system. As previously mentioned, microservices mainly use two different composition approaches: orchestration and choreography (Asri et al., 2022; Aydin & Çebi, 2022; Baškarada et al., 2018; Valderas et al., 2020). In communicating between services, orchestration has a single hard point dependency, whereas choreography has no central dependency point by nature (Bigheti et al., 2019; Valderas et al., 2020). Also, in an orchestration-based approach, the orchestrator is responsible for all communication within the system. All requests and responses are managed by the orchestrator (i.e., central controller).

In a microservice choreography, each microservice uses an asynchronous approach, performing its actions independently and not requiring any instructions. In a microservice orchestration, the orchestrator, which is a central microservice controller, handles all elements and interactions. Both choreography and orchestration composition mechanisms have boundaries, but these constraints can be overcome by using a combined single application based on the business requirements a solution called the hybrid approach (Valderas et al., 2020). Indeed, we have found some studies that use the hybrid approach, a combination of choreography and orchestration (Kazanaviˇcius & Maˇzeika, 2023; Megargel et al., 2021; Valderas et al., 2020). Although a hybrid method may prove better for microservices composition, it is hard to design and implement (Singhal et al., 2019a) as developers need to design both choreography and orchestration composition approaches in the same work flow to satisfy the business requirements.

3.3 Data Collection for RQ1 and RQ2

Our data extraction consisted of finding information from the relevant papers and writing a summary for each paper. To accomplish this, we use manual extraction methods. The extraction procedure has two steps. In the first step, we short-list studies, publication types, research styles, and research designs. **Table 3** shows the publications selected from which data was extracted for this literature review.

Table 3: List of Publications Selected for Microservices Composition

Paper ID	Authors	Paper Title	Publication Type	Research Style	Research Design	Year
P1	Shuo, Z., et al.	Research on the application of service choreography in the intelligent customer service system	Conference Paper	Exploratory	Mixed	2023
P2	L. de Castro, et al.	Relating Edge Computing and Microservices by means of Architecture Approaches and Features, Orchestration, Choreography, and Offloading: A Systematic Literature Review.	Journal Article	Exploratory	Inductive	2023
P3	Kazanaviˇcius & Maˇzeika	Evaluation of microservice communication while decomposing monoliths	Journal Article	Descriptive	Deductive	2023
P4	Zhang, X., et al.	An API Framework for Flexible and Lightweight Microservice Composition	Conference Paper	Descriptive	Mixed	2023

P5	Represa, J. G., et al.	Investigation of Microservice-Based Workflow Management Solutions for Industrial Automation.	Journal Article	Exploratory	Deductive	2023
P6	Koyya, K. M., et al.	A Survey of Saga Frameworks for Distributed Transactions in Event-driven Microservices.	Conference Paper	Exploratory	Inductive	2022
P7	Asri, S. A., et al.	Implementation of Asynchronous Microservices Architecture on Smart Village Application	Journal Article	Descriptive	Mixed	2022
P8	Aydin, S., et al.	Comparison of Choreography vs Orchestration Based Saga in Microservices.	Conference Paper	Exploratory	Deductive	2022
P9	Volynsky, E., et al.	Architect: A Framework for the Migration to Microservices	Conference Paper	Exploratory	Inductive	2022
P10	Nadeem, A., et al.	A Case for Microservices Orchestration Using Workflow Engines.	Conference Paper	Descriptive	Deductive	2022
P11	Koschel, A., et al.	Towards the Implementation of Workflows in a Microservices Architecture for Insurance Companies-The Coexistence of Orchestration and Choreography.	Conference Paper	Exploratory	Inductive	2022
P12	Nebel, A., et al.	MicroIP: A general-purpose microservice-based integration platform	Conference Paper	Descriptive	Mixed	2021
P13	Megargel, A. et al.	Microservices Orchestration vs. Choreography: A Decision Framework.	Conference Paper	Exploratory	Inductive	2021
P14	Dai, F., et al.	A Choreography Analysis Approach for Microservice Composition in Cyber-Physical-Social Systems.	Journal Article	Descriptive	Deductive	2020

P15	Cao, J.	Design on Deployment of Microservices on Container-based Cloud Platform.	Conference Paper	Explanatory	Inductive	2020
P16	Valderas, P. et al.	A microservice composition approach based on the choreography of BPMN fragments.	Journal Article	Explanatory	Mixed	2020
P17	Valderas, P. et al.	Supporting a Hybrid Composition of Microservices. The EUCalipTool Platform	Journal Article	Explanatory	Mixed	2020
P18	Pontarolli, R. P., et al.	Microservice Orchestration for Process Control in Industry 4.0.	Conference Paper	Explanatory	Deductive	2020
P19	Singhal, N. et al.	Selection Mechanism of Micro-Services Orchestration Vs. Choreography	Journal Article	Exploratory	Deductive	2019
P20	Müller, M.	Consistency and Autonomy in the Microservice Architecture	Journal Article	Exploratory	Inductive	2019
P21	Singhal et al.	Efficient hybrid research for QoS-aware microservice composition	Journal Article	Exploratory	Mixed	2019
P22	Rudrabhatla, C. K.	Comparison of Event Choreography and Orchestration Techniques in Microservice Architecture	Journal Article	Descriptive	Deductive	2018
P23	Baškarad, S., et al.	Architecting Microservices: Practical Opportunities and Challenges.	Journal Article	Exploratory	N/A	2018
P24	Monteir, D. et al.	Beethoven: An Event-Driven Lightweight Platform for Microservice Orchestration	Conference Paper	Descriptive	Mixed	2018
P25	Scattone et al.	A Microservices Architecture for Distributed Complex Event Processing in Smart Cities	Workshop	Exploratory	Deductive	2018
P26	Bogner, J. et al.	Analyzing the relevance of SOA for microservice-based systems	Workshop	Exploratory	Inductive	2018

In the second step, data was collected mainly for the first research question (**RQ1**), and a summary of the primary document proposal is presented. The data extracted identified composition types in microservice architectures which directly contribute to the evidence for the research question. In this procedure, we identified approaches, key features, advantages, and critiques and manually applied them from the selected papers, based on our research question (**RQ1**), to fill **Table 4**.

Table 4: Microservices Composition Styles Advantages and Limitations

RQ1. What are the advantages and disadvantages of microservices orchestration, choreography, and hybrid composition?				
Paper ID	Composition approach	Key Features and Advantages	Criticisms/ Limitations	Tools, Models, Languages
P1	Choreography Orchestration	Choreography - service choreography technology allows for flexible adaptation to various situations and requirements. The service choreography engine can also be seen as a type of workflow engine. Orchestration - the condition expression is the lifeline of service orchestration.	Choreography - The service choreography process needs to define the relevant service information for each process node. Orchestration - visual orchestration platform uses a modeler for customizing and relies on logical connections between nodes.	Zeebe, Conductor, Broker, Narcos, BPMN Models, Docker, Kubernetes, Camunda
P2	Choreography Orchestration	Choreography - in the choreographic mode, there is no central orchestrator. Instead, the coordination of microservices is distributed among all the individual microservices themselves. Orchestration - this mode involves having an orchestrator that acts as a conductor for the microservices. Orchestrator can collaborate more effectively with the API Gateway during the service composition process.	Choreography - the microservices function as a distributed architecture and communicate with each other through a message bus, it carries a higher complexity. Orchestration - When dealing with different infrastructures and environments, an orchestration approach can become too complex to manage effectively. Hybrid - in a hybrid composition, both orchestration	Container, Virtual Machines, BPMN Models

		Hybrid microservice composition could be supported and invoked by switching the microservice architecture topology.	and choreography work together and hard to manage.	
P3	Choreography Orchestration Hybrid	Choreography operates without a centralized element, allowing services to collaborate directly with each other. Orchestration - Service orchestration functions as a centralized business process, coordinating activities across various services and merging their results. The hybrid pattern merges the service registry and API gateway, and it substitutes the API gateway with the message bus.	In the hybrid approach, there is a strong coupling between services and the message bus. Instead of direct communication between microservices, they interact through the message bus, which serves as a mediator for their communication.	BPMN Models, C Sharp, .Net Core, Visual Studio 2022 IDE
P4	Choreography Orchestration	Choreography - in the service choreography pattern can operate with less resource utilization, service orchestration allows for service reuse in multiple contexts. Orchestration - Service orchestration relies on a central controller to manage the execution of APIs in orchestrated services.	Choreography is not a flexible microservice composition pattern and requires modification of microservices. Choreography does not support service reuse in multiple contexts. Orchestration may not be as efficient in terms of execution performance due to data relay overhead.	BPMN Models, Data Reception Algorithms
P5	Choreography Orchestration	Choreography - control is passed to the next system executing each step. Orchestration - the steps of a workflow are controlled by a central system (orchestration).	Choreography - to ensure interoperability of the modeling languages at the program code level, it is essential to implement the corresponding libraries.	BPMN Models, Petri Net, Web Services Description Language (WSDL)

P6	Choreography Orchestration	Choreography is more suitable when there are fewer participating microservices. For other situations, orchestration is the preferred approach.	Choreography is not as effective when there are more participating microservices.	BPMN Models
P7	Choreography Orchestration	In microservice choreography, each microservice operates autonomously without needing specific instructions. Microservice orchestrations are more like centralized services, as they call one service and wait for a response before calling the next one.	In the orchestration approach, there is an issue of interdependence between services.	BPMN Models
P8	Choreography Orchestration	Choreography event-based architecture is suitable for simple procedures with less participants and no coordinating require. Orchestration - useful for complex work processes, including numerous members or new added.	Because of decentralized nature, choreography might be more complicated to comprehend than orchestration. Numerous services can pick up a message simultaneously.	BPMN Models
P9	Choreography Orchestration	Orchestration - having all the logic in one place in orchestration makes the process easier to modify and trace.	In choreography, there is a risk of introducing cyclic dependencies that might occur in event choreography.	UML Designs BPMN Models
P10	Choreography Orchestration	Debugging is easier and faster in Orchestration. Choreography offer better independency of services.	Orchestration has hard dependency point. For larger system add/remove services is complex in choreography.	BPMN Models
P11	Choreography Orchestration	Choreography - is often combined with other patterns, for example, event-driven architecture and there exists no orchestrator. Orchestration - for workflow implementation involves coordinating the various process steps. These steps may include business services, technical	Excessive choreography can lead to chaos, while an abundance of orchestration can result in a monolithic system.	BPMN Models RaMicsV

		services, or even user tasks. The coordination is managed by a coordination unit or orchestrator.		
P12		Choreography offers the advantage of being decentralized and requiring less operational effort since there is no need for an orchestrator. Orchestration is better for complex event, main reason for this is that in complex scenarios, maintainability is considerably improved when coordination is managed from a centralized place.	Choreography is suitable for simple scenarios. Orchestration is better suited for complex interactions.	UML Designs
P13	Choreography Orchestration Hybrid	Choreography leads to solutions with less coupling and less chattiness. Orchestration leads to solutions with better process flow visibility.	Require downtime during deployment in order to avoid interruption (orchestration); 'hybrid' collaboration pattern choreography with a process engine.	UML Designs
P14	Choreography	Choreography-driven microservice composition approach is considered to provide a better way to integrate these Cyber-Physical-Social System (CPSS).	the complexity of choreography analysis grows rapidly with the number of Cyber-Physical-Social System (CPSS) services participating	UML Designs
P15	Choreography Orchestration	The process of organizing multiple containers in this way is referred to as container choreography. In large-scale application systems, one-click deployment can be achieved by using suitable orchestration tools.	NA	Docker
P16	Choreography Orchestration	Choreography (decentralized) is more appropriate for microservices. Orchestration can be a drawback to achieve the decoupling	The complexity of choreographic compositions has forced many companies to propose Orchestration.	BPMN Models
P17	Choreography Orchestration	Choreography - obtain a	Orchestration - centralized by nature, reducing the degree of	EUCalipTool Platform,

	Hybrid	major level of independence, flexible and decoupling by nature; Orchestration - global vision and management offered. Hybrid – takes advantages and combines choreography and orchestration styles.	decoupling, the adoption of this solution difficult	BPMN Models
P18	Choreography Orchestration	Orchestration - there is a central control and coordinates the call of other services in request/response way. Choreography - the call of service execution is predetermined.	Orchestration required central control whereas choreography offers predetermined services	BPMN Models, UML Designs
P19	Orchestration Choreography	Orchestration follows a request/response messages, provides an effective way for coordinate. Choreography - supports faster processing as services can be executed in a parallel fashion. Easier to add and update services, control is distributed several.	Orchestration- Coupling of the services together creating dependencies. Single point control and if it goes down, all processing stops. Choreography - async programming is often a significant mind shift for developers.	BPEL, BPMN Models
P20	Choreography Orchestration	Choreography is expected to be faster than orchestration since the logic and work are distributed among the services, making them more autonomous from each other. Orchestration offers the advantage of centralizing all the logic in one place, making implementation, testing, debugging, and changes easier. It also facilitates the detection of service failures and enables specific recovery actions to be initiated.	Choreography- the distribution of logic in choreography makes failure detection and recovery more complex. It is difficult in Choreography to determine the point in time when all services have received and successfully updated their data. Orchestration - a single orchestrator can become a bottleneck for the system. Implementing and adjusting the orchestrator to accommodate changes is an additional task.	N/A
P21	Choreography Orchestration Hybrid	Choreography - decentralized approach for service composition	Orchestration- security is a concern in this centralized approach.	Java eclipse, Eclipse IDE

		Orchestration - single centralized coordinated, good for small application scenario, maintenance is also easy it is good for localization, flexibility. Hybrid - better performances and an optimal solution	Choreography- highly specialized service component, it is not within the single control. Hybrid - hard to design, implement, and maintenance.	
P22	Choreography Orchestration	Event choreography is much faster in performance. Orchestration is useful when the transaction scenarios are complex	Choreography becomes very complex to code and handle if there are multiple events; Orchestration is slow.	Apache Active MQ, BPMN Models
P23	Choreography Orchestration	Choreography distributes the coordination responsibilities to individual microservices.	Orchestration offers central control for microservices.	BPMN Models
P24	Orchestration	Event processor can send or receive events and commands to or from another event processor using the event channel.	event processor component is responsible for processing a specific type of event and notifying	Beethoven
P25	Choreography Orchestration	Choreography offers low coupling and greater horizontal scalability, increasing its resilience; Orchestration - a failure in the core part can lead to the systems breakdown	N/A	N/A
P26	Choreography Orchestration	Choreography - potentially higher degree of technological heterogeneity.	Orchestration - low degree of standardization, centralization of control and management	N/A

3.4 Synthesis

We use the extracted data of Section 3.3 for analysis to discuss and answer the first research question (**RQ1:** What are the advantages and disadvantages of microservices orchestration, choreography, and hybrid composition?). To answer this question, we collect data from columns 2, 3, and 4 of **Table 4**. From column 2 the composition types are identified; column 3 identifies advantages of composition styles in each paper, and column 4 lists critiquing points about microservices composition techniques. The key features and advantages, as well as the disadvantages of these composition techniques, are compared with composition tools, models, and designs we have found in select studies.

Figure 4. Mind Map for Microservices Compositions

Figure 4 shows the mind map for extracted data and relevant sub concepts from the literature. To answer the second research question (**RQ2**: In what situation should we use the hybrid techniques for microservices composition?), we use **Table 4**, the collected data. Also compared are the how and why these approaches can be valuable for microservices-based software development. Finally, the data previously collected from the primary papers are summarized.

3.5 Data Results

In this section we present the results from our research process, broken down as per the research questions, and conclude with a discussion about our key findings.

3.5.1 Advantages and Limitation of Microservices Orchestration, Choreography, and Hybrid (RQ1):

***Choreography* Approach**:

A choreography-based composition for microservices supports faster processing than orchestration, as services can be executed in parallel instead of sequentially. Choreography-driven microservice composition is better than orchestration for integrating and analyzing the Cyber-Physical-Social System (CPSS) (Dai et al., 2020; Represa et al., 2023; Stutz et al., 2020). Such a choreography approach composes services in a more loosely coupled and manner flexible than orchestration (Shuo et al., 2023). Choreography-based style control is distributed, and it is easily adaptable for microservices. Moreover, with a choreography style, it is easy to introduce additional services since listen to the next event and respond accordingly. Its foremost advantage is that it is not tightly coupled, and microservices remain flexible to call each other (Scattone & Braghetto, 2018; Sorgalla et al., 2018). To achieve better decoupling, it is suggested to utilize choreography for microservices (Ortiz et al., 2020).

By contrast, adding or removing services is complex in choreography if the number of CPSS devices increases for a large system (Aydin & Çebi, 2022; Dai et al., 2020; Nadeem & Malik, 2022). One of the challenges with choreography, therefore, is that there is no central service to identify the tasks to ensure that all the services performed their task successfully. Choreography, in a decentralized control, is liable for defining protocols, how services should connect with each other, and exchanging messages between them (Cerny, Donahoo, & Trnka, 2018; de Castro & Rigo, 2023; Kazanaviˇcius & Maˇzeika, 2023; Represa et al., 2023; Zhang et al., 2023).

***Orchestration* Approach:**

In an orchestration-based composition, debugging is generally easier and faster compared to choreography (Nadeem & Malik, 2022). An orchestration approach is suitable for complex system design scenarios (Rudrabhatla, 2018). The main benefit of orchestration is that it provides an effective way to coordinate the flow of an application; as such, this builds microservices-based applications which are more convenient, i.e., easier to coordinate with each other. Microservices orchestration style performs based on central control and calls other microservices by using a request/response mechanism that confirms acknowledgement of service received/delivered (Monteiro et al., 2018; Pontarolli et al., 2020; Represa et al., 2023; Zhang et al., 2023).

On the other hand, orchestration performs communication in a request/response-centric way and, by its very nature, adds considerable latency, in addition to its dependency on the multiple services. Orchestration has a hard dependency point that is a controller and acts as a collaborator (Bogner et al., 2018). The decoupling features of microservices are harder to achieve in orchestration compared to choreography and require downtime during deployment (Megargel et al., 2021; Scattone & Braghetto, 2018; Valderas et al., 2020). Yet, some studies emphasize that orchestration, for developers, is more feasibly doable in implementation within their systems because it is less complex than choreography to develop and manage systems that use microservice architecture (Rudrabhatla, 2018).

***Hybrid* Approach:**

A hybrid approach exhibits better performance and generates optimal solutions compared to choreography and orchestration approaches (Kazanaviˇcius & Maˇzeika, 2023; Singhal et al., 2019a; Valderas et al., 2020).

3.5.2 Hybrid Techniques for Microservices Composition (RQ2):

One way to use a hybrid communication mechanism is to compose the entire system such that cooperation is based on an event-choreography. Thus, in a hybrid approach, the orchestrator connects to other services and then integrates with the rest of the applications using a choreography. In this way, the hybrid style offers benefits from the choreography approach and keeps the deployment process easy by using controllers for specific microservices in a composition (de Castro & Rigo, 2023; Valderas et al., 2020).

To achieve a better performance, the hybrid approach combines the choreography and orchestration methods to deal with different services in microservice composition (de Castro & Rigo, 2023; Koyya & Muthukumar, 2022; Singhal et al., 2019a; Valderas et al., 2020). We can use a hybrid modeling approach for microservices when scenarios are very complex but need to incorporate and manage the low coupling principle of microservices to achieve a control and collaboration confirmation within the microservice architecture. One should consider a hybrid mechanism for composition when better visibility or adding or removing services without downtime in a process are desired, and/or there are security concerns within a complex scenario. The strengths of choreography and orchestration can be combined to create a hybrid approach for microservices composition. In addition, we need to decide depending on the complexity of the scenario when to use choreography and when to use orchestration in the same workflow to achieve a hybrid approach's best performance.

3.6 Discussion

From the results, it can be observed that microservices architectural designs and composition approaches research is still under development. We discuss which composition approach is better and why that is, based on selected studies which present both their respective advantages and disadvantages in the data results section. The benefits and limitations of choreography and orchestration styles of microservices are subject to debate. We cannot simply say that one microservices composition approach is better than the other without knowing in detail why and for what purpose microservices architectures are implemented. It mainly depends on the purpose of the application one is building with microservices, how many applications or systems one wants to develop, and how frequently one needs to summon microservices.

Most previous studies are generally selective in choosing either choreography or orchestration composition styles for microservices. It is apparent that less attention has been given to the recently proposed hybrid style in prior research. From published studies, it is not clear why and when to use the hybrid approach. Here, we identify this research gap and the need for additional research to understand the complexity of microservices composition.

Results of **RQ1** see **Table 4**, identify three composition types for the microservices architecture; approaches, benefits, critiques, and the tools they use for the composition, are also presented. From the literature, it is evident that handling multiple actions without a central orchestration is difficult. In general, a choreography-based microservice composition style is suggested when fewer microservices are required for application development and fewer request/actions need to take place, that is, when the system is not complex to handle (Singhal et al., 2019b). However, the orchestration composition approach is recommended when application complexity is higher and it is better to have a central orchestrator, coordinator event, or microservice providing a single point control to better maintain the services.

We identified many studies that use BPMN tools for microservice composition, as well as UML-based tools such as Zeebe, Visual Studio 2022 IDE, Camunda, EUCalipTool Platform, Eclipse IDE, Beethoven, Broker, and Narcos. To answer **RQ2** see **Table 4**, we have limited information from the literature. Most papers used in this study are identified as exploratory research styles and deductive research designs. Moreover, we identified four explanatory journal and conference papers, two of them being seminal papers. This information is helpful for future researchers and software developers in finding proper solutions for designing microservices-based software.

3.7 Summary

The literature identifies 3 main composition styles: choreography, orchestration, and hybrid. The hybrid composition architecture is only a recent proposal, and we have found only four papers relevant to this approach. The choreography-based microservice approach is recommended when the microservices required to build a system are few or for a small application, that is, having to execute fewer actions; whereas the orchestration-based composition method is recommended for better control in composition, when more services are best accomplished via a coordinator (since a central microservices orchestrator performs management of those services easily). The choreography-based microservice composition approach performs better than orchestration when the scenarios are simple and fewer services are needed to coordinate. For composition, the orchestration-based composition style is recommended for better control of services for composition. However, a hybrid approach is suggested when there are both a need to manage higher complex scenarios and to maintain loosely coupled principles for microservices. These findings are beneficial for researchers who need an easy identification of any related research in microservices composition.

CHAPTER 4. BPMN Models of Microservices

This chapter explains how to create microservices using BPMN and how to deploy and monitor them automatically in the cloud. We start by showing how to design BPMN models using Camunda Modeler. The conceptual model illustrates the microservices architecture using BPMN activities, gateways, and messages. We use examples from the e-commerce domain to demonstrate how to design workflows using different approaches such as choreography, orchestration, and hybrid composition. These workflows show the steps involved in browsing an e-commerce website and placing an order, up to the delivery of the order to the customer.

4.1 Creation of BPMN Models Using Camunda Modeler

This section focuses on the design of BPMN models, which are the main components of our research. The BPMN modeling procedures and tools allow us to design and execute our models with proper logic based on e-commerce scenarios. We create our microservices using the Camunda Modeler desktop application. This tool is used for designing BPMN workflows and deploying them to the cloud. We use the open-source version of the Camunda Modeler, which has all the features of the Zeebe Modeler. To simplify the deployment process, we use "user task" instead of "service task". The Camunda Modeler allows us to produce executable XML codes for each BPMN model, and it provides options for designing, connecting, automating, and monitoring business processes. We can deploy our workflows to the cloud-native BPMN workflow engine using the endpoint configuration (REST Endpoint) of the Camunda Platform, which ensures high speed, scalability, and resilience in modeling and deployment process (“The Camunda Platform 7 Manual,” n.d.).

4.1.2 Why have we opted for the Camunda Platform?

During our literature review, requirement analysis, and system design phase, we explored various tools, including Zeebe, EuCalipTool, Camunda, EdrawMax, Eclipse Modeling Tools, Microsoft Visio, AjiL, and Visual Paradigm. Unfortunately, most of these tools could not generate XML codes, lacked support for cloud deployment, and were not scalable, as seen in EuCalipTool, Microsoft Visio, and AjiL. As a result, we narrowed our choices down to Zeebe and Camunda. However, due to the merger between Zeebe and Camunda in late 2020, the older version of Zeebe is no longer accessible. Considering our preference for open-source solutions, we have ultimately decided to use the Camunda tools. The main advantage of Camunda Tools is a dedicated graphics editor that is user-friendly even for non-developers (“The Camunda Platform 7 Manual,” n.d.).

4.2 BPMN Elements in Camunda Modeler

We followed certain guidelines called Camunda BPMN coverage components to select specific elements for our study (“BPMN coverage,” n.d.). When making BPMN models to define our microservices, we only used the elements supported by the Camunda Platform, which include start event, user tasks, XOR gateways, text annotation, group, data flow objects, message events, and end event. Our objective is to build models centered around tasks rather than events. This decision prompted us to make the workflows elements and activities simpler, avoiding intricate deployment challenges. To maintain this approach, we intentionally steered clear of incorporating message events and timers. It is important to note that the Camunda Platform does not impose any of these restrictions.

In BPMN modeling, a user task is used to represent work that needs to be done by a human. It has a task ID number and a general name. When a process instance reaches a user task, a new job similar to a service task is created, and the process instance stops until the job is finished (“BPMN coverage,” n.d.).

An exclusive gateway, also known as XOR gateway, is a tool used in BPMN structures to make decisions based on data received during the process. It functions like a decision box with conditions. If there are multiple outgoing sequence flows or message flows for an XOR gateway, all but one must have a condition expression to specify which flow will be taken and what decision will be made (yes or no). If none of the conditions are met, the gateway follows the default flow, otherwise it follows the conditional flow (“Exclusive gateway,” n.d.). Furthermore, in case the gateway lacks a default flow, it results in the creation of an incident in business process modeling. An XOR gateway has the capability to merge various incoming sequence flows and enhance the clarity of BPMN workflows (“Exclusive gateway,” n.d.).

4.3 e-Commerce Scenarios for BPMN Models

E-Commerce applications are delivering critical new features and a wide range of business services to their customers and, considering these functionalities, we have categorized our model scenarios into three types: small, mid, and end-to-end complex scenarios. Each workflow consists of multiple e-commerce services to address the users' needs. Likewise, each workflow explains a specific business logic for e-commerce applications such as user authentication, payment process, shipment, or delivery-related services.

These scenarios also assist researchers and developers in determining which services they need to develop an e-commerce system from scratch using microservices. In our previous work, we designed and evaluated the complexity of small, mid, and end-to-end scenarios (Ali et al., 2022). Throughout this research project, we tried out different tools like Camunda, Zeebe, EuCalipTool, AjiL, and Visual Paradigm for microservices composition. At first, we focused on designing choreography and orchestration. We were able to understand how complex it can be to create models for microservice choreography and orchestration. The results of our work were published in a paper while working on this thesis (Ali et al., 2022). Moreover, we will use these categories to create complex scenarios for e-commerce applications and evaluate the coupling, cohesion, and scalability of our end-to-end big-sized scenarios. We follow five steps to choose our e-commerce scenarios:

a) Define our workflows scenarios
b) Start our design processes based on the steps usually required by e-commerce sites, from the browsing of a site to the delivery of customers' desired products, to accepting customer comments
c) We begin with small-sized models, which typically include one module with multiple services.
d) We then merge small workflows to create mid-sized business processes.
e) Finally, we combine small and mid-sized workflows to create complex end-to-end big e-commerce scenarios.

4.4 Designing Choreography-based BPMN Models

In choreography-based modeling, each service performs a specific task in the BPMN workflow and calls the next service to continue the process. We do not rely on a central control, such as an orchestrator, in each e-commerce scenario module. In choreography-based BPMN modeling, services function independently without relying on one another. We create each workflow in small, mid, and end-to-end complex scenarios. To facilitate this choreography-based business process modeling, we use Camunda BPMN 2.0 Modeler and Camunda Cockpit to monitor incidents, run instances, and define jobs ("The Camunda Platform 7 Manual," n.d.).

4.5 Designing Orchestration-based BPMN Models

To create orchestration-based workflows, we use an orchestrator in each microservice to act as a communication controller. The orchestrator uses a call-and-response messaging style to communicate with the next service, waiting for a response before processing the next request. For example, in **Figure 6**, we use Checkout Orchestrator for the checkout process microservices BPMN-based model. In an end-to-end e-commerce scenario, we will add multiple orchestrators to make our models more readable and easier to understand. Although we have used a single orchestrator in our previous study, it makes the model more complex to understand and difficult to manage (Ali et al., 2022).

4.6 Designing Hybrid BPMN Models

When software projects adopt MSA, there are several concerns that developers face, including designing, developing, and deploying microservices that meet business requirements. This can be a challenging decision-making process, as developers need to consider logical architecture and modeling methods (Santos, Rodrigues, Ferreira, & Machado, 2019). In this chapter, we propose a new approach for microservices composition that can help developers make informed decisions when dealing with complex scenarios. Our approach combines choreography and orchestration styles into a single workflow, creating a hybrid approach that is suitable for mid-sized and end-to-end scenarios. When designing a hybrid BPMN-based model of microservices, at least one choreography-based model and one orchestration-based model are required. **Figure 16** shows an example of a hybrid model with one orchestration (Checkout Process) and one choreography (User Authentication). Our aim in designing hybrid-based BPMN models for microservices is to achieve better control over communication and services and reduce dependence on other services.

4.7 Small-sized BPMN models

In this research, we have created small BPMN models for single microservices such as shipment, checkout process, user authentication, and payment process. We have designed these workflows using either choreography-based or orchestration-based BPMN models.

4.7.1 Checkout Process

This service enhances the user's online shopping experience by providing a feature that allows them to add items to their cart when they start browsing. If the user wants to make a purchase, they can click on the cart and proceed to checkout. **Figure 5** shows the choreography model for the checkout process, which includes one start event and three user tasks.

Figure 5. Choreography Model for Checkout Process

The checkout process that we designed in **Figure 6** is based on orchestration-based composition. We added an extra user task called the "checkout orchestrator," which acts as a controller in this model. Moreover, we included an XOR gateway to differentiate between users who have "browsed" the website and those who are "new to the site." The condition type expression used is "added".

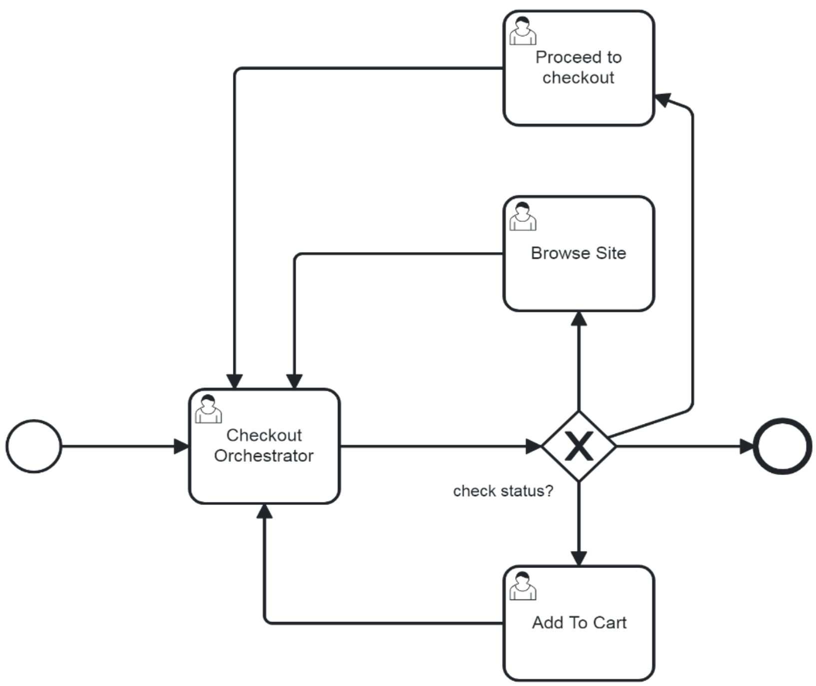


Figure 6. Orchestration Model for Checkout Process

4.7.2 User Authentication

In the checkout process that we designed in **Figure 6** users need to log in to access their accounts. To do this, they enter their userID, email, or phone number to verify their identity. If they have already registered, they are redirected to a login page where they enter their login credentials. If they are new to the website, they can either continue as a guest user or sign up to create an account. A choreography-based model for user authentication was designed in **Figure 7** using Camunda Modeler. This model consists of three user tasks and four XOR gateways, which determine the workflow of user authentication. The XOR gateways have two possible paths: one if the condition is true (yes) and the other if it is false (no). The process flow follows the path based on the condition at each XOR gateway. This model simplifies and streamlines the user authentication process.

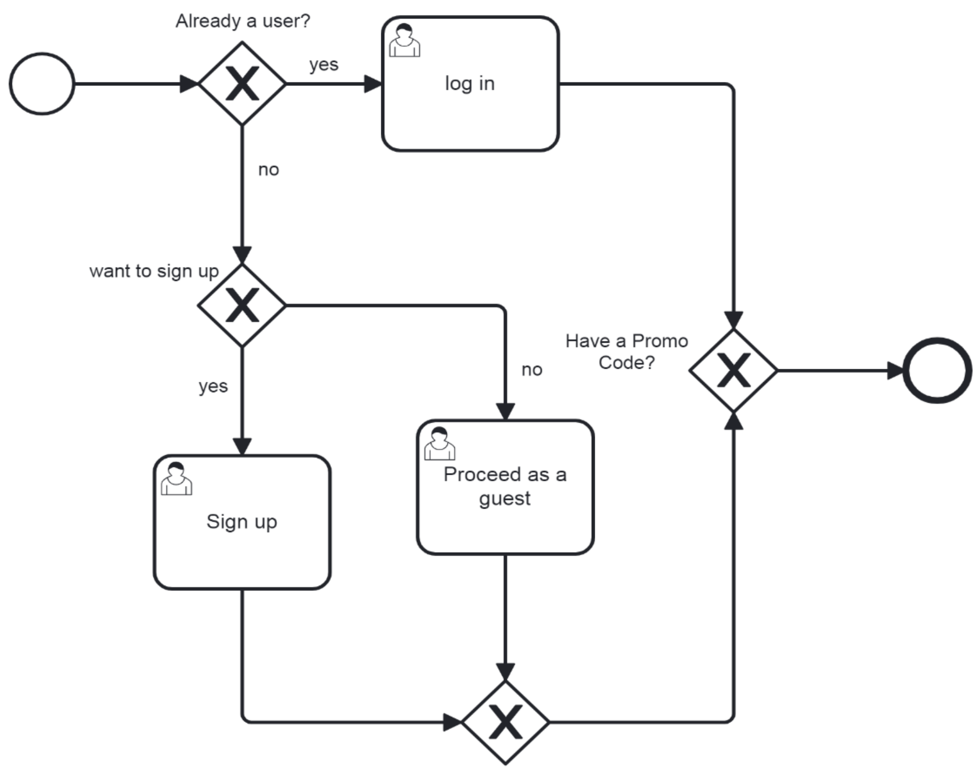


Figure 7. Choreography Model for User Authentication

Figure 8 shows how we use an orchestration-based approach for user authentication in an e-commerce system. There is one exclusive gateway and four user tasks, and the authentication orchestrator acts as the controller for the workflow, facilitating communication between the tasks.

Figure 8 is an updated version of the one defined in **Figure 7**, representing the BPMN model of the authentication orchestrator.

In this model, an XOR gateway determines whether the user is already registered on the site, wishes to sign up, or wants to proceed as a guest. The authentication orchestrator receives these decisions and coordinates the subsequent services in the workflow.

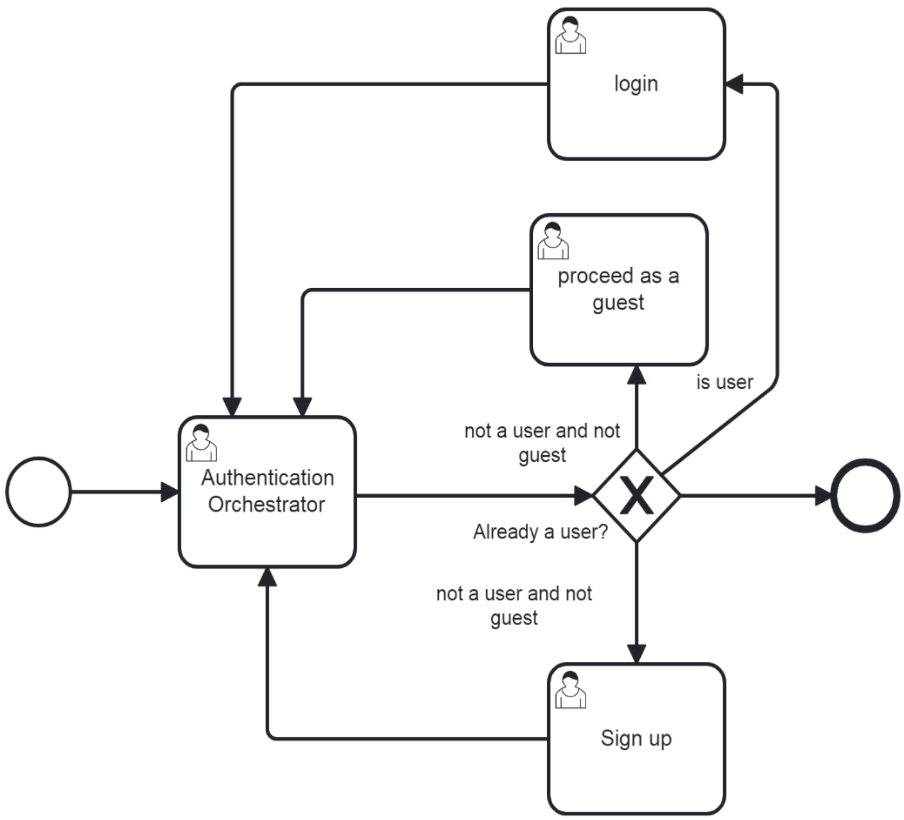


Figure 8. Orchestration Model for User Authentication

4.7.3 Shipment

This model has two shipping methods: shipping to an address or picking up at the store. **Figure 9** shows the choreography-based composition for a shipment workflow. When a customer chooses to pick up their order at a store, the system shows them the nearest locations. If they choose to have the order shipped, there are two shipping cost options. Free shipping is available for orders above a certain amount. This requires adding two expression condition types in our BPMN model: "${orderValue >= 100}" connected to "ship for free", and "${orderValue < 100}" connected to the user task, "shipping fees". In this scenario we use six sub-processes and three XOR gateways in the shipment process design.

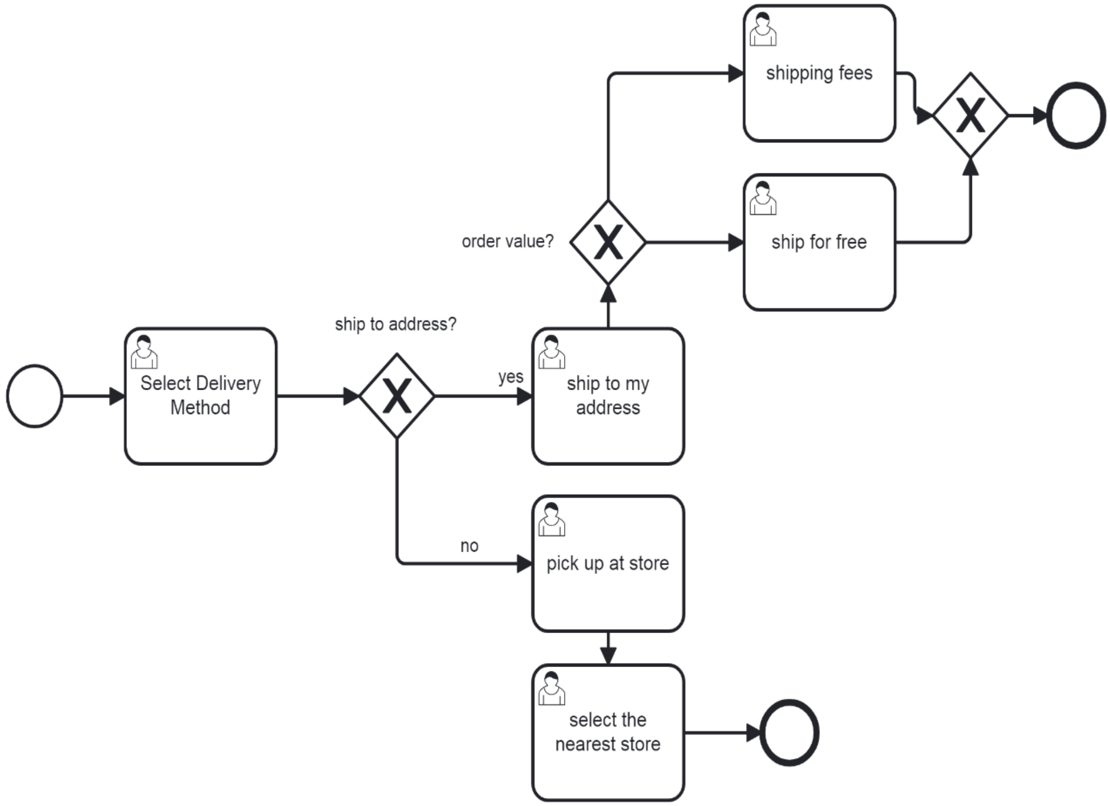


Figure 9. Choreography Model for Shipment Process

Figure 10 presents the shipment workflow as an orchestration. The delivery orchestrator is the controller of this process. This model is similar to the one shown in **Figure 9**, with seven tasks and one exclusive gateway. We defined True/False values and variables for the XOR gateway. The delivery orchestrator communicates between the tasks using request/response messages.

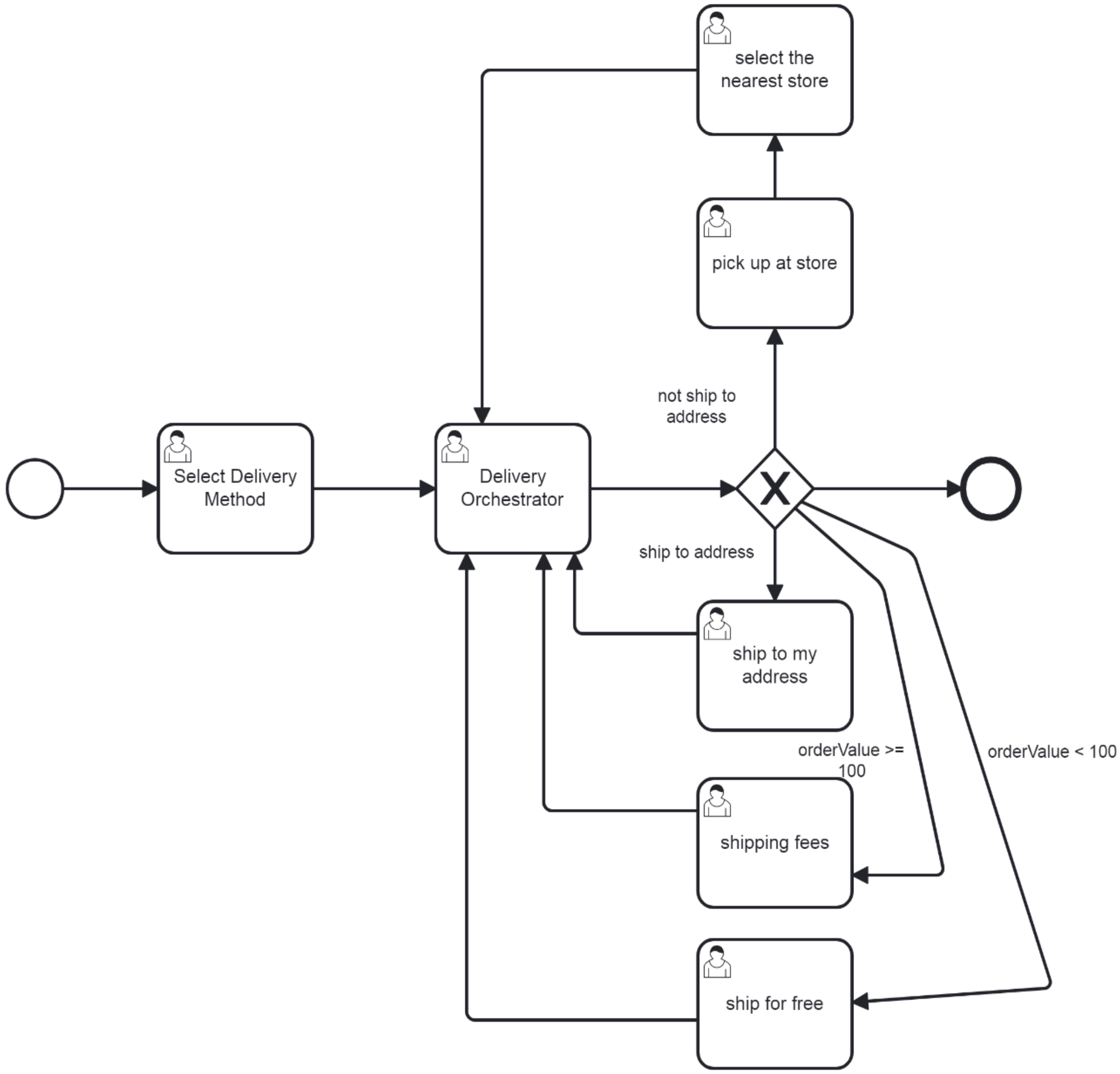


Figure 10. Orchestration Model for Shipment

4.7.4 Payment Process

The payment service lets customers pay for their purchases online using their bank details, such as debit or credit card, e-transfer, cheque, or digital wallet. We created a BPMN model using Camunda Modeler, shown in **Figure 11**, to manage the payment process. The model uses two user tasks and an XOR gateway to verify payment success. If payment is successful, the process moves on to the next service, "Payment Received." However, if the payment fails, the process ends.

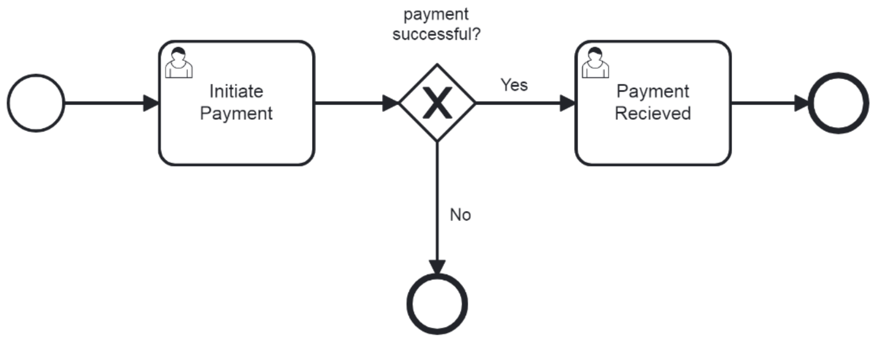


Figure 11. Choreography Model for Payment Process

4.7.5 Order Process

Once payment is made, the e-commerce website prepares the order for either pickup or delivery through a courier service. The process is shown in **Figure 12** using a BPMN model that follows the orchestration approach. The order orchestrator manages communication during the process. The workflow includes six user tasks, an XOR gateway, and an end event. The decision box allows users to track their orders and provide feedback about the business.

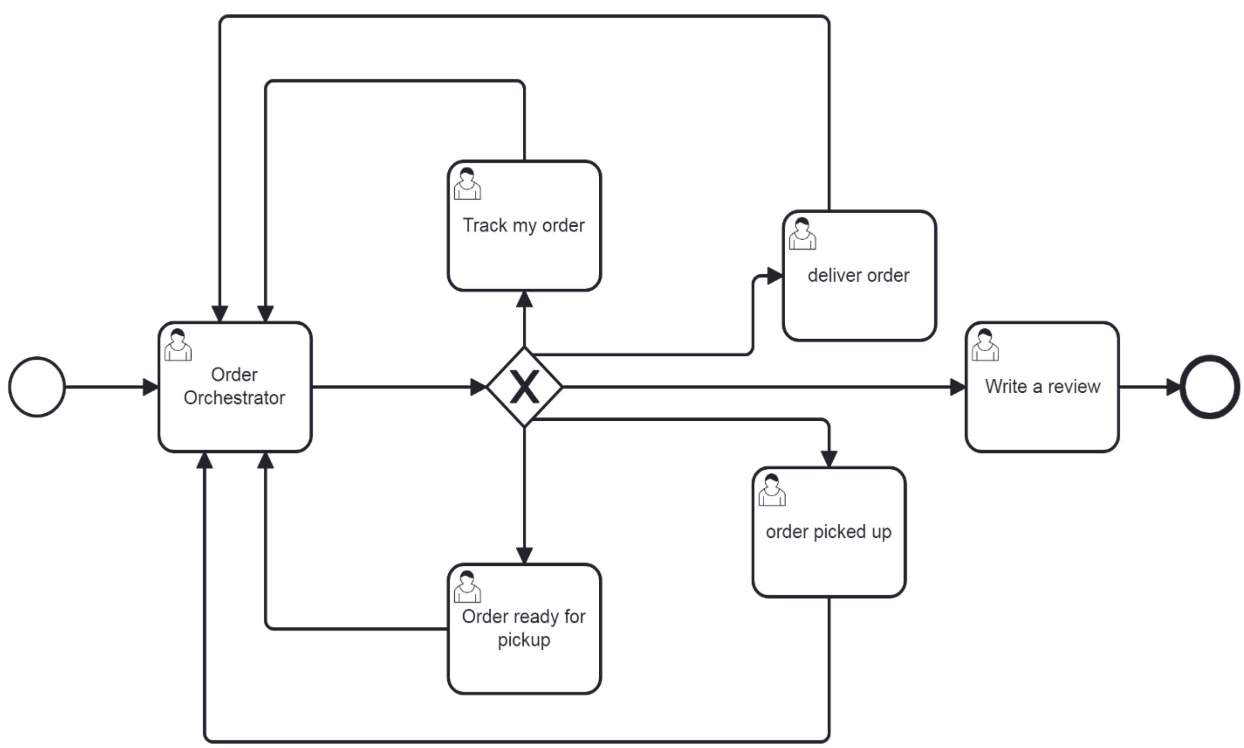


Figure 12. Orchestration Model for Order Process

4.8 Mid-sized BPMN Models

In this section, we expand on the models we designed in Section 4.7 by creating new ones that involve multiple services in each workflow. To do so, we are using Camunda Modeler to design, deploy, and carry out these models in the cloud. For instance, in **Figure 13**, we merge the checkout process with the user authentication service to create a new model based on choreography. Additionally, **Figure 14** shows an orchestration-based workflow for these services, while **Figure 15** presents a mid-sized BPMN model for shipment and payment services in the e-commerce industry.

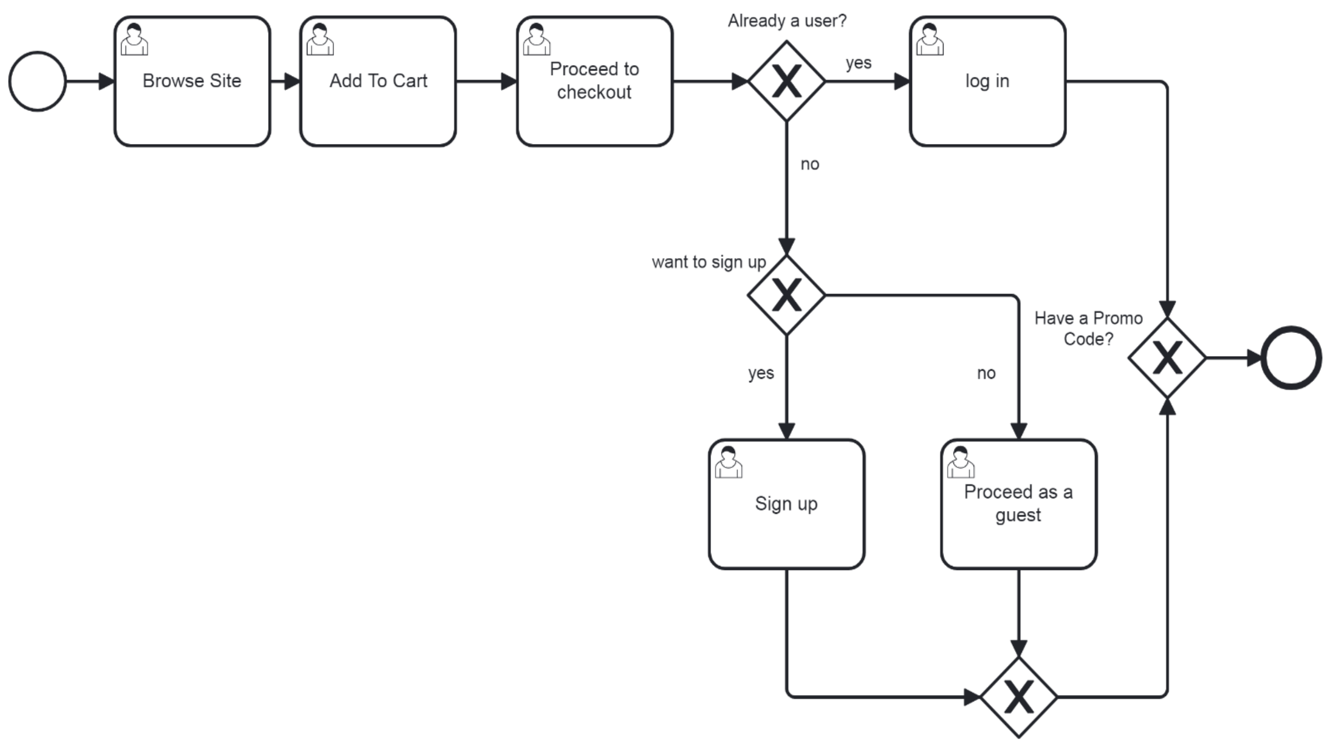


Figure 13. Choreography Model for Checkout-User Authentication Process

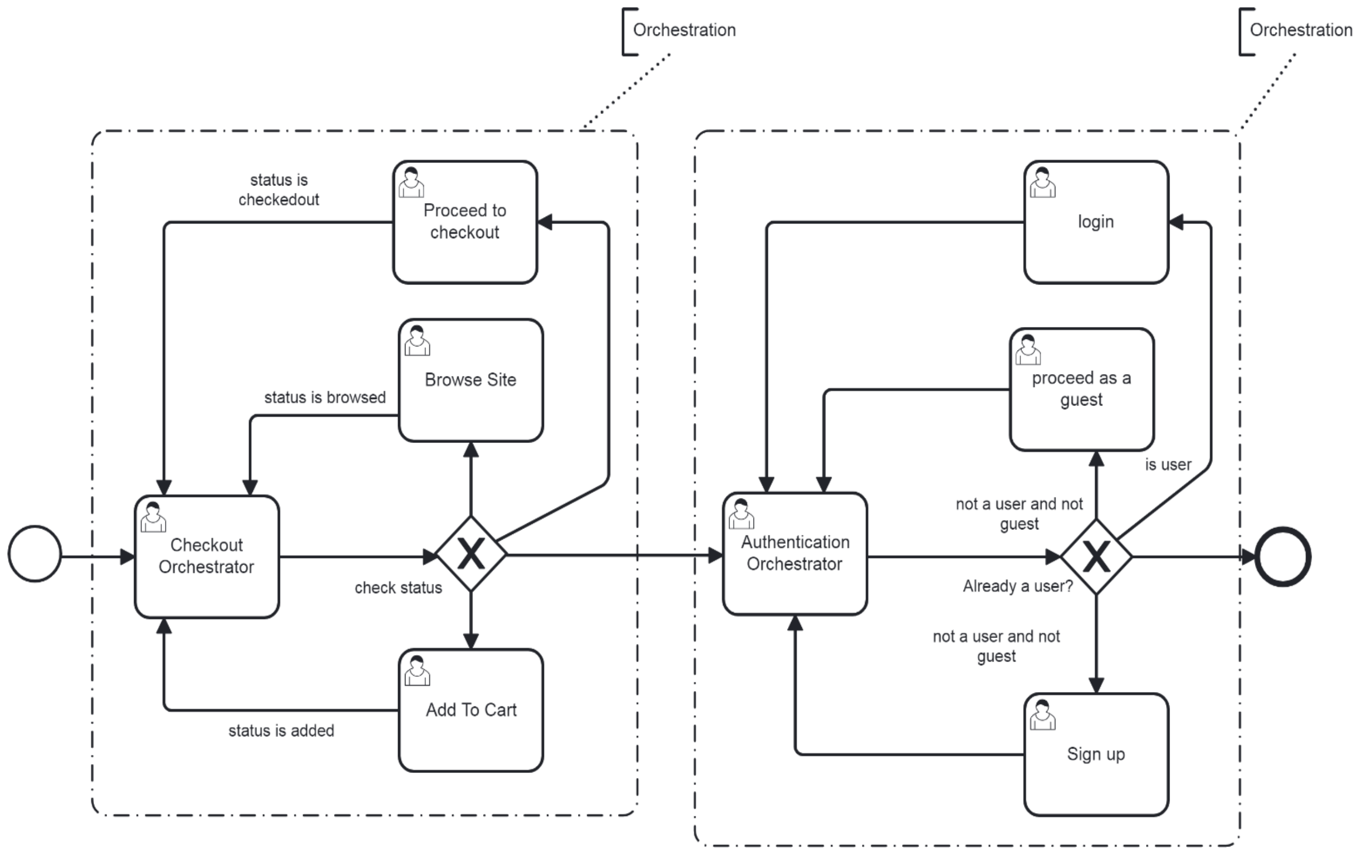


Figure 14. Orchestration Model for Checkout-User Authentication Process

In **Figure 14**, there are two orchestrators: the first one is the "Checkout Orchestrator," and the second one is the "Authentication Orchestrator." Both orchestrators are responsible for controlling and facilitating the messages in the Checkout-user authentication process. The Checkout Orchestrator manages and sends messages related to the checkout process, while the Authentication Orchestrator handles messages related to user authentication. Together, they ensure smooth communication and coordination in the overall Checkout-user authentication process.

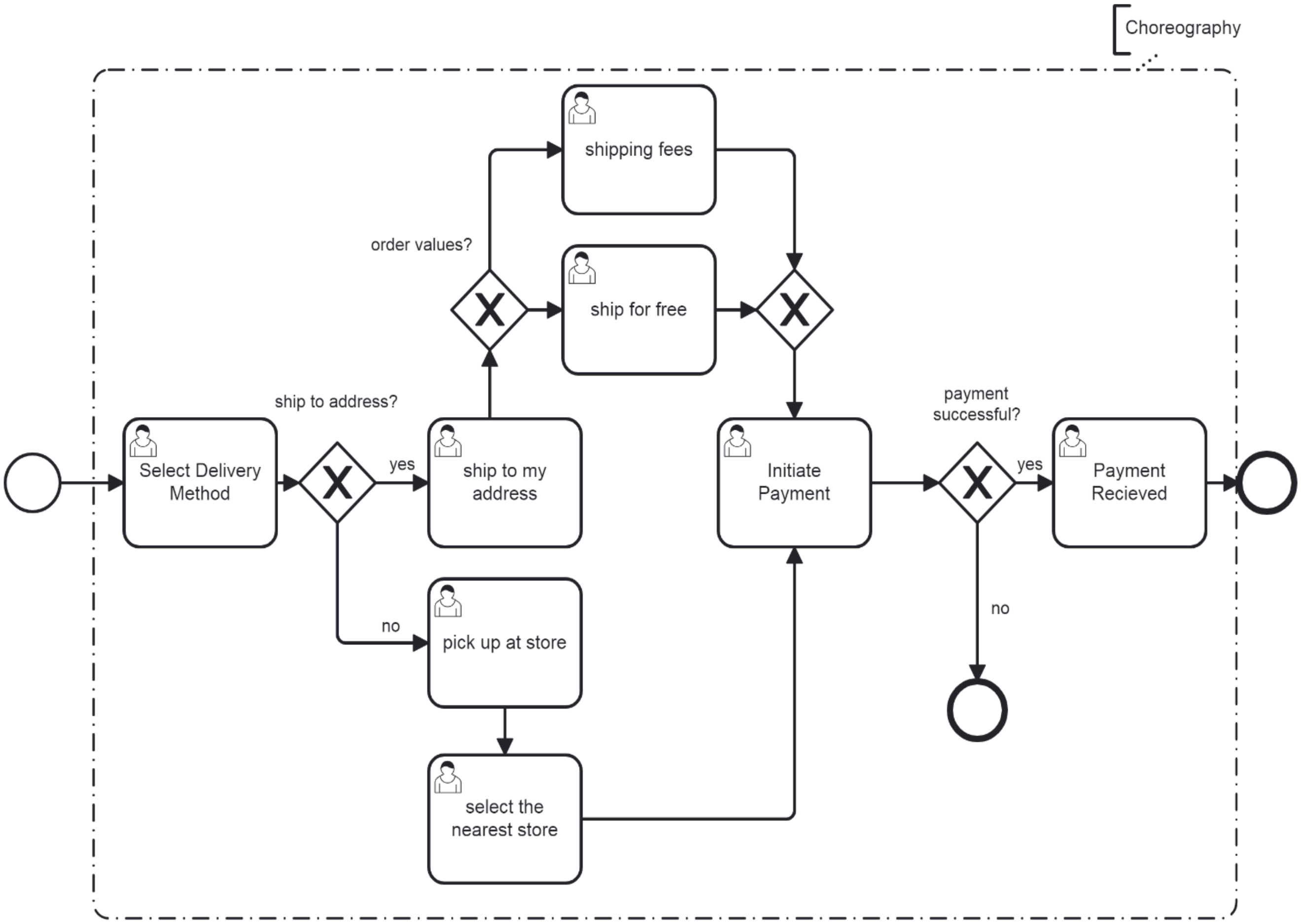


Figure 15. Choreography Model for Shipment and Payment Process

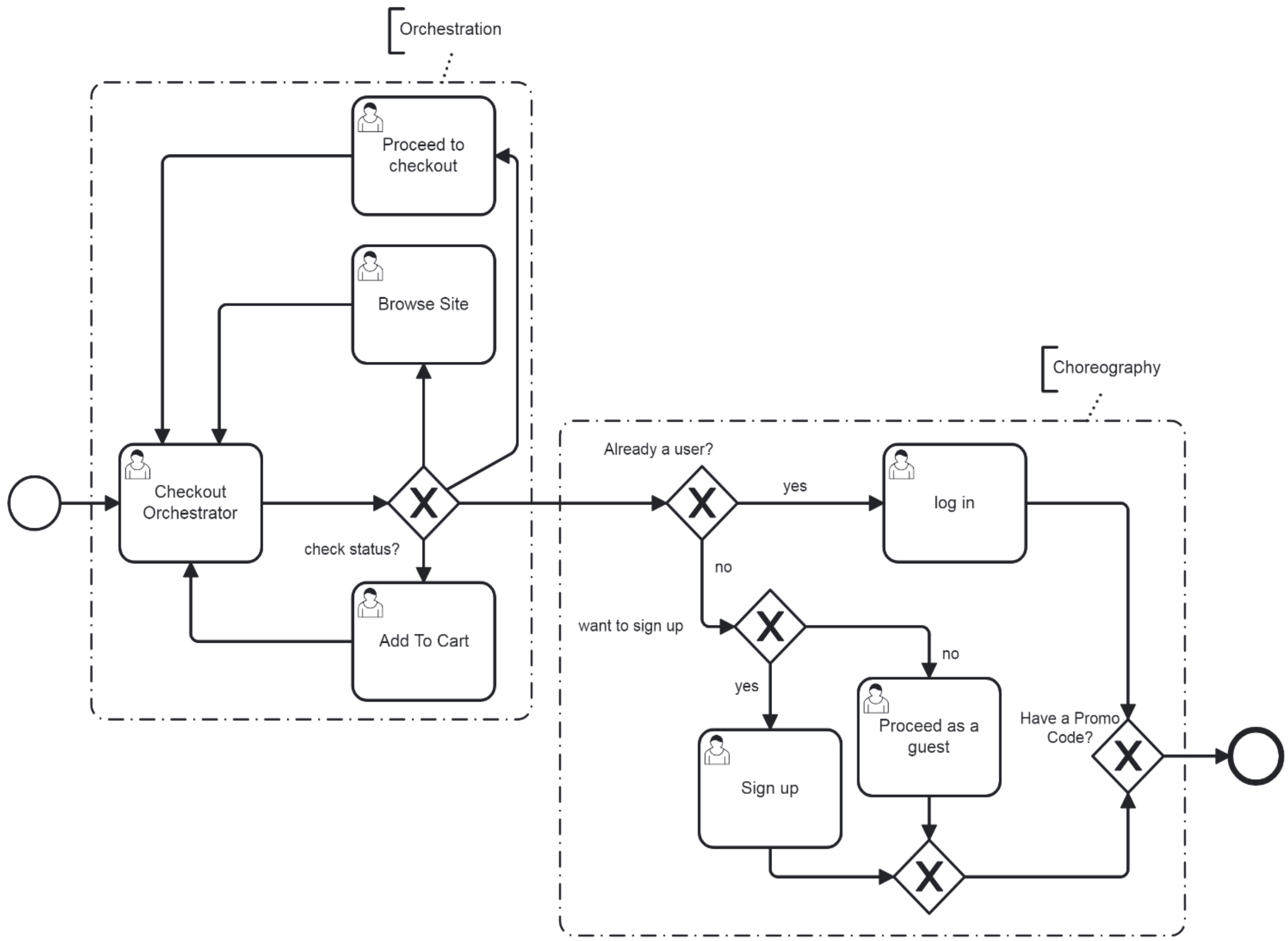


Figure 16. Hybrid Model with one Orchestration and one Choreography

In **Figure 16**, we present a mid-sized BPMN-based hybrid model with one orchestration (Checkout Process) and one choreography (User Authentication) for our research domain.

4.9 End-To-End BPMN Models

In this section, we used the models we created in sections 4.7 and 4.8 to make a new set of complex end-to-end models. We added multiple services to each workflow. We used the Camunda Modeler to design, deploy, and run the workflows. We created one choreography, one orchestration, and three hybrid styles for e-commerce scenarios. **Figure 17** shows the choreography-based process; **Figure 18** shows the orchestration-based process; and **Figures 19, 20,** and **21** show the models designed through our proposed hybrid approach for microservices composition.

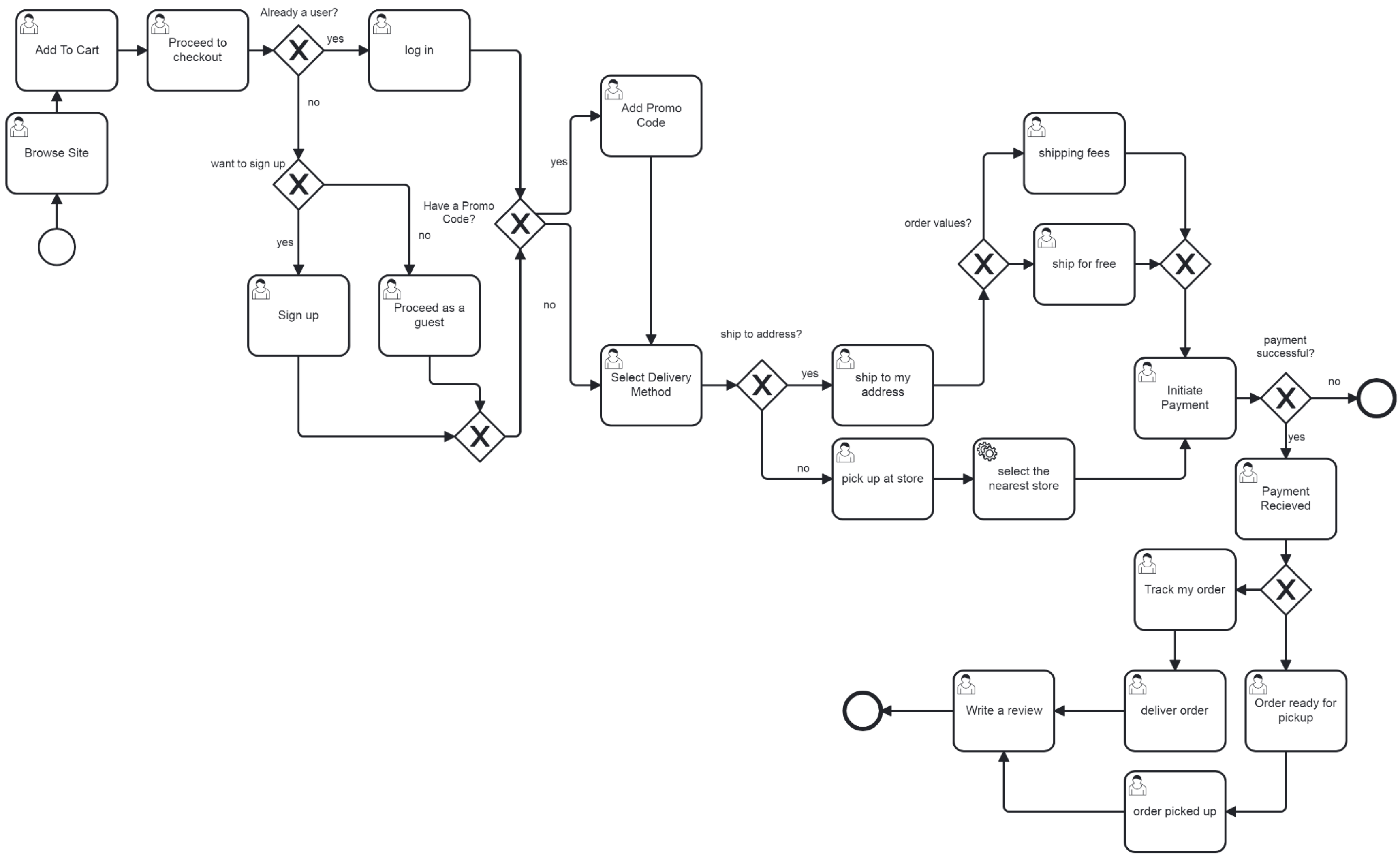


Figure 17. Choreography Model for end-to-end e-Commerce Scenarios

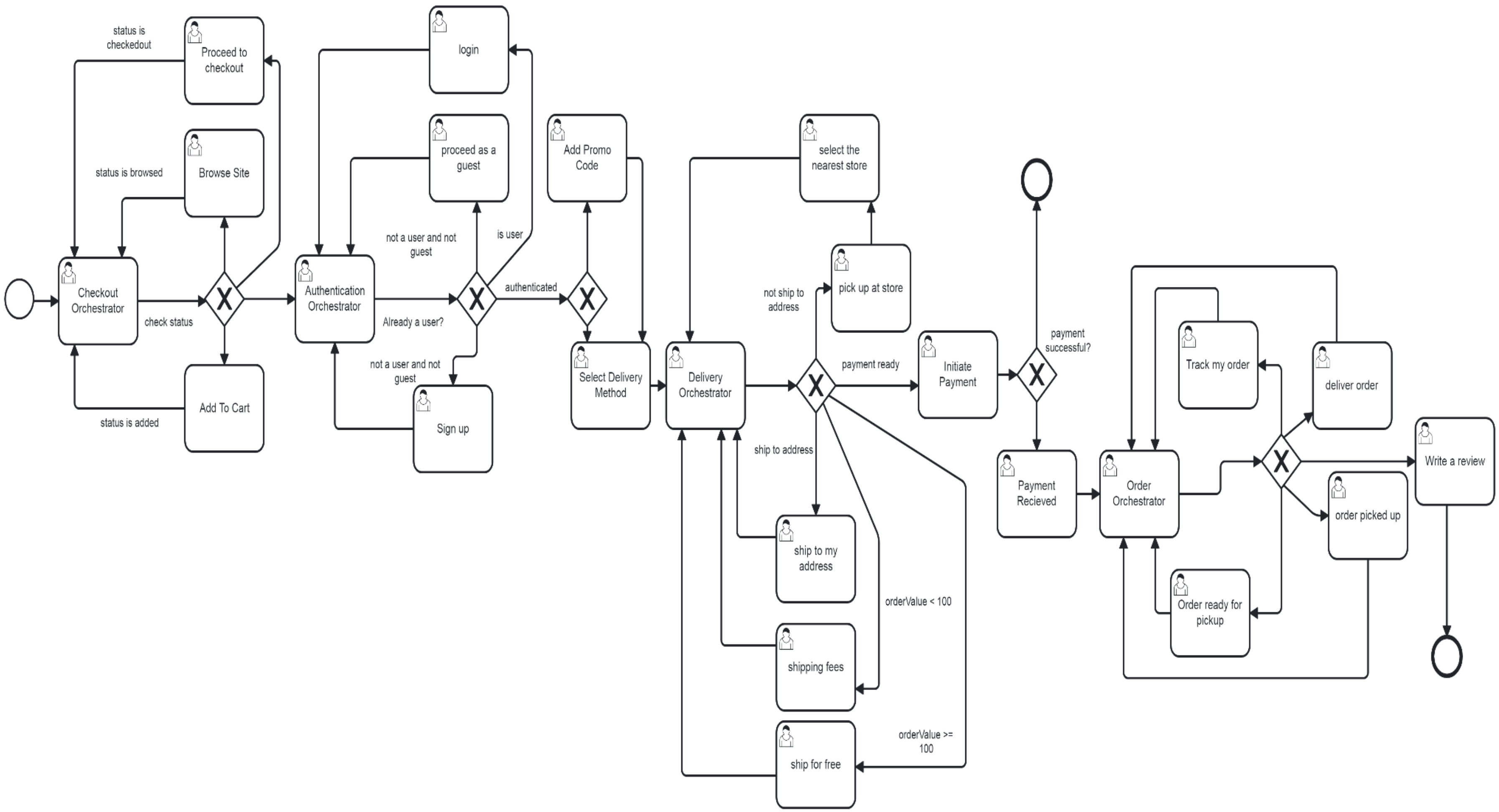


Figure 18. Orchestration Model for end-to-end e-Commerce Scenarios

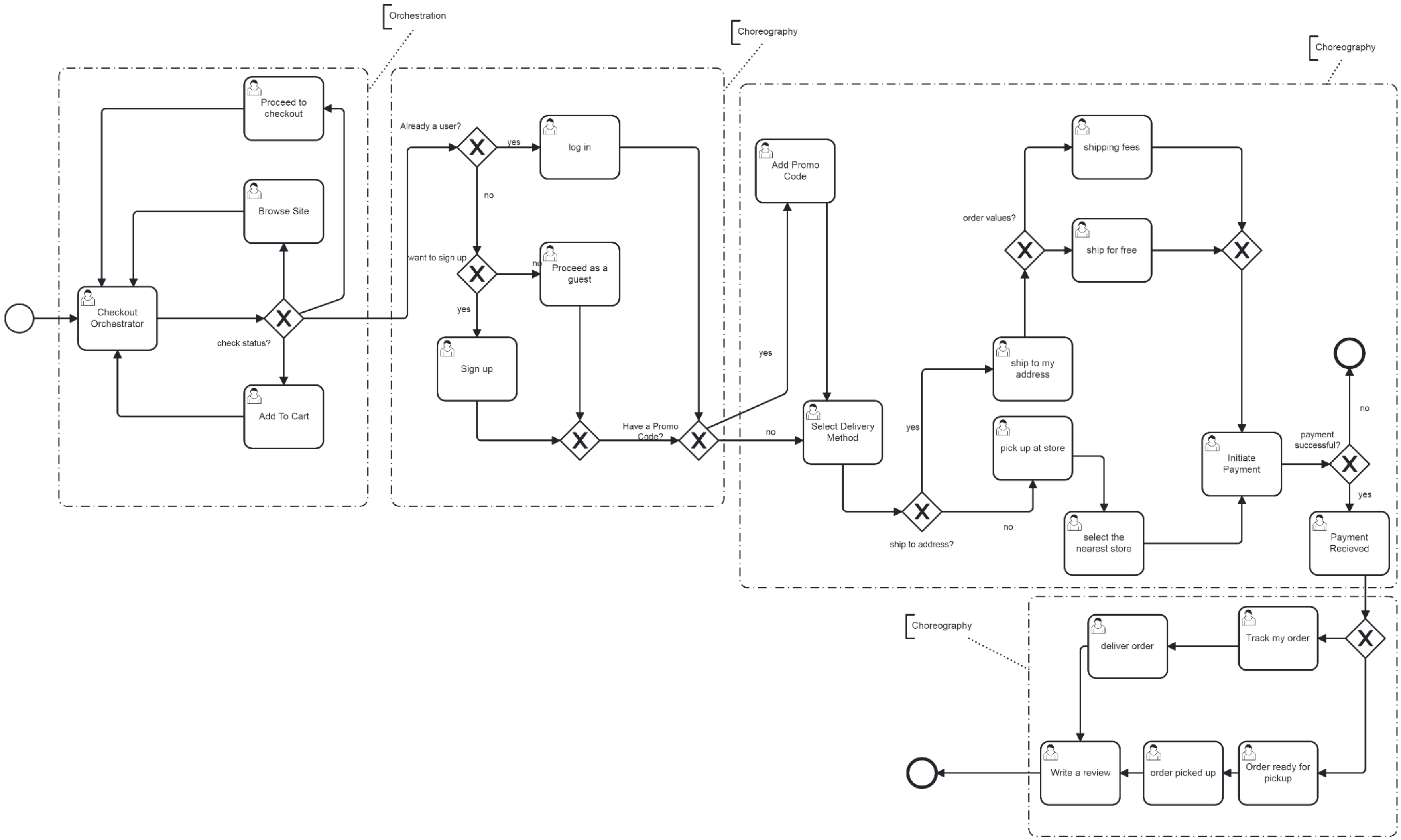


Figure 19. Hybrid (A) Model with one Orchestration and three Choreographies

In **Figure 19**, we present the Hybrid (A) model with one orchestration (Checkout Process) and three choreographies, combined (User Authentication, Payment, Shipment) for end-to-end e-commerce scenarios.

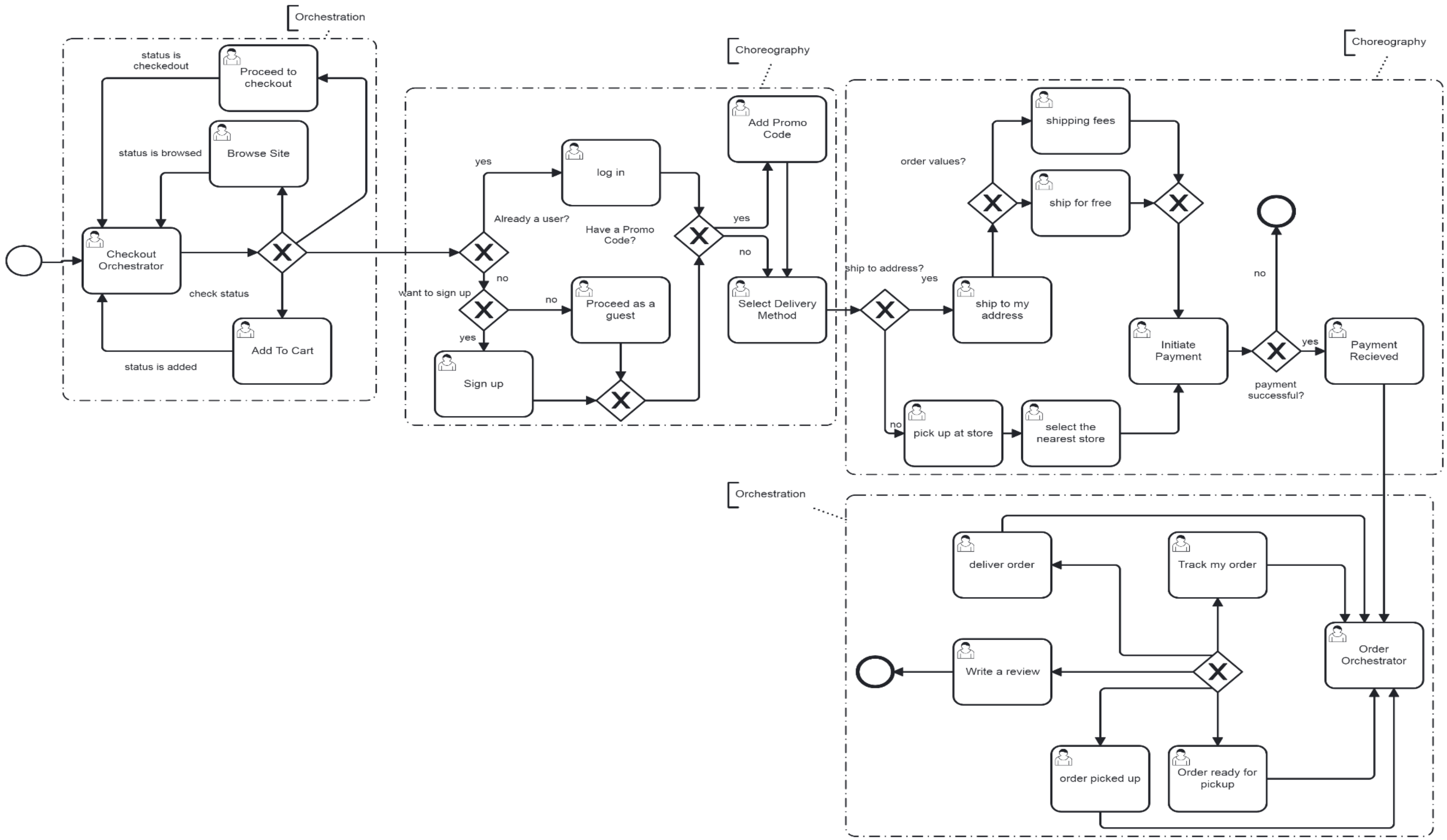


Figure 20. Hybrid (B) Model with two Orchestrations and two Choreographies

A hybrid model with two orchestrations (Checkout Process, Order Process) and two choreographies, combined (User Authentication, Shipment and Payment Process) for end-to-end e-commerce scenarios shows in **Figure 20**.

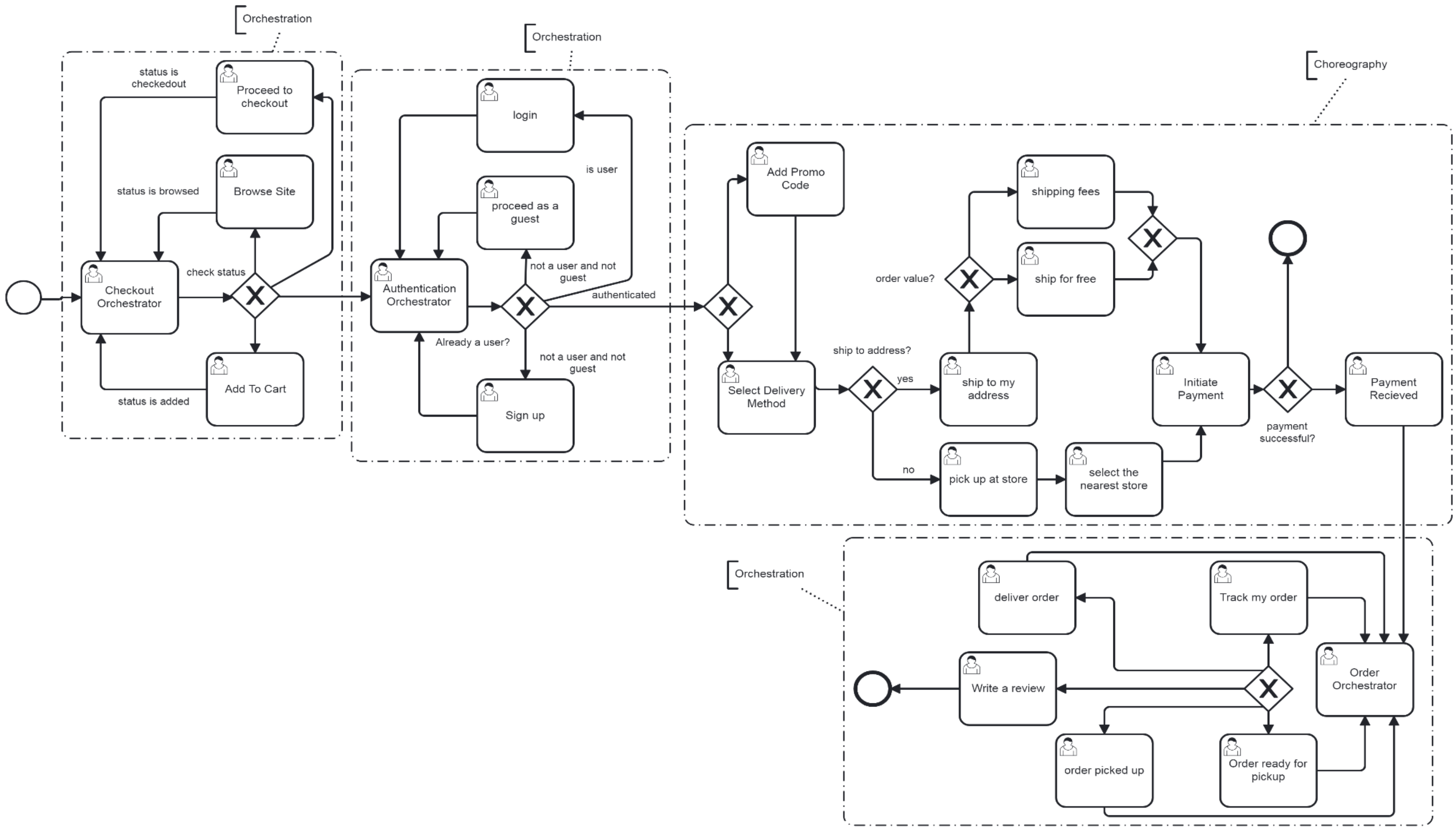


Figure 21. Hybrid (C) Model with three Orchestrations and one Choreography

In **Figure 21**, we define the Hybrid (C) model with three orchestrations (Checkout Process, User Authentication, Order Process) and one choreography, combined (Shipment and Payment Process) for end-to-end e-commerce scenarios.

4.9.1 Strategy for Developing Hybrid Composition

To create hybrid models and determine the optimal approach for microservices composition, we should adhere to the following guidelines:

1. Begin by identifying the business logic and requirements that need to be addressed in the design. Determine the necessary number of services and their respective functionalities. This will enable the breakdown of the entire scenario into smaller components. It is essential to apply microservices principles such as loose coupling, high cohesion, scalability, single responsibility, and resilience.
2. Design small components that involve fewer activities or tasks using the choreography approach. This approach provides loosely coupled services for composition.
3. Define the starting and ending points for each small scenario model.
4. For services with more activities or tasks, employ the orchestration style in the microservices design. This approach ensures a higher level of cohesion in the composition mechanism.
5. To create mid-sized scenarios for hybrid composition, we can combine the start and end events of small-sized scenarios within the workflow.
6. In mid-sized scenarios, prioritize maintainability, coordination between services, and improved communication resilience. We can consider employing orchestration for these aspects.
7. Building a hybrid model involves combining two or more small components using choreography or orchestration-based approaches. We need to remember that the end event of one small or mid-sized model serves as the next sub-process's start event.
8. When integrating multiple small and mid-sized models, we need to keep in mind that to achieve better output and service delivery acknowledgment, it is advisable to minimize the use of choreography in the hybrid model.
9. In complex scenarios involving numerous services and hundreds of tasks, opt for more orchestrations in the hybrid model for optimal practices. This approach provides enhanced control, fault tolerance, and a higher level of cohesion in the business logic of each microservice.

4.10 Deployment of BPMN Models

Camunda Modeler uses visual representations of BPMN to make it easy for both technical and non-technical team members to collaborate, design, and deploy models using the same interface. The Desktop Modeler lets you deploy BPMN models and start process instances directly in Camunda Platform 8 ("Deploy your first diagram," n.d.). According to the Camunda documentation, there are three deployment options available: Helm/Kubernetes, Docker, and manual deployment on a virtual or local machine with Java Virtual Machine (JVM) support. For this research, we chose the Helm/Kubernetes option to deploy our models to an independent host using Amazon Elastic Kubernetes Service (Amazon EKS), which is a managed container service for running and scaling Kubernetes applications in the cloud ("Camunda Platform 8 on Kubernetes," n.d.). To set up the Camunda Platform on our managed host, we followed the steps outlined in the Camunda Platform 8 Helm deployment guide ("Camunda Platform 8 Helm deployment," n.d.).

To begin, we designed our BPMN workflows using the Camunda Modeler. Then, we used the Camunda cloud-native workflow engine powered by Zeebe to deploy our proposed BPMN models. Camunda Cockpit tool allowed us to view BPMN processes in real-time and identify technical incidents or stop workflows if necessary. This tool provides various features such as checking running process instances, open incidents, human tasks, deployment time, activities, creation date, adding variables, and more. Additionally, if developers want to deploy their models in their local machine, Linux/Windows/MacOS, JVM, and Elasticsearch are required. They also need to configure web applications to use an available port. The web applications like Operate and Tasklist listen to port 8080 of the local machine ("Manual installation on local machine," n.d.).

In our study, we used the server IP address 54.203.144.XX to access our cloud host and used port 8181 to deploy our workflows. We prioritized managed host deployment over localhost deployment so that our team members could access and deploy their models. To prevent unauthorized access, editing, or deletion, we added team member IP addresses to the white list under the security group inbound rules of Amazon Elastic Compute Cloud (Amazon EC2). We manually uploaded our entire models and tested them for any incidents during deployment. Typically, deployment failures in the cloud are caused by problems with the model's design, but if the deployment and process instances start successfully, it means the model is functioning properly.

4.11 Discussion

In our work, we thoroughly tested our e-commerce models, ensuring they cover the entire process from placing an order to order delivery. Instead of creating overly complex models with thousands of services, we began with smaller individual components and gradually built a larger system for end-to-end e-commerce scenarios. We deployed the model and manually checked the state, incidents, running instances, and token flow for each task. We looked at the status, any issues, tasks that were currently running, and how the tokens were moving around. All our models are based on "user tasks," and we do not use service tasks, call activities, send tasks, business rule tasks, or any other tasks. We rely solely on XOR gateways for designing these models. Camunda automation engine requires scripts and more technical integration to check the process automation, view the executed workflows' heatmap, understand how processes are functioning, and optimize the flows.

In the model design and deployment phase, we noticed something interesting for choreography modeling, we needed fewer condition expressions and fewer activities including the message flows compared to orchestration and hybrid styles for microservices composition. It was simple to add more tasks in choreography models due to less interaction between activities. Because of this, choreography models generated fewer lines of code in XML which is directly related to the measurement of how big a model is.

However, we had some trouble keeping track of technical incidents that happened with tasks in the choreography approach. On the other hand, orchestration and hybrid approaches needed more condition expression and more activities and message flows. We figured out incidents easily that happened with tasks in orchestration and hybrid approaches because of central orchestrators in the communication mechanism.

We deployed these models to verify the effectiveness of our designs and repeat this process to make the models error free. If the models are deployed successfully and generated tokens from the start point to end point, it means they are error-free. However, if there are any errors in the model, Camunda automation engine prevented us from deploying the microservices models.

CHAPTER 5. Quality Metrics in Software Engineering and Evaluation of Microservices BPMN Models

This chapter describes the concepts of quality metrics for both business process models and software designs. First, we show what quality metrics are available in the software design domain. Second, we present the similarities between quality metrics in software design and business process modeling. We describe the transformation and adaptation of quality metrics, from the software design domain to the business process modeling domain.

In this chapter, we evaluate a hybrid composition of microservices using quality metrics. We use three quality metrics to assess our BPMN models, which we designed in **Chapter 4**. These models include choreography, orchestration, and three end-to-end hybrid models, which we compare in terms of models structural coupling, cohesion, and scalability.

5.1 Introduction

A wide variety of quality metrics have been developed in the area of software engineering to design properly structured programs. Many researchers have identified similarities between business process modeling and software programming, recognizing quality metrics (Vanderfeesten et al., 2007; Khlif et al., 2009; Cardoso, Vanderfeesten, & Reijers, 2010; Yaqin et al., 2017). A well-structured software program's logic is easy to understand, its modules' identification in its design is easy, and its maintainability is easy. In many studies, practitioners and researchers agree that microservices, by nature, must be loosely coupled and highly cohesive (Müller, 2019; Panichella et al., 2021). By adopting quality metrics, we can measure the coupling and cohesiveness of microservices composition in BPMN models.

The majority of studies to evaluate business process models' quality adapt software metrics, and one possible reason is that business processes and software programs/products are quite similar (Panichella et al., 2021; Yaqin et al., 2017). Since workflow models and business process models have much in common with software designs/programs, it is recognized that quality measures for software engineering also have the potential to improve business process modeling.

5.2 Quality Metrics in Software Programs and Business Process Modeling

From the literature, we have identified coupling, cohesion, scalability, complexity, size, and modularity as the most common quality metrics for evaluating software products. They are described below.

Coupling: Coupling describes how strongly modules are related or interconnected with each other (Vanderfeesten et al., 2007).

Cohesion: Cohesion describes the relationship between elements within a module of any software/design product (Vanderfeesten, Cardoso, et al., 2007). It defines how much different operations within one activity are related to each other.

Scalability: the scalability of a system is the ability to handle its growth on the number of jobs or its potential to handle that workload. In the same way, business process modeling scalability can be defined as the ability of a business process model to handle the growth in the number of processes and activities, or the potential of the business process models to be enlarged (Yaqin et al., 2017).

Complexity: Design Complexity in software describes the number and size of the control constructs. When the number of control constructs grows, the size and the number of modules grows and the design complexity increases (Vanderfeesten et al., 2007; Cardoso et al., 2010).

Size: Size describes simply how big a software is, according to certain measures which determine the overall dimension of the software product (Vanderfeesten et al., 2007; Cardoso et al., 2010). In business process modeling, size is a measurement of how big a model is. The size of a business process model can be measured by simple measures, such as the number of lines of code in software programs (Jorge Cardoso et al., 2010).

Modularity: Modularity describes how the software is decomposed into smaller pieces with standard interfaces; it measures the extent to which a service provides independent functionality without relying on another service (Choi & Kim, 2008). For a business process, modularity measures the degree to which a model's design is split up into several modules.

Among these six principles, many studies evaluating the quality of software focus particularly on coupling, cohesion, scalability, and complexity.

5.3 Adaptation of Metrics in Software Programs and Business Process Modeling

Coupling in business process models expresses how tightly the activities in a process are connected and how much they share one or more information elements. Software products and business processes have a similar compositional structure (Oberhauser & Stigler, 2017; Yaqin et al., 2017). Every software program consists of several modules or functions, which are sets of actions. These actions take a certain set of inputs and produce an output. Similarly, every business process model has activities that are composed of elementary operations (events, activities, gateways, message flow). Each operation uses more information to create new information (Yaqin et al., 2017). We have found several studies describing the similarities between software products and business process models and notation (BPMN). The similarities of the software programming modules, methods, variables, and business processes are based on the activity, operation, and compositional structure used in both areas. **Table 5** shows the similarities between software programming and business process modeling (Vanderfeesten et al., 2007).

Table 5: Similarities in Software Designs and Business Process Modeling (Vanderfeesten et al., 2007)

Software Programming	Business Process Modeling
Module/Class	Activity
Method/Function	Operation
Variable/Constant	Data element

Moreover, based on the similarities between object-oriented software and BPMN concepts, Khlif et al (2009) propose the matching shown in **Table 6**.

Table 6: Matching between Object Oriented Software and BPMN (Khlif et al., 2009)

Object Oriented Software	BPMN Notation
Class/package	Process, sub-process
Method	Task
Variable/Constant	Data object
Comment line	Annotation
Interface of a class	Interface of a process/sub process: the set of tasks in a process which send or receive a flow messages.

Local data in a class.	Process task data objects: data objects related to process tasks by associations.
Data used by a class	Data object used by process tasks: data objects associated with message flows going into tasks in the process
Method invocation	Reception of a sequence flow or a message flow by a task.

5.4 Why coupling, cohesion, and scalability for evaluating models?

In our study, we created our models using BPMN. These models were then deployed to the AWS cloud using Camunda microservice-oriented tools. Evaluating cohesion, coupling, and scalability during the design of microservices is crucial because these factors profoundly influence the quality, maintainability, and overall success of the microservices architecture. Low coupling ensures that each microservice can function independently. High cohesion ensures that each microservice has a well-defined purpose, making it easier to understand its role and functionality.

Scalable microservices can be adjusted in response to changing demand, ensuring efficient use of computing resources. Scalable microservices support the growth of the system over time, accommodating increased user numbers and ensuring the system remains robust and these principles are well known practice for microservice architectures (Asri et al., 2022; Asrowardi et al., 2020; de Castro & Rigo, 2023; Shuo et al., 2023). During our study, we identified that cloud native proposed microservices metrics share similarities, such as service complexity, service coupling, and service cohesion (Panichella et al., 2021). Because of this similarity, we decided to use these metrics to assess microservices composition. We believe that these metrics are well-suited to achieve our thesis objectives and helpful in conducting a quality metrics-based analysis.

5.5 Coupling Metrics Adaptation

In general, how modules are interconnected is measured by the coupling metric within business process modeling. For a process that consists of a set of activities *(S)*, the process coupling *cp* is defined as follows (Vanderfeesten et al, 2007):

$$cp = \begin{cases} \dfrac{\left|\left\{(T_1, T_2) \in S \times S \mid \bar{T}_1 \neq \bar{T}_2 \wedge \left(\hat{T}_1 \cap \hat{T}_2\right) \neq \emptyset\right\}\right|}{|\bar{S}| \cdot (|\bar{S}| - 1)}, & \text{for } |S| > 1 \\ 0, & \text{for } |S| \leq 1 \end{cases} \tag{1}$$

Where:

T_1 and T_2 stand for incoming and outgoing flows for each activity (task) (consider the average of incoming & outgoing flows) and S stands for the set of activities in the process.

$\bar{T}_1 and\ \bar{T}_2$ are the sets of resources used by activities T_1 and T_2 respectively.

$\hat{T}_1\ and\ \hat{T}_2$ are the sets of inputs/outputs produced by activities T_1 and T_2 respectively.

$|\bar{S}|$ is the number of elements of set S.

The degree of coupling depends on how complicated the connections are and also on the type of connections between activities (AND, OR, XOR).

5.5.1 What are Imported and Exported Coupling in a Process?

In the software programming domain there are two types of coupling (Khlif et al., 2009):

IC - Imported Coupling which counts all interactions for each class C used by another class.

EC - Exported Coupling which counts all interactions in which C is used.

In business process modeling, we apply these two coupling metrics in the following way:

ICP - Imported Coupling of a Process counts the number of sequences or message flows sent by either the tasks of the (sub-) process or by itself for each (sub-) process.

ECP - Exported Coupling of a Process counts the number of sequences or message flows received by either the tasks of the (sub-) process or by itself for each (sub-) process.

5.5.2 Coupling Metrics Measurement for BPMN Models

In this part, we explain how we used the coupling metrics (which we talked about in the previous chapter) to measure the level of coupling in our designed BPMN models for microservices composition. We used five different workflows: Choreography (D), Orchestration (E), Hybrid (A), Hybrid (B), and Hybrid (C) shown in **Figures 17** to **21** as our examples to apply these metrics for e-commerce scenarios that involve the entire process. We created workflows of different sizes, including small-sized, mid-sized, and end-to-end workflows, (see Chapter 4). However, for this research, we are focusing on complex scenarios to measure the level of coupling. We have measured the complexity of small and mid-sized workflows in our previous work (Ali et al., 2022).

The Response for a Class (RFC) metric is part of a group of measures used for measuring software at the class level. It specifically focuses on examining coupling in terms of control flow. In the business domain, this metric is referred to as Response for a Process (RFP). Locality of data (LD) is a coupling metric that establishes a connection between the data used within an activity (process or sub-process) and the overall data utilized by that activity.

We have identified four coupling metrics in which adaptation is possible in business process modeling: IC- Imported Coupling, EC- Exported Coupling, RFC- Response for Class Coupling, and LD- Locality of Data-based coupling (Khlif et al., 2009).

Our models are centered from storefront and customer perspectives. We did not include administrative viewpoints, so we did not use a database for that purpose. However, some of these ways do not work for our e-commerce scenarios because they need extra elements like BPMN and databases which are not used in our models.

5.5.3 Measurement of Imported and Exported Coupling in a Process

To measure imported coupling in a BPMN process, we focused on activities or elements that depend on inputs from external sources. One way to measure this is by calculating the number of data or message flows coming into the process:

ICP = number of incoming data/message flows

To measure exported coupling in a BPMN process, we assessed the activities or elements that send data or messages to external entities. This we measured by counting the number of outgoing data or message flows:

ECP = number of outgoing data/message flows

We used the ICP and ECP to evaluate all our end-to-end workflows, including hybrid models, choreography, and orchestration. The summary of our findings is presented in **Table 7**.

Table 7: Comparison of ICP and ECP Metrics Results for Composition Models

SL No	Business Process Name	ICP	ECP
1	Hybrid (A)	26	21
2	Hybrid (B)	30	22
3	Hybrid (C)	34	23
4	Choreography (D)	23	20
5	Orchestration (E)	38	23

Looking at the results in **Table 7**, we can see that the choreography approach has the lowest ICP and ECP values. On the other hand, the orchestration approach has the highest ICP and ECP values. Among the hybrid styles, the hybrid (C) model stands out with the highest ICP and ECP values, in this model, we used three smaller orchestration sub-processes. From what we have learned, it is clear that when we include more orchestration in our workflows, the ICP and ECP values tend to become bigger.

5.5.4 Calculating Process Coupling

In a business process, the coupling metric is determined by the number of activities that are related to each other. To measure this, we calculate the average coupling by adding up the number of connections for all activities and dividing the result by the total number of activities (Vanderfeesten et al., 2007). When calculating the process coupling for our designed models, we counted all pairs of activities twice. To obtain the results for each BPMN model, we divide the average coupling by the maximum number of couplings, which is the number of activities minus one (Vanderfeesten et al., 2007). We have applied the above-mentioned metrics to our end-to-end designed workflows and we summarized the results in **Table 8** for hybrid models, choreography, and orchestration.

Table 8: Process Model and Process Coupling (cp)

Process Model	Average Coupling	Process Coupling
Hybrid (A)	23.5	0.056
Hybrid (B)	26	0.056
Hybrid (C)	28.5	0.056
Choreography (D)	21.5	0.057
Orchestration (E)	30.5	0.06

A lower value of cp indicates looser coupling. Looking at the results in **Table 8**, we can see that our proposed hybrid models are less tightly coupled. All the hybrid models have similar process coupling values. On the other hand, the orchestration approach has a higher value, indicating that it is more tightly connected in comparison. Meanwhile, the choreography approach has a better process coupling value when compared to orchestration.

5.6 Cohesion Metrics Adaptation

In business process modeling, the cohesion of an activity is the product of both information cohesion and relation cohesion (Khlif et al., 2009). The overall cohesion of a BPMN model can then be determined by the average activity cohesion. We can determine the overall cohesion of the workflow process by calculating the average activity cohesion. For a business process which consists of a set of activities *(S)* on the operations structure and *c(t)* is the activity cohesion, the process cohesion *ch* can be defined as follows (Vanderfeesten et al., 2007):

$$ch = \frac{\sum_{t \in \tilde{S}} c(t)}{|\bar{S}|} \qquad (2)$$

Where:

ch represents the process cohesion, which is a measure of how closely related or interconnected the activities within the process.

$\sum_{t \in \tilde{S}} c(t)$ is the summation of cohesion values $c(t)$ for each activity t in the set $\tilde{S}$ which typically represents a subset of activities within the process.

$c(t)$ is the cohesion of activity t, indicating how well the elements within the activity are focused and related. A higher cohesion value implies a stronger internal relationship.

$|\bar{S}|$ is the number of elements of the set $\bar{S}$, which represents the total set of activities in the process.

5.6.1 Cohesion Metrics Measurement for BPMN Models

As discussed in section 5.6, we applied the cohesion metric to our sample models, including Hybrid (A), Hybrid (B), Hybrid (C), Choreography (D), and Orchestration (E). The cohesion metric measures the total incoming flows in a business process, which is then divided by the outgoing flows of the business process multiplied by the outgoing flows of the business process minus one (Vanderfeesten et al., 2007).

As we can see from **Figure 17**, in the Hybrid (A) workflow, there are 26 incoming flows and 21 outgoing flows. Process Cohesion (ch) increase if the cohesion level of a workflow increases. Higher ch values signify greater internal consistency and a tightly cohesive structure. **Table 9** provides insights into the cohesion of different methods. The orchestration method stands out with the highest process cohesion value, indicating that this model has a strong internal connection or highly cohesive. Among the models we proposed, the two hybrid models show similar levels of process cohesion. One of the hybrid models follows closely with the second highest process cohesion value.

Table 9: Process models and Process Cohesion (ch)

SL No	Process Model	Process Cohesion
1	Hybrid (A)	0.06
2	Hybrid (B)	0.06
3	Hybrid (C)	0.07
4	Choreography (D)	0.06
5	Orchestration (E)	0.08

5.7 Coupling/Cohesion Ratio

The process coupling/cohesion ratio, ρ can be defined as follows:

$$\rho = \frac{cp}{ch} \qquad (3)$$

ρ represents the process coupling and cohesion ratio, which provides an integrated assessment of both coupling and cohesion aspects within the process.

cp stands for process coupling, a metric that quantifies the degree of interdependence or interaction among activities within the process.

ch represents process cohesion, a measure of how well-related and internally focused the elements within activities are.

We applied coupling and cohesion metrics to all our designed end-to-end/complex workflows, and we presented the results in **Table 8** and **Table 9** for hybrid, choreography, and orchestration models. These metrics help us identify the most preferable business process modeling design by considering its execution maintainability and quality. If a model has a minimal coupling/cohesion ratio, it is the most favorable model. The coupling/cohesion ratio can help us choose the best option from design models based on the specified metrics (Vanderfeesten et al., 2007). **Table 10** presents five workflows coupling and cohesion ratios.

Table 10: Process models and Coupling Vs Cohesion Ratio

SL No.	Process Model	Coupling Vs. Cohesion Ratio
1	Hybrid (A)	0.93
2	Hybrid (B)	0.93
3	Hybrid (C)	0.80
4	Choreography (D)	0.95
5	Orchestration (E)	0.75

In the case of our sample workflows, the coupling/cohesion ratios for the five designs are 0.75, 0.80, 0.93, 0.93, and 0.95. Based on these ratios, if we are considering a hybrid model for an end-to-end e-commerce scenario, then the Hybrid (C) model is the most suitable. This model includes three orchestrations (Checkout Process, User Authentication, and Order Process) and two choreographies (Shipment and Payment Process) combined.

5.8 The Control Flow Complexity (CFC)

The control flow complexity (CFC) refers to the various decision points, or gateways, within a process flow. Counting all types of gateways is necessary to determine the CFC, which increases with the number of gateway types. CFC metrics help analyze the complexity of a business process model. In simpler terms, CFC is a measure of how complicated a process is based on the number and types of decision points within it (J Cardoso et al., 2006; Yaqin et al., 2017):

X stands for the number of Processes and A is an Activity. We can express the CFC measurement equation as below, and this is formulated as number (4) for our research.

$$\textbf{CFC(X)=} \sum_{A \in P, A\ is\ a\ XOR-split} CFC_{XOR}\ (A) + \sum_{A \in P, A\ is\ a\ OR-split} CFC_{OR}\ (A) + \sum_{A \in P, A\ is\ a\ AND-split} CFC_{AND}\ (A) \qquad (4)$$

- $CFC_{XOR-split}$(A)= fan − out(A). The control-flow complexity of XOR- splits is determined by the number of outgoing branches that emerge after each XOR split.
- $CFC_{OR-split}$(A)= fan−out(A)−1. The control-flow complexity of OR-splits is determined by the number of outgoing branches that emerge after each OR split.
- $CFC_{AND-split}$(A)=1. The control-flow complexity of AND-splits is determined by the number of induced states that are introduced with the split.

5.9 Scalability Metrics in Business Process Modeling

Scalability in a business process can be defined as the ability of the process to accommodate growth or handle the growing number of business processes (Yaqin et al., 2017).

Process similarity is the identification of similarities in business process models. The similarity value determines whether a business process is a subset of other business processes or not. In order to assess the scalability of a business process, we need structural and behavioral similarities as scalability parameters (Yaqin et al., 2017). With the Jaccard coefficient method, we can calculate workflow similarities.

The structural similarity is defined as the similarity of two graph structures; the graph structure consists of all elements in the model/graph. In our research, we use the Jaccard coefficient method to measure structural and behavioral similarities of workflows.

Jaccard coefficient is formulated as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad (5)$$

The Jaccard coefficient is a way to find out how similar two business process models are. We can calculate it using the following formula:

$J(A, B)$ = (Number of elements that are in both business process model A and business process model B)/(Number of elements in either business process model A or business process model B, or both).

In other words, the Jaccard coefficient = (Number of common elements)/(Total number of unique elements in both models)

In equation 5, we need to use activities, sequence flows, gateways, start events, and end events, which are represented by process model A and process model B. These elements are essential for calculating the Jaccard coefficient, denoted as $J(A, B)$.

Behavioral similarity refers to the least number of total sequence flows common to each workflow. The behavioral similarity value is derived from the relationships between the existing business process activities of workflows. Two scalability parameters for assessing business process models are structural and behavioral similarities. A business process as a subset of other business processes or not is identified based on the value of the similarity determined (Yaqin et al., 2017).

$$Sim(A, B) = average\ (simS(A, B), SimB(A, B)) \qquad (6)$$

In equation 6, sim$S$ stands for simulated value of structural similarities and sim$B$ stands for simulated value of behavioural similarities.

Business process scale refers to comparing one business process to another by multiplying the number of elements it has with the complexity of its flow control (CFC). This helps us measure the size and complexity of a business process compared to others (Yaqin et al., 2017).

$$Scale(A) = E(A) * CFC(A) \qquad (7)$$

Here, *E(A)* is the number of elements for process model A and *CFC(A)* the control flow complexity of process model A.

5.9.1 Scale Value Comparison

To obtain scalability, we need to compare the scale value of one model to the scale values of other business process models. According to Yaqin et al., we can compare one scale value with the formula number (8) (Yaqin et al., 2017).

$$scale(A,B) = \frac{scale\ (A)}{scale\ (B)} \qquad (8)$$

Here, *Scale(A)* is the scale value of process model A and *Scale(B)* the scale value for process model B.

5.9.2 Scalability Measurement for BPMN Models

In this section, we explain how we used the scalability metrics discussed in section 5.9 to evaluate the scalability of our designed models. To assess the scalability of the models, we need to measure the similarities between workflows based on their structural and behavioral aspects.

We applied this metric to the workflows shown in **Figures 17, 18, 19, 20,** and **21**, and the results are presented in **Table 11**. To calculate the structural behavior, we counted the total number of elements in a process, which includes tasks, sequence flows, decision boxes, and start and end events. These elements primarily define the structural behavior of the process.

The Transition Adjacency Relations (TAR) is the sequence flow between the elements presented in the process model. To calculate the TAR, we count all the messages or steps in each sample model. The sequence flows primarily define the behavior of the process.

To measure the complexity of the control flow, we look at the number of decision boxes or gateways in the figures mentioned above. We apply the CFC formula as described in Section 5.3.4 of Chapter 5. **Table 11** is a useful tool to compare the structural and behavioral similarities when designing each sample model. It summarizes the necessary characteristics of the five models:

Table 11: Elements, Transition Adjacency Relations, and Control Flow Complexity

SL No.	Process Model	No. of Elements	Transition Adjacency Relations (TAR)	Control Flow Complexity
1	Hybrid (A)	34	41	10
2	Hybrid (B)	35	46	10
3	Hybrid (C)	34	46	8
4	Choreography (D)	32	37	9
5	Orchestration (E)	33	47	6

5.9.3 Calculating the Structural Aspect

In Section 5.9, we use the Jaccard Coefficient formula to measure the structural similarity aspect of the models. This involves dividing the number of elements in the Hybrid (A) model of the workflow sample by the number of elements in the Hybrid (B) model (Yaqin et al., 2017). We applied all the mentioned metrics to all of our end-to-end e-commerce scenarios and summarized the results in **Table 12** for hybrid models, choreography, and orchestration.

Table 12: Results of the Structural Similarity Calculation with Jaccard Coefficient

	Process Model	A	B	C	D	E
Structural similarity	Hybrid (A)	1	0.97	1	1.06	1.03
	Hybrid (B)	0.97	1	1.03	1.09	1.06
	Hybrid (C)	1	1.03	1	1.06	1.03
	Choreography (D)	1.06	1.09	1.06	1	0.97
	Orchestration (E)	1.03	1.06	1.03	0.97	1

5.9.4 Calculating the Behavioral Aspect

As explained in section 5.9, Behavior Similarity refers to the least number of Transition Adjacency Relations (TAR) or sequence flows used in a workflow. Based on the workflow in **Figures 17 to 21,** the behavioral similarity was calculated by the Transition Adjacency Relations (TAR) Jaccard Coefficient formula of equation number (5). Table 13 provides a summary of the results for hybrid, choreography, and orchestration models. To calculate the behavioral similarity, we divided the number of TAR in the Hybrid (A) model of the workflow sample by the number of TAR in the Hybrid (B) model (Yaqin et al., 2017). We applied these metrics to all our end-to-end workflows.

Table 13: Results of the Behavioral Similarity Calculation with Jaccard Coefficient

	Process Model	A	B	C	D	E
Behavioral Similarity	Hybrid (A)	1	0.89	0.89	1.11	0.87
	Hybrid (B)	0.89	1	1	1.24	0.98
	Hybrid (C)	0.89	1	1	1.24	0.98
	Choreography (D)	1.11	1.24	1.24	1	0.79
	Orchestration (E)	0.87	0.98	0.98	0.79	1

5.9.5 Simulated Average Value of Similarity

Table 14 shows the average values of structural and behavioral similarities obtained from our end-to-end workflow designs. We used formula number (6) from section 5.9 to calculate these averages based on the workflows presented in **Figures 17** to **21**. The similarity value is determined by whether a business process is a subset of other business processes or not (Yaqin et al., 2017). The following table presents the average structural and behavioral similarities of the 5 models given in our study. The value is always '**1**' within the same model.

Table 14: Results of the average value of Structural and Behavioral similarity

	Process Model	A	B	C	D	E
Average of Structural & Behavioral Similarity	Hybrid (A)	1	0.93	0.95	1.09	0.95
	Hybrid (B)	0.93	1	1.02	1.17	1.02
	Hybrid (C)	0.95	1.02	1	1.15	1.01
	Choreography (D)	1.09	1.17	1.15	1	0.88
	Orchestration (E)	0.95	1.02	1.01	0.88	1

5.9.6 Determine the Control Flow Complexity and the Scale Value

In section 5.8, we use the Control Flow Complexity (CFC) metric to measure the complexity of our workflows. This metric is calculated by adding up all the split builders using a formula discussed in that section. We apply the CFC metric to the sample workflows for hybrid models, choreography, and orchestration shown in **Figures 17** to **21**. Since our designed models have only the XOR gateway, the CFC for each workflow equals the total number of CFC_{XOR} for formula number (4).

Table 15 summarizes the Control Flow Complexity based on the XOR gateway used in our workflows. The table shows that the orchestration model has a CFC of 6, indicating a lower level of complexity compared to choreography and hybrid models. However, when considering the hybrid models, Hybrid (C) in **Figure 19** has the lowest complexity among all three hybrid models.

Table 15: Control Flow Complexity and the Scale Value

	Process Model	No. of Elements	CFC	Scale Value
Calculation of Scalability Step 1 - Determine the Scale Value of Each Model	Hybrid (A)	34	10	340
	Hybrid (B)	35	10	350
	Hybrid (C)	34	8	272
	Choreography (D)	32	9	288
	Orchestration (E)	33	6	198

To measure the Scale Value, we use the formula in section 5.9, equation number (7) Scale Value (scale(A)) was obtained by multiplying the number of business process model elements by the Control Flow Complexity (CFC) value. As evident from **Table 15**, Hybrid (B) workflow has the highest scale value of 350 among the three hybrid models, and Orchestration (E) shows the lowest scale value of 198.

5.9.7 Business Process Models Scale Value Comparison

As discussed in section 5.9.1, we apply the scale value comparison formula, equation number (8**)** to sample models in **Figures 17** to **21,** our designed business process models, and we summarized the results in **Table 16,** below, which includes the scale values of the hybrids, choreography, and orchestration workflows.

Table 16: Business Process Models Scale Value Comparison

	Process Model	A	B	C	D	E
Calculation of Scalability Step 2 - Comparing Scale Value Between Models	Hybrid (A)	1	0.97	1.25	1.18	1.72
	Hybrid (B)	0.97	1	1.29	1.22	1.77
	Hybrid (C)	1.25	1.29	1	0.94	1.37
	Choreography (D)	1.18	1.22	0.94	1	1.45
	Orchestration (E)	1.72	1.77	1.37	1.45	1

5.9.8 Comparison of the Scalability Metric Results

We have applied all the scalability metrics to our sample end-to-end workflows, and the results are summarized in Table 17 for hybrid, choreography, and orchestration models.

Scalability value (A, B) was obtained by comparing the scale value of process models A and B obtained in section 5.9.7, **Table 16**, multiplied by the value of similarity obtained in section 5.9.5 and **Table 14**.

If the scalability value for a specific model is zero, it means that the workflow being assessed is the same and cannot be scaled any further. If the value is one or more, it means the two workflows as completely different and can be grown based on requirements. **Table 17** summarizes the scalability factor for the five given models, with the same model always having a value of '0'. To better explain this table, we can compare it with the process model names - Hybrid (A), Hybrid (B), Hybrid (C), Choreography (D), and Orchestration (E). Our goal is to assess how well each model can handle additional growth by adding tasks or activities to meet the business requirements for delivering expected services related to e-commerce.

Table 17: Business Process Models Scalability Comparison

	Process Model	A	B	C	D	E
Calculation of Scalability Step 3 - Scalability Value	Hybrid (A)	0	0.9	1.19	1.29	1.63
	Hybrid (B)	0.9	0	1.32	1.43	1.81
	Hybrid (C)	1.19	1.32	0	1.08	1.38
	Choreography (D)	1.29	1.43	1.08	0	1.28
	Orchestration (E)	1.63	1.81	1.38	1.28	0

5.10 Discussion

We have analyzed the structural coupling, cohesion, and scalability of our proposed microservices models. We made three different sizes for the choreography, orchestration, and hybrid models. However, for the small size, we could not create a hybrid composition, and for the mid-sized scenarios, we could only combine one choreography with one orchestration approach for the hybrid model. Our goal was to evaluate end-to-end scenarios for e-Commerce. We did not find enough reasons to assess the coupling, cohesion, and scalability of small and mid-sized scenarios. Nevertheless, they are still valuable in helping us design the end-to-end model and compare them with five different types of workflows.

We found the smallest ratio of coupling cohesion for the orchestration style, which is the most favorable model among those we designed, according to the findings in **Table 10**. If we are only focusing on coupling metrics, our suggested hybrid models are loosely coupled, as shown in **Table 8**. However, if someone values cohesion more and wants to prioritize highly cohesive models based on their business needs, then they might prefer the orchestration style, as indicated in **Table 9**. Our assessment clearly shows that the orchestration approach is highly cohesive. We also observed that the Hybrid (C) model has the second highest process cohesion value and is slightly different from the orchestration model, as seen in the same table.

We have assessed scalability from a static viewpoint, focusing on the structure of the models. According to **Table 17**, if someone wants to combine orchestration and choreography in one model, the Hybrid (B) model is a good choice for an e-commerce scenario considering scalability. Among the hybrid models, the Hybrid (B) workflow is the second best for handling more growth in the table. Based on the results, the orchestration model is the most scalable among all the models listed in the table. However, we found that the choreography model is the least capable of handling growth in this study.

CHAPTER 6. Conclusions and Future Work

6.1 Discussion

Our research focuses on developing a way to compose microservices and identifying when hybrid methods for modeling microservices should be used. To support our findings, we have used three quality metrics from the literature to evaluate our BPMN-based designed workflows. Furthermore, we have compared the results of our proposed models' coupling, cohesion, and scalability metrics with two other composition techniques, namely choreography and orchestration. Our goal is to provide insights into the potential advantages of using hybrid approaches for microservices composition.

The findings of the first research question (**RQ1)**, as in **Table 4**, section **3.3**, present advantages and disadvantages of orchestration, choreography, and hybrid composition.

In choreography-based composition, it is easier to make changes to microservices and BPMN because control is shared among different services that are not tightly connected. This means it is easy to add or remove services as long as the system is straightforward. If a scenario is more complicated and requires coordinating multiple services, adding or removing tasks, debugging is simpler and faster when using the orchestration approach for microservices composition. However, there is a risk of downtime during development because of the strict dependency point in orchestration.

The results of **RQ2,** as in **Table 4,** section **3.3,** indicate that smaller applications with fewer services are better suited for choreography-based composition when only a few services are needed. Developers and designers should consider various factors, including business requirements, number of services, and functionalities of each service, when choosing composition methods.

For medium-sized applications with more services, orchestration is more suitable as it offers better maintainability and coordination between services. In microservices composition, having better visibility is important for adding or removing services without disrupting communication.

In complex scenarios where thousands of tasks, critical new features and services need to be delivered and a wide range of functionalities is required, such as in an e-commerce system, it is essential to confirm communication between services. In this case, a hybrid composition style should be considered.

In order to answer our **RQ3,** we have designed four hybrid models in **Chapter 4** of this study using multiple orchestrators and combined choreography and orchestration-based approaches in a single workflow. We have developed various ways to combine choreography and orchestration in microservice architectures, all within a single flow. To accomplish this, we divided the process into smaller components then employed multiple choreography-based sub-processes and orchestrations within each workflow. Those workflows represent the way to compose hybrid approaches for microservices. We have compared hybrid composition to choreography and orchestration in terms of coupling, cohesion, scaling, and complexity for e-commerce end-to-end scenario.

We tested the models we designed to make sure they work for the entire e-commerce process from the storefront view. Instead of creating complicated models with lots of services, we started by creating small components and then gradually put them together to make a bigger system for the end-to-end e-commerce process. Our goal was to include all the necessary business logic for active e-commerce operations.

Even though we found several papers in the literature suggesting that choreography results in less tightly coupled microservices, our evaluation findings in **Table 8** shows that our designed hybrid models are less tightly coupled when compared to both orchestration and choreography approaches. The findings of **Table 9** show that the orchestration style is highly cohesive for microservice composition compared to choreography and hybrid models.

In **Table 10**, results show that the orchestration model has the lowest coupling/cohesion ratio, and the Hybrid (C) workflow has the second-lowest coupling vs. cohesion ratio. The Hybrid (C) model uses three orchestrations (Checkout Process, User Authentication, and Order Process) and two choreographies (Shipment and Payment Process). However, if we want better control over microservices composition, we should consider using the Hybrid (C) workflow.

Another conclusion derived from the results is that the Hybrid (A) and Hybrid (B) workflows have a similar coupling/cohesion ratio. The Choreography (D) workflow has the highest coupling vs. cohesion ratio for end-to-end e-commerce scenarios. The results of our evaluation support minimal coupling and cohesion, as per the preferable workflow proposed by Vanderfeesten et al., (2007).

Based on the analysis results in **Table 17**, we can see that among hybrid models, Hybrid (B) workflow has the second highest scalability value. According to this result, the Orchestration (E) end-to-end scenario has better scalability than all the other composition models in the table.

The analysis shows that using more orchestration sub-processes in the workflow improves the coupling, cohesion, and scalability of the hybrid model, which combines orchestration and choreography styles.

In conclusion, based on our metrics-based analysis and considering our modeling approaches for microservice composition, we can say that the hybrid style is not just a concept but can be implemented in real-time business operations. The hybrid composition design is possible for microservices with multiple orchestrators and a more readable visual model. However, our study found orchestration has better coupling and cohesion ratio, and scalability features compared with choreography, and is only slightly different from the hybrid model. We have found that our proposed hybrid models are loosely coupled in comparison to our designed choreography, orchestration, and hybrid models, based on the analysis of coupling metrics. If e-commerce companies or other enterprises wish to reduce dependencies between services, they can use the hybrid technique for composition instead of orchestration.

6.2 Conclusion

For our three research questions formulated for this study, we were able to identify microservices composition approaches and BPMN-based modeling to answer them. Choreography, orchestration, and hybrid are the 3 main composition styles discovered in the literature. We have answered our "**RQ1. What are the advantages and disadvantages of microservices orchestration, choreography, and hybrid composition?**" and "**RQ2: In what situation should we use the hybrid techniques for microservices composition?**". We have discussed and explored the differences between these composition approaches, and analysis in the literature has indicated why one is better than the other and in what circumstances enterprises or designers should select choreography, orchestration, or hybrid styles.

The third research question was "**RQ3: How does hybrid composition compare to choreography and orchestration in terms of coupling, cohesion, scaling, and complexity for e-commerce systems?".** This we answered by designing multiple hybrid workflows and analyzing them based on three metrics. We suggested a way to compose microservices using a hybrid approach with BPMN and a workflow engine. Instead of using a single orchestrator, we used multiple orchestrators in an e-commerce system. We also discussed how coupling, cohesion, and scalability can affect the design and visibility of the composition styles.

We explored an interesting result: using a combination of orchestration and choreography makes it possible to achieve loosely coupled features for microservices composition (our designed hybrid method). The more orchestrations we combine with choreography, the higher cohesion and better scalability they offer in microservices composition. These results can be useful for researchers, engineers, developers, and organizations who need guidance on how to compose microservices. We analyzed metrics to get insight into modeling and understand the structure of choreography, orchestration, and hybrid approaches from a design perspective. This research is helpful to a designer in deciding when to apply a hybrid model and why they should apply this style in microservices composition.

6.3 Research Contributions

With our work, we intend to make a contribution to the hybrid-based composition approach research area of microservices. We conducted a literature review on microservices composition to compare benefits and offer critiques about choreography and orchestration. Furthermore, our literature review in Chapter 3 identified conditions for adopting a hybrid approach. This study provides insights into the BPMN-based composition of microservices in choreography, orchestration, and hybrid models in the e-commerce domain.

One contribution of our research is providing a comprehensive understanding of how to use multiple orchestrators and choreographies to design a hybrid model. These insights can be applied more broadly beyond the e-commerce domain, as researchers and practitioners can use them as general concepts for adopting hybrid styles in similar scenarios. Our models and analysis can be helpful to a wide range of professionals, including enterprise users, managers, business analysts, software architects, and developers.

We have provided our models, artifacts, and codes online for researchers and practitioners who are interested. They can use them for reusing or regenerating purposes, while also following the principles of open data and repository available in the appendices section. Our research outcomes will benefit both academic knowledge and software development activities in various industries. They provide more comprehensive evidence for practitioners to choose the appropriate modeling tools and techniques, including when and where to use choreography, orchestration, or a hybrid approach to develop software and increase productivity.

6.4 Limitations and Future Work

In our study, we have used Camunda Platform 8, Camunda Modeler, for modeling microservices. Moreover, we have used “user tasks” for the tasks and only the XOR gateways to design microservices. We did not use other tasks such as service tasks, script task and other types of gateways for our proposed models. We omitted those due to code adaptation complexity in BPMN modeling, which we consider a limitation of our research.

Our design models are deployable for the specific version of the Camunda Platform 8. The generated models and XML codes are not deployable to the older version of this tool. which we consider another limitation of our research.

In Chapter 5, we used three metrics - coupling, cohesion, and scalability to determine the best-fit composition approach in e-commerce scenarios. We have identified more metrics in the literature, e.g., modularity and reusability, but we do not analyze our models based on these two metrics. Our designed models are based on the e-commerce domain and for other applications such as in the health care system or an enterprise resource planning applications; for those scenarios are completely different, and analysis results may differ for models based on the number of tasks, types of gateway, and incoming/outgoing flows. Our models are designed from the customer’s end or storefront point of view, and future work can be done from the point of view of administration panels for e-commerce.

We did not test or use our designed models with varying workloads to truly analyze their scalability. This limitation is significant in our study, and future research could focus on time-based performance and workload analysis with full automation.

The five models we designed cover e-commerce end-to-end scenarios, and they do not involve a large number of microservices. This presents a potential challenge to fully validate our study. To address this, a separate study could be conducted using a large number of microservices to determine whether the results differ.

Considering the above limitations, more investigation can be done to analyze the reusability and modularity of microservices composition using other BPMN modeling tools. Furthermore, using service task, send task, receive task, business rule task, script task, and call activity, as tasks for modeling microservices composition, represent a potential improvement of scope in future work.

References

Ali, M. H., Hasan, R., & Benyoucef, M. (2022). Measuring the modeling complexity of microservice choreography and orchestration- The case of e-commerce applications. *Proceedings of The International Conference on Electronic Business (ICEB'22)*, 299–314. Bangkok, Thailand.

Alpers, S., Becker, C., Oberweis, A., & Schuster, T. (2015). Microservice based tool support for business process modelling. *Proceedings of the 2015 IEEE 19th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, EDOCW 2015*, 71–78. https://doi.org/10.1109/EDOCW.2015.32

Artamonov, I. V., & Sukhodolov, A. P. (2018). Cpn tools-based software solution for reliability analysis of processes in microservice environments. *International Journal of Simulation: Systems, Science and Technology*, *19*(6), 56.1-56.8. https://doi.org/10.5013/IJSSST.a.19.06.56

Asri, S. A., Astawa, I. N. G. A., Sunaya, I. G. A. M., Nugroho, I. M. R. A., & Setiawan, W. (2022). Implementation of Asynchronous Microservices Architecture on Smart Village Application. *International Journal on Advanced Science, Engineering and Information Technology*, *12*(3), 1236–1243. https://doi.org/10.18517/ijaseit.12.3.13897

Asrowardi, I., Putra, S. D., & Subyantoro, E. (2020). Designing microservice architectures for scalability and reliability in e-commerce. *Journal of Physics: Conference Series*, *1450*(1). https://doi.org/10.1088/1742-6596/1450/1/012077

Aydin, S., & Çebi, C. B. (2022). Comparison of Choreography vs Orchestration Based Saga Patterns in Microservices. *International Conference on Electrical, Computer and Energy Technologies (ICECET)*, (July), 20–22. https://doi.org/10.1109/ICECET55527.2022.9872665

Baškarada, S., Nguyen, V., & Koronios, A. (2018). Architecting Microservices: Practical Opportunities and Challenges. *Journal of Computer Information Systems*, *00*(00), 1–9. https://doi.org/10.1080/08874417.2018.1520056

Bogner, J., Zimmermann, A., & Wagner, S. (2018). Analyzing the relevance of SOA patterns for microservice-based systems. *CEUR Workshop Proceedings*, *2072*, 9–16.

BPMN coverage. (n.d.). Retrieved June 9, 2022, from https://docs.camunda.io/docs/components/modeler/bpmn/bpmn-coverage/#tasks

Butzin, B., Golatowski, F., & Timmermann, D. (2016). Microservices Approach for the Internet of Things. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* (Vol. 2016-Novem). https://doi.org/10.1109/ETFA.2016.7733707

Camunda Platform 8 Helm deployment. (n.d.). Retrieved August 8, 2021, from https://docs.camunda.io/docs/self-managed/platform-deployment/helm-kubernetes/deploy/

Camunda Platform 8 on Kubernetes. (n.d.). Retrieved July 7, 2022, from https://docs.camunda.io/docs/self-managed/platform-deployment/helm-kubernetes/overview/

Cao, J. (2020). Design on deployment of microservices on container-based cloud platform. *Journal of Physics: Conference Series*, *1624*(6), 4–8. https://doi.org/10.1088/1742-6596/1624/6/062008

Cardoso, J, Mendling, J., Neumann, G., & Reijers, H. A. (2006). A Discourse on Complexity of Process Models (Survey Paper). *BPM 2006 Workshops, Lecture Notes in Computer Science 4103*, 117–128.

Cardoso, Jorge, Vanderfeesten, I., & Reijers, H. A. (2010). Computing coupling for business process models.

Cerny, T., Donahoo, M. J., & Trnka, M. (2018). Contextual understanding of microservice architecture: Current and Future Directions. *ACM SIGAPP Applied Computing Review*, *17*(4), 29–45. https://doi.org/10.1145/3183628.3183631

Choi, S. W., & Kim, S. D. (2008). A quality model for evaluating reusability of services in SOA. *Proceedings - 10th IEEE Joint Conference on E-Commerce Technology and the 5th Enterprise Computing, E-Commerce and E-Services, CEC 2008 and EEE 2008*, 293–298. https://doi.org/10.1109/CECandEEE.2008.134

Christudas, B. (2019). Practical Microservices Architectural Patterns. In *Practical Microservices Architectural Patterns*. https://doi.org/10.1007/978-1-4842-4501-9

Dai, F., Mo, Q., Qiang, Z., Huang, B., Kou, W., & Yang, H. (2020). A Choreography Analysis Approach for Microservice Composition in Cyber-Physical-Social Systems. *IEEE Access*, *8*, 53215–53222. https://doi.org/10.1109/ACCESS.2020.2980891

de Castro, L. F. S., & Rigo, S. (2023). *Relating Edge Computing and Microservices by means of Architecture Approaches and Features, Orchestration, Choreography, and Offloading: A Systematic Literature Review*. 1–40. https://doi.org/https://doi.org/10.48550/arXiv.2301.07803

Deploy your first diagram. (n.d.). Retrieved September 7, 2022, from https://docs.camunda.io/docs/components/modeler/desktop-modeler/connect-to-camunda-cloud/

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices : Yesterday , Today , and Tomorrow. *Springer International Publishing*, 195–216. https://doi.org/https://doi.org/10.1007/978-3-319-67425-4_12

Exclusive gateway. (n.d.). Retrieved August 7, 2022, from https://docs.camunda.io/docs/components/modeler/bpmn/exclusive-gateways/

Filho, A. H., Ribeiro, R. C., do Prado, H. A., Ferneda, E., & Thalheimer, J. M. (2021). A lightweight microservice-oriented platform for development of intelligent agent-based enterprise applications. *ICAART 2021 - Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, *1*(Icaart), 376–383. https://doi.org/10.5220/0010311003760383

Francesco, P. Di, Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, *150*, 77–97. https://doi.org/https://doi.org/10.1016/j.jss.2019.01.001

Goetsch, K. (2017). *Microservices for Modern Commerce*. O'Reilly Media, Inc.

Hamidehkhan, P., & Wurster, M. (2019). Analysis and Evaluation of Modeling and Composition Languages for Microservices. *OPUS - Publication Server of the University of Stuttgart*. Retrieved from https://elib.uni-stuttgart.de/handle/11682/10336

Hasselbring, W., & Steinacker, G. (2017). Microservice architectures for scalability, agility and reliability in e-commerce. *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*, 243–246.

Kappagantula, S. (2019). What Is Microservices – Introduction To Microservice Architecture. Retrieved June 18, 2019, from https://www.edureka.co/blog/what-is-microservices/

Karwatka, P., Gil, M., Grabowski, M., Graf, A., Jędrzejewski, P., Kurzeja, M., … Picho, B. (2018). *Microservices Architecture for eCommerce*.

Kazanaviˇcius, J., & Maˇzeika, D. (2023). Evaluation of microservice communication while decomposing monoliths. *Computing and Informatics*, *42*(1), 1–36. https://doi.org/10.31577/cai_2023_1_1

Khlif, W., Makni, L., Zaaboub, N., & Ben-Abdallah, H. (2009). Quality metrics for business process modeling. *Proceedings of the 9th WSEAS International Conference on Applied Computer Science, ACS '09*, 195–200.

Koschel, A., & Schulze, C. (2022). Towards the Implementation of Workflows in a Microservices Architecture for Insurance Companies The Coexistence of Orchestration and Choreography. *In SERVICE COMPUTATION 2022, The Fourteenth International Conference on Advanced Service Computing*, (c), 1–5.

Koyya, K. M., & Muthukumar, B. (2022). A Survey of Saga Frameworks for Distributed Transactions in Event-driven Microservices. *Third International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, 1–6. https://doi.org/10.1109/ICSTCEE56972.2022.10099533

Lewis, J., & Fowler, M. (2014). Microservices Guide. Retrieved January 20, 2020, from https://martinfowler.com/microservices/

Manual installation on local machine. (n.d.). Retrieved September 9, 2021, from https://docs.camunda.io/docs/self-managed/platform-deployment/manual/

Mark, R. (2016). Microservices Architecture vs. Service-Oriented Architecture. In *The Open Group Standards*. https://doi.org/10.1109/MS.2016.64

Megargel, A., Poskitt, C. M., & Shankararaman, V. (2021). Microservices Orchestration vs. Choreography: A Decision Framework. *Proceedings - 2021 IEEE 25th International Enterprise Distributed Object Computing Conference*, *EDOC 2021*, 134–141. https://doi.org/10.1109/EDOC52215.2021.00024

Monteiro, D., Gadelha, R., Maia, P. H. M., Rocha, L. S., & Mendonça, N. C. (2018). Beethoven: An event-driven lightweight platform for microservice orchestration. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11048 LNCS*, 191–199. https://doi.org/10.1007/978-3-030-00761-4_13

Müller, M. (2019). Consistency and Autonomy in the Microservice Architecture. *(IJACSA) International Journal of Advanced Computer Science and Applications, 9 (8)*, 1–10.

Nadeem, A., & Malik, M. (2022). A Case for Microservices Orchestration Using Workflow Engines. *ArXiv Computer Science*. https://doi.org/10.48550/arXiv.2204.07210

Oberhauser, R., & Stigler, S. (2017). Microflows: Enabling agile business process modeling to orchestrate semantically-annotated microservices. *BMSD 2017 - Proceedings of the 7th International Symposium on Business Modeling and Software Design*, 19–28. https://doi.org/10.5220/0006527100190028

Ortiz, J., Torres, V., & Valderas, P. (2020). Characterization of bottom-up microservice composition evolution. An approach based on the choreography of BPMN fragments. *CEUR Workshop Proceedings*, *2716*, 101–114.

Pahl, C., & Jamshidi, P. (2016). Microservices: A Systematic Mapping Study. *6th International Conference on Cloud Computing and Services Science (CLOSER 2016)*, 137–146. https://doi.org/10.5220/0005785501370146

Panichella, S., Rahman, M. I., & Taibi, D. (2021). Structural Coupling for Microservices. *International Conference on Cloud Computing and Services Science, CLOSER - Proceedings*, *2021-April*(Closer), 280–287. https://doi.org/10.5220/0010481902800287

Pontarolli, R. P., Bigheti, J. A., Fernandes, M. M., Domingues, F. O., Risso, S. L., & Godoy, E. P. (2020). Microservice Orchestration for Process Control in Industry 4.0. *2020 IEEE International Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2020 - Proceedings*, 245–249. https://doi.org/10.1109/MetroInd4.0IoT48571.2020.9138228

Rademacher, F., Sachweh, S., & Zündorf, A. (2018). Towards a UML profile for domain-driven design of microservice architectures. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10729 LNCS*(i), 230–245. https://doi.org/10.1007/978-3-319-74781-1_17

Rademacher, F., Sorgalla, J., Sachweh, S., & Zuendorf, A. (2019). A Model-driven Workflow for Distributed Microservice Development. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*.

Rademacher, F., Sorgalla, J., Sachweh, S., & Zündorf, A. (2018). *Towards a Viewpoint-specific Metamodel for Model-driven Development of Microservice Architecture*. (ii). Retrieved from http://arxiv.org/abs/1804.09948

Richardson, C. (2020). Microservice Architecture. Retrieved February 12, 2020, from https://microservices.io

Rudrabhatla, C. K. (2018). Comparison of Event Choreography and Orchestration Techniques in Microservice Architecture. *International Journal of Advanced Computer Science and Applications*, *9*(8), 18–22. https://doi.org/10.14569/IJACSA.2018.090804

Santos, N., Rodrigues, H., Ferreira, N., & Machado, R. J. (2019). Inputs from a Model-Based Approach Towards the Specification of Microservices Logical Architectures: An Experience Report. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11915 LNCS*(November), 473–488. https://doi.org/10.1007/978-3-030-35333-9_33

Savchenko, D., Radchenko, G., Hynninen, T., & Taipale, O. (2018). Microservice Test Process : Design and Implementation. *International Journal on Information Technologies & Security*, *10*, 13–24.

Scattone, F. F., & Braghetto, K. R. (2018). A Microservices Architecture for Distributed Complex Event Processing in Smart Cities. *IEEE 37th International Symposium on Reliable Distributed Systems Workshops (SRDSW)*, 6–9. https://doi.org/10.1109/SRDSW.2018.00012

Shuo, Z., Rui, Y., Yin, X., Yan, Z., Wei, Z., Yeteng, A., … Liyang, X. (2023). Research on the application of service choreography in the intelligent customer service system. *IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms, EEBDA*, 1238–1242. https://doi.org/10.1109/EEBDA56825.2023.10090735

Singhal, N., Sakthivel, U., & Raj, P. (2019a). Efficient hybrid research for QoS-aware microservice composition. *International Journal of Recent Technology and Engineering*, *8*(2), 5251–5255. https://doi.org/10.35940/ijrte.B1055.078219

Singhal, N., Sakthivel, U., & Raj, P. (2019b). Selection Mechanism of Micro-Services Orchestration Vs. Choreography. *International Journal of Web & Semantic Technology*, *10*(1), 01–13. https://doi.org/10.5121/ijwest.2019.10101

Sorgalla, J., Rademacher, F., Sachweh, S., & Zündorf, A. (2018). On collaborative model-driven development of microservices. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 596–603.

https://doi.org/10.1007/978-3-030-04771-9_45

Stutz, A., Fay, A., Barth, M., & Maurmaier, M. (2020). Choreographies in Microservice-Based Automation Architectures: Next Level of Flexibility for Industrial Cyber-Physical Systems. *Proceedings - 2020 IEEE Conference on Industrial Cyberphysical Systems, ICPS 2020*, 411–416. https://doi.org/10.1109/ICPS48405.2020.9274719

Taibi, Davide, Lenarduzzi, V., & Pahl, C. (2017). Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. *Association for Computing Machinery*.

Taibi, Davide, Lenarduzzi, V., & Pahl, C. (2018). Architectural patterns for microservices: A systematic mapping study. *CLOSER 2018 - Proceedings of the 8th International Conference on Cloud Computing and Services Science*, (CLOSER), 221–232.

The Camunda Platform 7 Manual. (n.d.). Retrieved July 8, 2022, from https://docs.camunda.org/manual/7.18/

Valderas, P, Torres, V., & Pelechano, V. (2020). Supporting a Hybrid Composition of Microservices. The EUCalipTool Platform. *Journal of Software Engineering Research and Development*, (8:1). https://doi.org/10.5753/jserd.2019.457

Valderas, Pedro, Torres, V., & Pelechano, V. (2020). A microservice composition approach based on the choreography of BPMN fragments. *Information and Software Technology*, *127*(July 2019). https://doi.org/10.1016/j.infsof.2020.106370

Vanderfeesten, I., Cardoso, J., Reijers, H. A., & Aalst, W. Van Der. (2007). Quality Metrics for Business Process Models. *BPM and Workflow Handbook*, (August 2015), 179–190. https://doi.org/10.1007/978-3-540-89224-3

Vanderfeesten, I., Reijers, H. A., & van der Aalst, W. M. P. (2007). Evaluating workflow process designs using cohesion and coupling metrics. *Computers in Industry*, *59*(5), 420–437. https://doi.org/10.1016/j.compind.2007.12.007

Venable, J. R., Pries-heje, J., & Baskerville, R. (2017). Choosing a Design Science Research Methodology. *Australasian Conference on Information Systems*, 1–11.

Volynsky, E., Mehmed, M., & Krusche, S. (2022). Architect: A Framework for the Migration to Microservices. *International Conference on Computing, Electronics and Communications Engineering. ICCECE 2022*, 71–76. https://doi.org/10.1109/iCCECE55162.2022.9875096

What Is Service-Oriented Architecture? (n.d.). Retrieved October 11, 2021, from https://www.ibm.com/cloud/learn/soa#toc-benefits-o-YQlLJ50n

Wu, M., Ding, X., & Hou, R. (2019). Design and Implementation of B2B E-Commerce Platform Based on Microservices Architecture. *Proceedings of the 2nd International Conference on Computer Science and Software Engineering.*, (1), 30–34.

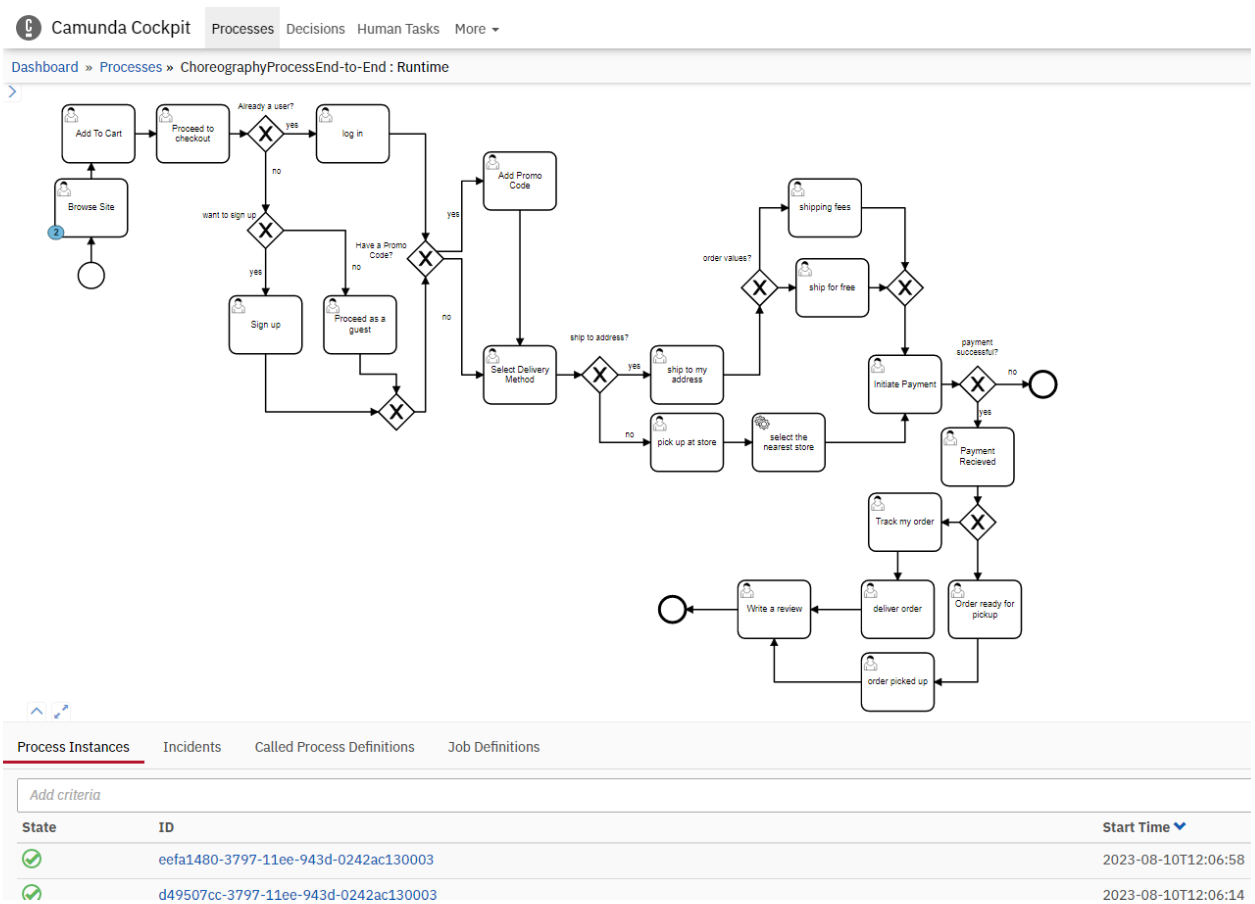
Yaqin, M. A., Sarno, R., & Fauzan, A. C. (2017). Scalability measurement of business process model using business processes similarity and complexity. *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, *2017-Decem*(September), 1–7. https://doi.org/10.1109/EECSI.2017.8239129

Zhang, X., Liang, D., Zhang, Y., & Liu, Z. (2023). An API Framework for Flexible and Lightweight Microservice Composition. *8th International Conference on Computer and Communication Systems (ICCCS)*, 701–707. https://doi.org/10.1109/icccs57501.2023.10150530

Appendices

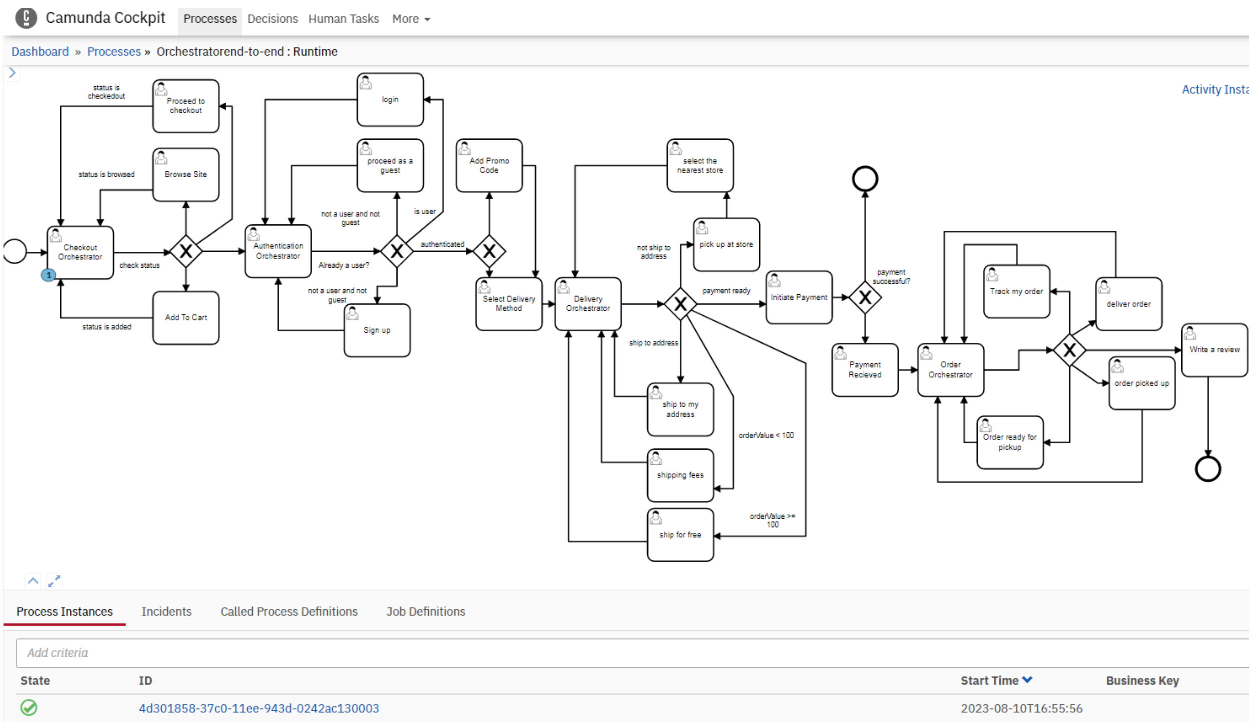
Appendix A. Simulated flow of Choreography end-to-end e-Commerce scenarios on Camunda Cockpit

The end-to-end e-Commerce scenarios flow token was generated and started from the “Browse Site” activity and ended after the “Write a review” task.



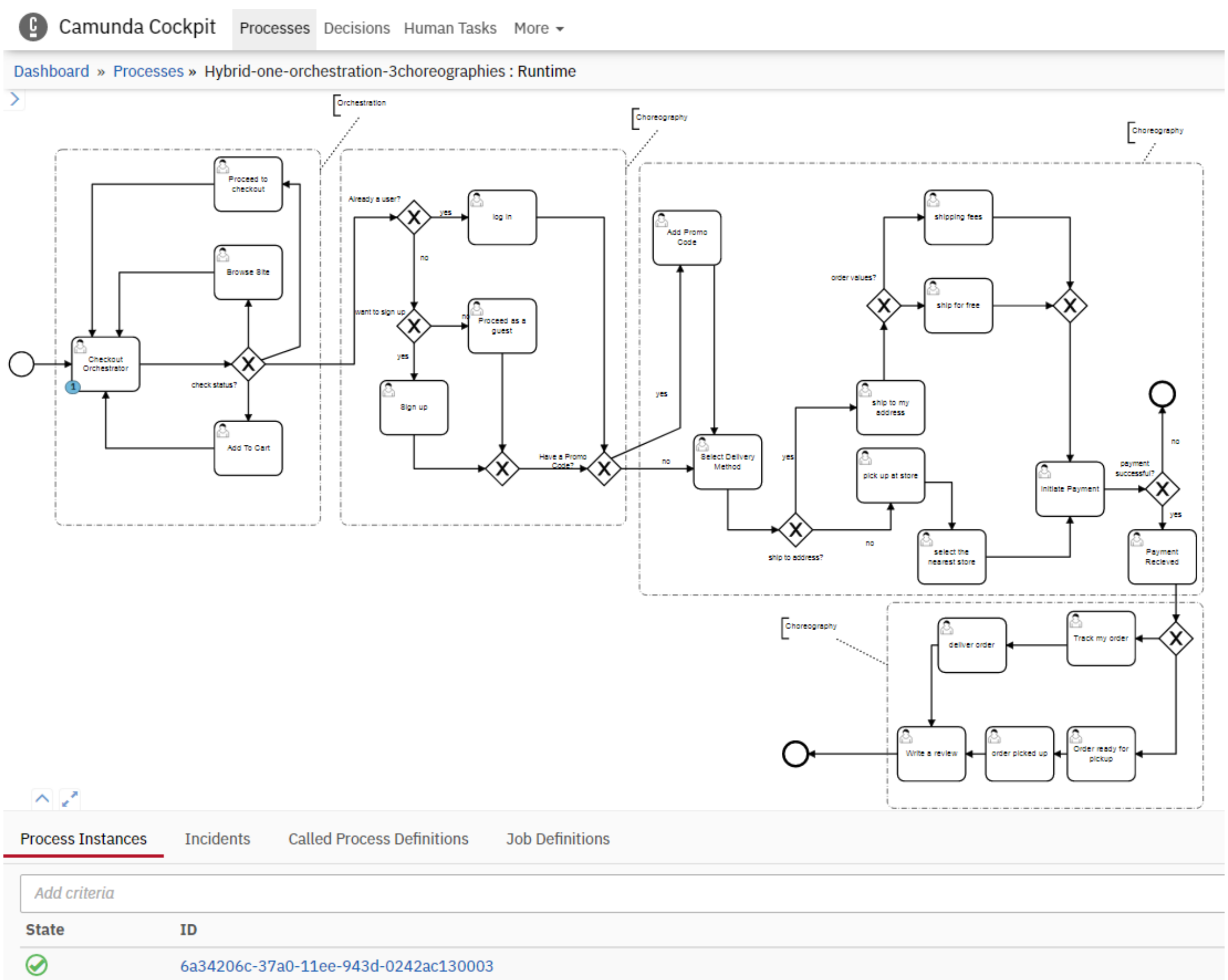
Appendix B. Simulated flow of Orchestration end-to-end e-Commerce scenarios on Camunda Cockpit

The orchestration-based end-to-end e-Commerce scenarios flow token generated and started from the “Checkout Orchestrator” process and ended after the “Write a review” activity**.**



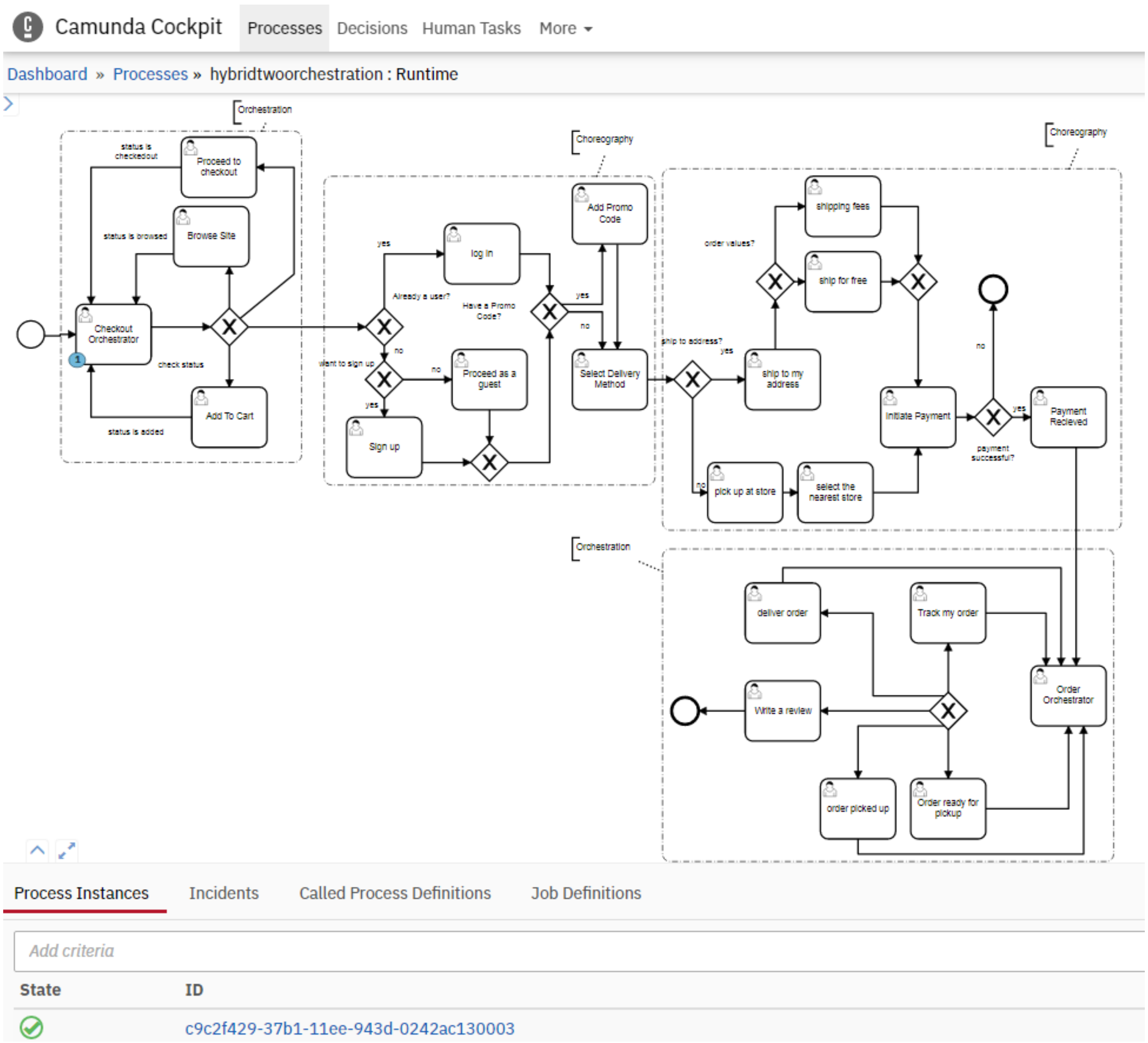
Appendix C. Simulated flow of Hybrid (A) (One-Orchestration-Three-Choreographies) end-to-end e-Commerce scenarios on Camunda Cockpit

In this hybrid end-to-end e-Commerce workflow, we used one orchestration and three choreographies sub-process where the flow token generated from the "Checkout Orchestrator" task and ended after the "Write a review" activity.



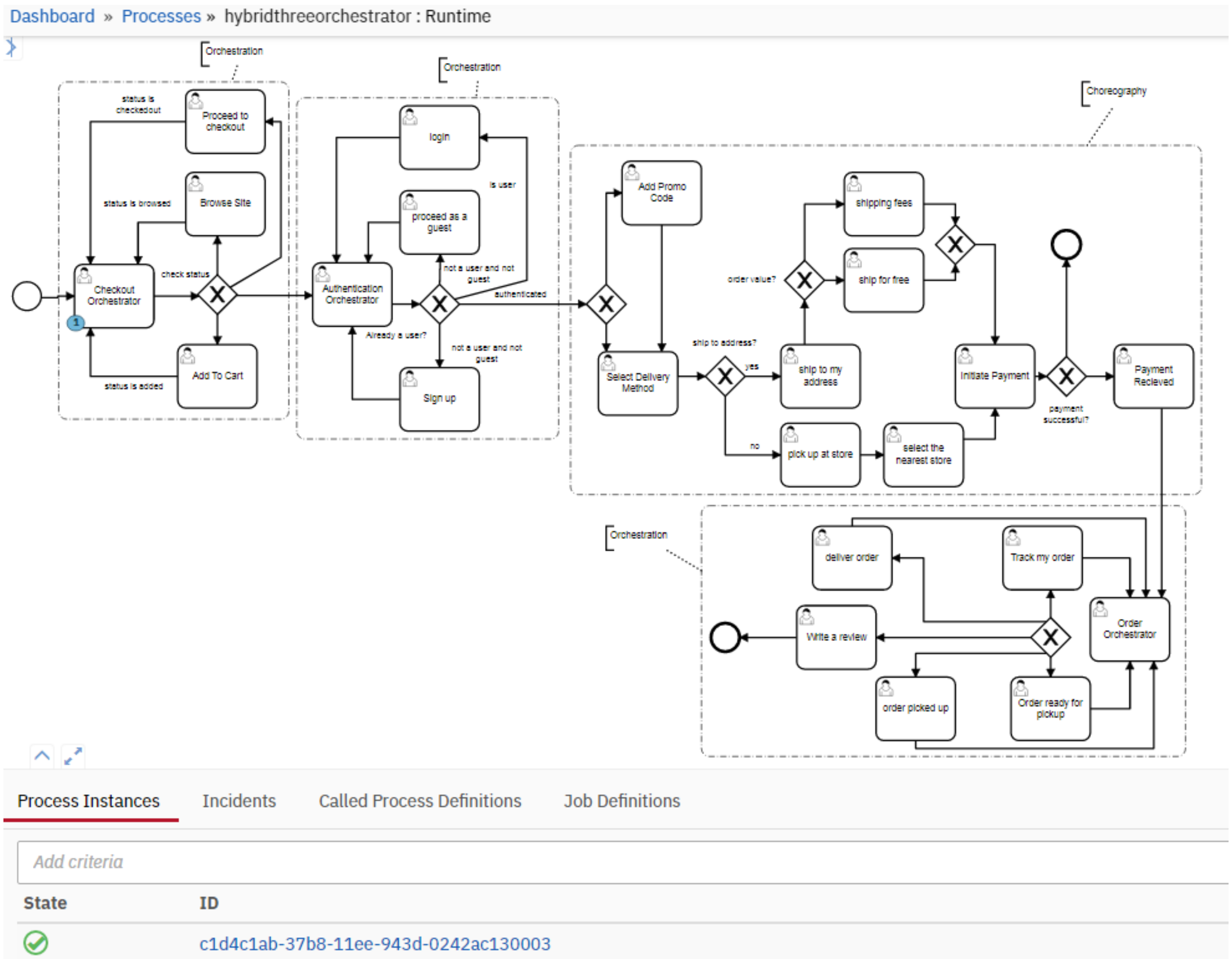
Appendix D. Simulated flow of Hybrid (B) (Two-Orchestrations-Two-Choreographies) end-to-end e-Commerce scenarios on Camunda Cockpit

In this hybrid end-to-end e-Commerce workflow, we used two orchestrations and two choreographies sub-process where the flow token in the deployment phase generated from the “Checkout Orchestrator” task and ended in orchestration sub-process after the “Write a review” task.



Appendix E. Simulated flow of Hybrid (C) (Three-Orchestrations-One-Choreography) end-to-end e-Commerce scenarios on Camunda Cockpit

In this hybrid end-to-end e-Commerce workflow, we used three orchestrations and one choreography sub-process after two orchestrations sub-process where the flow token in deployment phase generated and started from the "Checkout Orchestrator" task and ended in orchestration sub-process after the "Write a review" task.



Appendix F. XML code for Hybrid (A) (One-Orchestration-Three-Choreographies) end-to-end e-Commerce scenarios on Camunda Cockpit

The repository for this work is available at: https://github.com/rhasa044/microservices-composition-hybrid-style. It contains artifacts and code generated in the microservices composition hybrid style study. Researchers interested in replicating or building upon this research can access the repository, ensuring transparency and reproducibility of findings.

Hybrid (A) Toggle XML


```xml
<?xml version="1.0" encoding="UTF-8"?>
<bpmn:definitions xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
xmlns:dc="http://www.omg.org/spec/DD/20100524/DC" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:di="http://www.omg.org/spec/DD/20100524/DI" id="Definitions_1w39o4q"
targetNamespace="http://bpmn.io/schema/bpmn" exporter="Camunda Modeler" exporterVersion="5.13.0">
  <bpmn:process id="Hybrid-one-orchestration-3choreographies" isExecutable="true">
    <bpmn:startEvent id="StartEvent_1">
      <bpmn:outgoing>SequenceFlow_1u05py2</bpmn:outgoing>
    </bpmn:startEvent>
```

```
    <bpmn:exclusiveGateway id="ExclusiveGateway_12207ye" name="Already a user?">
      <bpmn:incoming>SequenceFlow_1nj1l69</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0wba7ds</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_1cv6x8z</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_0wba7ds" name="yes" sourceRef="ExclusiveGateway_12207ye"
targetRef="Task_0dcjzef">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${isUser==
true}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:exclusiveGateway id="ExclusiveGateway_1wnf1j3" name="want to sign up">
      <bpmn:incoming>SequenceFlow_1cv6x8z</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1u6tvcu</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_0mcl6gf</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_1cv6x8z" name="no" sourceRef="ExclusiveGateway_12207ye"
targetRef="ExclusiveGateway_1wnf1j3">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${isUser==
false}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_1u6tvcu" name="yes" sourceRef="ExclusiveGateway_1wnf1j3"
targetRef="Task_009ozkj">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${signup==
true}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_0mcl6gf" name="no" sourceRef="ExclusiveGateway_1wnf1j3"
targetRef="Task_0tlg9au">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${signup==
false}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:exclusiveGateway id="ExclusiveGateway_01sk276">
      <bpmn:incoming>SequenceFlow_11s4zb3</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_0r22vdd</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_02d3c5k</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_11s4zb3" sourceRef="Task_009ozkj"
targetRef="ExclusiveGateway_01sk276" />
    <bpmn:sequenceFlow id="SequenceFlow_0r22vdd" sourceRef="Task_0tlg9au"
targetRef="ExclusiveGateway_01sk276" />
    <bpmn:exclusiveGateway id="ExclusiveGateway_0w45w04" name="Have a Promo Code?">
      <bpmn:incoming>SequenceFlow_00dm4px</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_02d3c5k</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_13bm0fm</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_0t2wvrg</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_00dm4px" sourceRef="Task_0dcjzef"
targetRef="ExclusiveGateway_0w45w04" />
    <bpmn:sequenceFlow id="SequenceFlow_02d3c5k" sourceRef="ExclusiveGateway_01sk276"
targetRef="ExclusiveGateway_0w45w04" />
    <bpmn:sequenceFlow id="SequenceFlow_13bm0fm" name="yes" sourceRef="ExclusiveGateway_0w45w04"
targetRef="Task_06mbo2b">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${havePromoCode ==
true}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_0t2wvrg" name="no" sourceRef="ExclusiveGateway_0w45w04"
targetRef="Task_1qjbvnf">
```

```
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${havePromoCode ==
false}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_01woab4" sourceRef="Task_06mbo2b" targetRef="Task_1qjbvnf"
/>
    <bpmn:exclusiveGateway id="ExclusiveGateway_1ji0df2" name="ship to address?">
      <bpmn:incoming>SequenceFlow_1rqoomg</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_11ejla7</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_16uenhz</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_1rqoomg" sourceRef="Task_1qjbvnf"
targetRef="ExclusiveGateway_1ji0df2" />
    <bpmn:sequenceFlow id="SequenceFlow_11ejla7" name="yes" sourceRef="ExclusiveGateway_1ji0df2"
targetRef="Task_1ky7hfx">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${shipToAddress==
true}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:exclusiveGateway id="ExclusiveGateway_1blk8rt" name="order values?">
      <bpmn:incoming>SequenceFlow_1tjh507</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0yuz5o0</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_1eag7yq</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_1tjh507" sourceRef="Task_1ky7hfx"
targetRef="ExclusiveGateway_1blk8rt" />
    <bpmn:sequenceFlow id="SequenceFlow_0yuz5o0" sourceRef="ExclusiveGateway_1blk8rt"
targetRef="Task_07ki0pb">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${orderValue &lt;
100}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_1eag7yq" sourceRef="ExclusiveGateway_1blk8rt"
targetRef="Task_1xj8x4l">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${orderValue &gt;=
100}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:exclusiveGateway id="ExclusiveGateway_10d4tn7">
      <bpmn:incoming>SequenceFlow_1k5xysm</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_07nvgk7</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1olp77n</bpmn:outgoing>
    </bpmn:exclusiveGateway>
    <bpmn:sequenceFlow id="SequenceFlow_1k5xysm" sourceRef="Task_07ki0pb"
targetRef="ExclusiveGateway_10d4tn7" />
    <bpmn:sequenceFlow id="SequenceFlow_07nvgk7" sourceRef="Task_1xj8x4l"
targetRef="ExclusiveGateway_10d4tn7" />
    <bpmn:sequenceFlow id="SequenceFlow_1olp77n" sourceRef="ExclusiveGateway_10d4tn7"
targetRef="Task_1ub5u9c" />
    <bpmn:sequenceFlow id="SequenceFlow_16uenhz" name="no" sourceRef="ExclusiveGateway_1ji0df2"
targetRef="Task_08mptt4">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${shipToAddress==
false}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_14gbbiz" sourceRef="Task_08mptt4" targetRef="Task_085t7lv" />
    <bpmn:sequenceFlow id="SequenceFlow_0eg7ulc" sourceRef="Task_085t7lv" targetRef="Task_1ub5u9c" />
    <bpmn:exclusiveGateway id="ExclusiveGateway_1scfuq2" name="payment successful?">
      <bpmn:incoming>SequenceFlow_1b9pd6i</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0x07e52</bpmn:outgoing>
      <bpmn:outgoing>SequenceFlow_1insc67</bpmn:outgoing>
```

```
    &lt;/bpmn:exclusiveGateway&gt;
    <bpmn:sequenceFlow id="SequenceFlow_1b9pd6i" sourceRef="Task_1ub5u9c"
targetRef="ExclusiveGateway_1scfuq2" />
    <bpmn:sequenceFlow id="SequenceFlow_0x07e52" name="yes" sourceRef="ExclusiveGateway_1scfuq2"
targetRef="Task_0iscgei">
      &lt;bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"&gt;${paymentSuccessful==
true}&lt;/bpmn:conditionExpression&gt;
    &lt;/bpmn:sequenceFlow&gt;
    &lt;bpmn:endEvent id="EndEvent_02xnrs6"&gt;
      &lt;bpmn:incoming&gt;SequenceFlow_1insc67&lt;/bpmn:incoming&gt;
    &lt;/bpmn:endEvent&gt;
    <bpmn:sequenceFlow id="SequenceFlow_1insc67" name="no" sourceRef="ExclusiveGateway_1scfuq2"
targetRef="EndEvent_02xnrs6">
      &lt;bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"&gt;${paymentSuccessful==
false}&lt;/bpmn:conditionExpression&gt;
    &lt;/bpmn:sequenceFlow&gt;
    &lt;bpmn:exclusiveGateway id="ExclusiveGateway_0vq37ec"&gt;
      &lt;bpmn:incoming&gt;SequenceFlow_11v1trp&lt;/bpmn:incoming&gt;
      &lt;bpmn:outgoing&gt;SequenceFlow_0toxbj8&lt;/bpmn:outgoing&gt;
      &lt;bpmn:outgoing&gt;SequenceFlow_0so8lpt&lt;/bpmn:outgoing&gt;
    &lt;/bpmn:exclusiveGateway&gt;
    &lt;bpmn:endEvent id="EndEvent_0p3yr1b"&gt;
      &lt;bpmn:incoming&gt;SequenceFlow_08sc29q&lt;/bpmn:incoming&gt;
    &lt;/bpmn:endEvent&gt;
    <bpmn:sequenceFlow id="SequenceFlow_08sc29q" sourceRef="Task_18tj109"
targetRef="EndEvent_0p3yr1b" />
    &lt;bpmn:sequenceFlow id="SequenceFlow_1u05py2" sourceRef="StartEvent_1" targetRef="Task_1n39hpz" /&gt;
    <bpmn:sequenceFlow id="SequenceFlow_11v1trp" sourceRef="Task_0iscgei"
targetRef="ExclusiveGateway_0vq37ec" />
    <bpmn:sequenceFlow id="SequenceFlow_0toxbj8" sourceRef="ExclusiveGateway_0vq37ec"
targetRef="Task_0h9nja0">
      &lt;bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"&gt;${shipToAddress==
true}&lt;/bpmn:conditionExpression&gt;
    &lt;/bpmn:sequenceFlow&gt;
    <bpmn:sequenceFlow id="SequenceFlow_0so8lpt" sourceRef="ExclusiveGateway_0vq37ec"
targetRef="Task_09kpo8l">
      &lt;bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"&gt;${shipToAddress==
false}&lt;/bpmn:conditionExpression&gt;
    &lt;/bpmn:sequenceFlow&gt;
    &lt;bpmn:sequenceFlow id="SequenceFlow_1vks2be" sourceRef="Task_09kpo8l" targetRef="Task_1dzlnt5" /&gt;
    &lt;bpmn:sequenceFlow id="SequenceFlow_0gljw3v" sourceRef="Task_0h9nja0" targetRef="Task_037zc08" /&gt;
    &lt;bpmn:sequenceFlow id="SequenceFlow_0ljigac" sourceRef="Task_037zc08" targetRef="Task_18tj109" /&gt;
    &lt;bpmn:sequenceFlow id="SequenceFlow_1jem9ox" sourceRef="Task_1dzlnt5" targetRef="Task_18tj109" /&gt;
    <bpmn:sequenceFlow id="SequenceFlow_1011qdu" sourceRef="Task_1n39hpz"
targetRef="ExclusiveGateway_0ro5yum" />
    &lt;bpmn:exclusiveGateway id="ExclusiveGateway_0ro5yum" name="check status?"&gt;
      &lt;bpmn:incoming&gt;SequenceFlow_1011qdu&lt;/bpmn:incoming&gt;
      &lt;bpmn:outgoing&gt;SequenceFlow_16d1x3e&lt;/bpmn:outgoing&gt;
      &lt;bpmn:outgoing&gt;SequenceFlow_1r0ubpp&lt;/bpmn:outgoing&gt;
      &lt;bpmn:outgoing&gt;SequenceFlow_1nj1l69&lt;/bpmn:outgoing&gt;
      &lt;bpmn:outgoing&gt;SequenceFlow_112ikiw&lt;/bpmn:outgoing&gt;
    &lt;/bpmn:exclusiveGateway&gt;
    <bpmn:sequenceFlow id="SequenceFlow_16d1x3e" sourceRef="ExclusiveGateway_0ro5yum"
targetRef="Task_15o7fct">
      &lt;bpmn:conditionExpression xsi:type="bpmn:tFormalExpression"&gt;${status ==
"new"}&lt;/bpmn:conditionExpression&gt;
```

```xml
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_0dlosaf" sourceRef="Task_03jcel6" targetRef="Task_1n39hpz" />
    <bpmn:sequenceFlow id="SequenceFlow_1r0ubpp" sourceRef="ExclusiveGateway_0ro5yum"
targetRef="Task_03jcel6">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${status ==
"browsed"}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_1wqj52d" sourceRef="Task_04vusa4" targetRef="Task_1n39hpz"
/>
    <bpmn:sequenceFlow id="SequenceFlow_15kdgkc" sourceRef="Task_15o7fct" targetRef="Task_1n39hpz" />
    <bpmn:sequenceFlow id="SequenceFlow_1nj1l69" sourceRef="ExclusiveGateway_0ro5yum"
targetRef="ExclusiveGateway_12207ye">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${status ==
"checkedout"}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:sequenceFlow id="SequenceFlow_112ikiw" sourceRef="ExclusiveGateway_0ro5yum"
targetRef="Task_04vusa4">
      <bpmn:conditionExpression xsi:type="bpmn:tFormalExpression">${status ==
"added"}</bpmn:conditionExpression>
    </bpmn:sequenceFlow>
    <bpmn:userTask id="Task_04vusa4" name="Proceed to checkout">
      <bpmn:incoming>SequenceFlow_112ikiw</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1wqj52d</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_15o7fct" name="Browse Site">
      <bpmn:incoming>SequenceFlow_16d1x3e</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_15kdgkc</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_1n39hpz" name="Checkout Orchestrator">
      <bpmn:incoming>SequenceFlow_1u05py2</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_0dlosaf</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_1wqj52d</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_15kdgkc</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1011qdu</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_0dcjzef" name="log in">
      <bpmn:incoming>SequenceFlow_0wba7ds</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_00dm4px</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_03jcel6" name="Add To Cart">
      <bpmn:incoming>SequenceFlow_1r0ubpp</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0dlosaf</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_009ozkj" name="Sign up">
      <bpmn:incoming>SequenceFlow_1u6tvcu</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_11s4zb3</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_0tlg9au" name="Proceed as a guest">
      <bpmn:incoming>SequenceFlow_0mcl6gf</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0r22vdd</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_06mbo2b" name="Add Promo Code">
      <bpmn:incoming>SequenceFlow_13bm0fm</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_01woab4</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_1qjbvnf" name="Select Delivery Method">
```

```xml
      <bpmn:incoming>SequenceFlow_0t2wvrg</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_01woab4</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1rqoomg</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_085t7lv" name="select the nearest store">
      <bpmn:incoming>SequenceFlow_14gbbiz</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0eg7ulc</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_08mptt4" name="pick up at store">
      <bpmn:incoming>SequenceFlow_16uenhz</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_14gbbiz</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_1ky7hfx" name="ship to my address">
      <bpmn:incoming>SequenceFlow_11ejla7</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1tjh507</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_1ub5u9c" name="Initiate Payment">
      <bpmn:incoming>SequenceFlow_1olp77n</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_0eg7ulc</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1b9pd6i</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_1xj8x4l" name="ship for free">
      <bpmn:incoming>SequenceFlow_1eag7yq</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_07nvgk7</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_07ki0pb" name="shipping fees">
      <bpmn:incoming>SequenceFlow_0yuz5o0</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1k5xysm</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_09kpo8l" name="Order ready for pickup">
      <bpmn:incoming>SequenceFlow_0so8lpt</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1vks2be</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_0iscgei" name="Payment Recieved">
      <bpmn:incoming>SequenceFlow_0x07e52</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_11v1trp</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_0h9nja0" name="Track my order">
      <bpmn:incoming>SequenceFlow_0toxbj8</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0gljw3v</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_1dzlnt5" name="order picked up">
      <bpmn:incoming>SequenceFlow_1vks2be</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_1jem9ox</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_18tj109" name="Write a review">
      <bpmn:incoming>SequenceFlow_0ljigac</bpmn:incoming>
      <bpmn:incoming>SequenceFlow_1jem9ox</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_08sc29q</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:userTask id="Task_037zc08" name="deliver order">
      <bpmn:incoming>SequenceFlow_0gljw3v</bpmn:incoming>
      <bpmn:outgoing>SequenceFlow_0ljigac</bpmn:outgoing>
    </bpmn:userTask>
    <bpmn:group id="Group_1bfu1nn" categoryValueRef="CategoryValue_0cw5eur" />
    <bpmn:textAnnotation id="TextAnnotation_1ujpp1t">
```

```
      <bpmn:text>Orchestration</bpmn:text>
    </bpmn:textAnnotation>
    <bpmn:association id="Association_16ygme5" sourceRef="Group_1bfu1nn"
targetRef="TextAnnotation_1ujpp1t" />
    <bpmn:group id="Group_1v4zn2e" categoryValueRef="CategoryValue_0yqsp81" />
    <bpmn:textAnnotation id="TextAnnotation_1tla79q">
      <bpmn:text>Choreography</bpmn:text>
    </bpmn:textAnnotation>
    <bpmn:association id="Association_1356ifj" sourceRef="Group_1v4zn2e"
targetRef="TextAnnotation_1tla79q" />
    <bpmn:group id="Group_11llgzu" categoryValueRef="CategoryValue_0hw1mha" />
    <bpmn:textAnnotation id="TextAnnotation_06dntmg">
      <bpmn:text>Choreography</bpmn:text>
    </bpmn:textAnnotation>
    <bpmn:association id="Association_0xxc6se" sourceRef="Group_11llgzu"
targetRef="TextAnnotation_06dntmg" />
    <bpmn:group id="Group_153xl6j" categoryValueRef="CategoryValue_1flj3c3" />
    <bpmn:textAnnotation id="TextAnnotation_0dnryjb">
      <bpmn:text>Choreography</bpmn:text>
    </bpmn:textAnnotation>
    <bpmn:association id="Association_0mpzs4g" sourceRef="Group_153xl6j"
targetRef="TextAnnotation_0dnryjb" />
  </bpmn:process>
  <bpmn:message id="Message_0x8q0rf" name="payment recieved" />
  <bpmn:category id="Category_0z4a0z4">
    <bpmn:categoryValue id="CategoryValue_0cw5eur" />
  </bpmn:category>
  <bpmn:category id="Category_1c6l92q">
    <bpmn:categoryValue id="CategoryValue_0yqsp81" />
  </bpmn:category>
  <bpmn:category id="Category_1acgf2t">
    <bpmn:categoryValue id="CategoryValue_0hw1mha" />
  </bpmn:category>
  <bpmn:category id="Category_1wynd3v">
    <bpmn:categoryValue id="CategoryValue_1flj3c3" />
  </bpmn:category>
  <bpmndi:BPMNDiagram id="BPMNDiagram_1">
    <bpmndi:BPMNPlane id="BPMNPlane_1" bpmnElement="Hybrid-one-orchestration-3choreographies">
      <bpmndi:BPMNShape id="TextAnnotation_1tla79q_di" bpmnElement="TextAnnotation_1tla79q">
        <dc:Bounds x="1070" y="100" width="100" height="30" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="TextAnnotation_1ujpp1t_di" bpmnElement="TextAnnotation_1ujpp1t">
        <dc:Bounds x="630" y="80" width="100" height="30" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="TextAnnotation_0dnryjb_di" bpmnElement="TextAnnotation_0dnryjb">
        <dc:Bounds x="1290" y="850" width="100" height="30" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="TextAnnotation_06dntmg_di" bpmnElement="TextAnnotation_06dntmg">
        <dc:Bounds x="1800" y="120" width="100" height="30" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_0ro5yum_di" bpmnElement="ExclusiveGateway_0ro5yum"
isMarkerVisible="true">
        <dc:Bounds x="479" y="451" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="420" y="501" width="68" height="14" />
        </bpmndi:BPMNLabel>
```

```xml
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1yureh0_di" bpmnElement="Task_04vusa4">
        <dc:Bounds x="454" y="171" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1rwivry_di" bpmnElement="Task_15o7fct">
        <dc:Bounds x="454" y="301" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_0hoswjp_di" bpmnElement="Task_1n39hpz">
        <dc:Bounds x="244" y="436" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_0kq87w7_di" bpmnElement="Task_03jcel6">
        <dc:Bounds x="454" y="560" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="_BPMNShape_StartEvent_2" bpmnElement="StartEvent_1">
        <dc:Bounds x="152" y="458" width="36" height="36" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_12207ye_di" bpmnElement="ExclusiveGateway_12207ye"
isMarkerVisible="true">
        <dc:Bounds x="723" y="235" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="651" y="227" width="77" height="14" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_18wbvb0_di" bpmnElement="Task_0dcjzef">
        <dc:Bounds x="828" y="220" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_02ny4ov_di" bpmnElement="Task_0tlg9au">
        <dc:Bounds x="828" y="380" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_1wnf1j3_di" bpmnElement="ExclusiveGateway_1wnf1j3"
isMarkerVisible="true">
        <dc:Bounds x="723" y="395" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="661" y="393" width="74" height="14" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1n8o6hc_di" bpmnElement="Task_009ozkj">
        <dc:Bounds x="698" y="500" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_01sk276_di" bpmnElement="ExclusiveGateway_01sk276"
isMarkerVisible="true">
        <dc:Bounds x="853" y="605" width="50" height="50" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_0w45w04_di" bpmnElement="ExclusiveGateway_0w45w04"
isMarkerVisible="true">
        <dc:Bounds x="1003" y="605" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="932" y="606" width="71" height="27" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1rpbdmb_di" bpmnElement="Task_1qjbvnf">
        <dc:Bounds x="1160" y="580" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_06jw73y_di" bpmnElement="Task_1ky7hfx">
        <dc:Bounds x="1400" y="500" width="100" height="80" />
      </bpmndi:BPMNShape>
```

```xml
      <bpmndi:BPMNShape id="Activity_0v463ho_di" bpmnElement="Task_1ub5u9c">
        <dc:Bounds x="1664" y="620" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1qhemt9_di" bpmnElement="Task_085t7lv">
        <dc:Bounds x="1490" y="720" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_1ji0df2_di" bpmnElement="ExclusiveGateway_1ji0df2"
isMarkerVisible="true">
        <dc:Bounds x="1285" y="695" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="1269.5" y="755" width="81" height="14" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1oux83m_di" bpmnElement="Task_08mptt4">
        <dc:Bounds x="1400" y="600" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_1scfuq2_di" bpmnElement="ExclusiveGateway_1scfuq2"
isMarkerVisible="true">
        <dc:Bounds x="1825" y="635" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="1781" y="616.5" width="58" height="27" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_05x6i88_di" bpmnElement="Task_06mbo2b">
        <dc:Bounds x="1100" y="250" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_0duhxqy_di" bpmnElement="Task_07ki0pb">
        <dc:Bounds x="1500" y="220" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_18ro7kj_di" bpmnElement="Task_1xj8x4l">
        <dc:Bounds x="1500" y="350" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_10d4tn7_di" bpmnElement="ExclusiveGateway_10d4tn7"
isMarkerVisible="true">
        <dc:Bounds x="1689" y="365" width="50" height="50" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="ExclusiveGateway_1blk8rt_di" bpmnElement="ExclusiveGateway_1blk8rt"
isMarkerVisible="true">
        <dc:Bounds x="1415" y="365" width="50" height="50" />
        <bpmndi:BPMNLabel>
          <dc:Bounds x="1362" y="343" width="67" height="14" />
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="EndEvent_02xnrs6_di" bpmnElement="EndEvent_02xnrs6">
        <dc:Bounds x="1832" y="502" width="36" height="36" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1kz11gi_di" bpmnElement="Task_0iscgei">
        <dc:Bounds x="1800" y="720" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_0g24ygr_di" bpmnElement="Task_037zc08">
        <dc:Bounds x="1520" y="850" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_0gs6qat_di" bpmnElement="Task_18tj109">
        <dc:Bounds x="1460" y="1010" width="100" height="80" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Activity_1bb6p7t_di" bpmnElement="Task_0h9nja0">
```

```
      <dc:Bounds x="1710" y="840" width="100" height="80" />
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="Activity_0wdvnjm_di" bpmnElement="Task_1dzlnt5">
      <dc:Bounds x="1590" y="1010" width="100" height="80" />
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="Activity_0b6gvza_di" bpmnElement="Task_09kpo8l">
      <dc:Bounds x="1710" y="1010" width="100" height="80" />
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="ExclusiveGateway_0vq37ec_di" bpmnElement="ExclusiveGateway_0vq37ec"
isMarkerVisible="true">
      <dc:Bounds x="1845" y="855" width="50" height="50" />
    </bpmndi:BPMNShape>
    <bpmndi:BPMNShape id="EndEvent_0p3yr1b_di" bpmnElement="EndEvent_0p3yr1b">
      <dc:Bounds x="1292" y="1032" width="36" height="36" />
    </bpmndi:BPMNShape>
    <bpmndi:BPMNEdge id="Association_1356ifj_di" bpmnElement="Association_1356ifj">
      <di:waypoint x="1060" y="188" />
      <di:waypoint x="1108" y="130" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="Association_16ygme5_di" bpmnElement="Association_16ygme5">
      <di:waypoint x="610" y="189" />
      <di:waypoint x="669" y="110" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="Association_0mpzs4g_di" bpmnElement="Association_0mpzs4g">
      <di:waypoint x="1445" y="919" />
      <di:waypoint x="1369" y="880" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="Association_0xxc6se_di" bpmnElement="Association_0xxc6se">
      <di:waypoint x="1821" y="180" />
      <di:waypoint x="1840" y="150" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1011qdu_di" bpmnElement="SequenceFlow_1011qdu">
      <di:waypoint x="344" y="476" />
      <di:waypoint x="479" y="476" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_16d1x3e_di" bpmnElement="SequenceFlow_16d1x3e">
      <di:waypoint x="504" y="451" />
      <di:waypoint x="504" y="381" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1r0ubpp_di" bpmnElement="SequenceFlow_1r0ubpp">
      <di:waypoint x="504" y="501" />
      <di:waypoint x="504" y="560" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1nj1l69_di" bpmnElement="SequenceFlow_1nj1l69">
      <di:waypoint x="529" y="476" />
      <di:waypoint x="660" y="476" />
      <di:waypoint x="660" y="260" />
      <di:waypoint x="723" y="260" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_112ikiw_di" bpmnElement="SequenceFlow_112ikiw">
      <di:waypoint x="523" y="470" />
      <di:waypoint x="580" y="450" />
      <di:waypoint x="580" y="211" />
      <di:waypoint x="554" y="211" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1wqj52d_di" bpmnElement="SequenceFlow_1wqj52d">
```

```
    <di:waypoint x="454" y="211" />
    <di:waypoint x="274" y="211" />
    <di:waypoint x="274" y="436" />
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_15kdgkc_di" bpmnElement="SequenceFlow_15kdgkc">
    <di:waypoint x="454" y="341" />
    <di:waypoint x="314" y="341" />
    <di:waypoint x="314" y="436" />
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_1u05py2_di" bpmnElement="SequenceFlow_1u05py2">
    <di:waypoint x="188" y="476" />
    <di:waypoint x="244" y="476" />
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_0dlosaf_di" bpmnElement="SequenceFlow_0dlosaf">
    <di:waypoint x="454" y="600" />
    <di:waypoint x="294" y="600" />
    <di:waypoint x="294" y="516" />
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_0wba7ds_di" bpmnElement="SequenceFlow_0wba7ds">
    <di:waypoint x="773" y="260" />
    <di:waypoint x="828" y="260" />
    <bpmndi:BPMNLabel>
      <dc:Bounds x="786" y="247" width="17" height="14" />
    </bpmndi:BPMNLabel>
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_1cv6x8z_di" bpmnElement="SequenceFlow_1cv6x8z">
    <di:waypoint x="748" y="285" />
    <di:waypoint x="748" y="395" />
    <bpmndi:BPMNLabel>
      <dc:Bounds x="757" y="314" width="13" height="14" />
    </bpmndi:BPMNLabel>
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_00dm4px_di" bpmnElement="SequenceFlow_00dm4px">
    <di:waypoint x="928" y="260" />
    <di:waypoint x="1028" y="260" />
    <di:waypoint x="1028" y="605" />
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_0mcl6gf_di" bpmnElement="SequenceFlow_0mcl6gf">
    <di:waypoint x="773" y="420" />
    <di:waypoint x="828" y="420" />
    <bpmndi:BPMNLabel>
      <dc:Bounds x="818" y="400" width="13" height="14" />
    </bpmndi:BPMNLabel>
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_0r22vdd_di" bpmnElement="SequenceFlow_0r22vdd">
    <di:waypoint x="878" y="460" />
    <di:waypoint x="878" y="605" />
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_1u6tvcu_di" bpmnElement="SequenceFlow_1u6tvcu">
    <di:waypoint x="748" y="445" />
    <di:waypoint x="748" y="500" />
    <bpmndi:BPMNLabel>
      <dc:Bounds x="723" y="458" width="17" height="14" />
    </bpmndi:BPMNLabel>
  </bpmndi:BPMNEdge>
  <bpmndi:BPMNEdge id="SequenceFlow_11s4zb3_di" bpmnElement="SequenceFlow_11s4zb3">
```

```xml
      <di:waypoint x="748" y="580" />
      <di:waypoint x="748" y="630" />
      <di:waypoint x="853" y="630" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_02d3c5k_di" bpmnElement="SequenceFlow_02d3c5k">
      <di:waypoint x="903" y="630" />
      <di:waypoint x="1003" y="630" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_13bm0fm_di" bpmnElement="SequenceFlow_13bm0fm">
      <di:waypoint x="1038" y="615" />
      <di:waypoint x="1140" y="570" />
      <di:waypoint x="1140" y="330" />
      <bpmndi:BPMNLabel>
        <dc:Bounds x="1104" y="513" width="17" height="14" />
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0t2wvrg_di" bpmnElement="SequenceFlow_0t2wvrg">
      <di:waypoint x="1053" y="630" />
      <di:waypoint x="1160" y="630" />
      <bpmndi:BPMNLabel>
        <dc:Bounds x="1113" y="613" width="13" height="14" />
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_01woab4_di" bpmnElement="SequenceFlow_01woab4">
      <di:waypoint x="1190" y="330" />
      <di:waypoint x="1190" y="580" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1rqoomg_di" bpmnElement="SequenceFlow_1rqoomg">
      <di:waypoint x="1210" y="660" />
      <di:waypoint x="1210" y="720" />
      <di:waypoint x="1285" y="720" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_11ejla7_di" bpmnElement="SequenceFlow_11ejla7">
      <di:waypoint x="1310" y="695" />
      <di:waypoint x="1310" y="540" />
      <di:waypoint x="1400" y="540" />
      <bpmndi:BPMNLabel>
        <dc:Bounds x="1290" y="607" width="17" height="14" />
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1tjh507_di" bpmnElement="SequenceFlow_1tjh507">
      <di:waypoint x="1440" y="500" />
      <di:waypoint x="1440" y="415" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1olp77n_di" bpmnElement="SequenceFlow_1olp77n">
      <di:waypoint x="1714" y="415" />
      <di:waypoint x="1714" y="620" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0eg7ulc_di" bpmnElement="SequenceFlow_0eg7ulc">
      <di:waypoint x="1590" y="760" />
      <di:waypoint x="1714" y="760" />
      <di:waypoint x="1714" y="700" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1b9pd6i_di" bpmnElement="SequenceFlow_1b9pd6i">
      <di:waypoint x="1764" y="660" />
      <di:waypoint x="1825" y="660" />
```

```
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_14gbbiz_di" bpmnElement="SequenceFlow_14gbbiz">
      <di:waypoint x="1500" y="650" />
      <di:waypoint x="1540" y="650" />
      <di:waypoint x="1540" y="720" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0x07e52_di" bpmnElement="SequenceFlow_0x07e52">
      <di:waypoint x="1850" y="685" />
      <di:waypoint x="1850" y="720" />
      <bpmndi:BPMNLabel>
        <dc:Bounds x="1861" y="692" width="17" height="14" />
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_11v1trp_di" bpmnElement="SequenceFlow_11v1trp">
      <di:waypoint x="1870" y="800" />
      <di:waypoint x="1870" y="855" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_16uenhz_di" bpmnElement="SequenceFlow_16uenhz">
      <di:waypoint x="1335" y="720" />
      <di:waypoint x="1450" y="720" />
      <di:waypoint x="1450" y="680" />
      <bpmndi:BPMNLabel>
        <dc:Bounds x="1413" y="733" width="13" height="14" />
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1insc67_di" bpmnElement="SequenceFlow_1insc67">
      <di:waypoint x="1850" y="635" />
      <di:waypoint x="1850" y="538" />
      <bpmndi:BPMNLabel>
        <dc:Bounds x="1863" y="583" width="13" height="14" />
      </bpmndi:BPMNLabel>
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0toxbj8_di" bpmnElement="SequenceFlow_0toxbj8">
      <di:waypoint x="1845" y="880" />
      <di:waypoint x="1810" y="880" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0so8lpt_di" bpmnElement="SequenceFlow_0so8lpt">
      <di:waypoint x="1870" y="905" />
      <di:waypoint x="1870" y="1050" />
      <di:waypoint x="1810" y="1050" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0gljw3v_di" bpmnElement="SequenceFlow_0gljw3v">
      <di:waypoint x="1710" y="890" />
      <di:waypoint x="1620" y="890" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_0ljigac_di" bpmnElement="SequenceFlow_0ljigac">
      <di:waypoint x="1520" y="890" />
      <di:waypoint x="1510" y="890" />
      <di:waypoint x="1510" y="1010" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_1jem9ox_di" bpmnElement="SequenceFlow_1jem9ox">
      <di:waypoint x="1590" y="1050" />
      <di:waypoint x="1560" y="1050" />
    </bpmndi:BPMNEdge>
    <bpmndi:BPMNEdge id="SequenceFlow_08sc29q_di" bpmnElement="SequenceFlow_08sc29q">
      <di:waypoint x="1460" y="1050" />
```

```xml
        <di:waypoint x="1328" y="1050" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_1vks2be_di" bpmnElement="SequenceFlow_1vks2be">
        <di:waypoint x="1710" y="1050" />
        <di:waypoint x="1690" y="1050" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_0yuz5o0_di" bpmnElement="SequenceFlow_0yuz5o0">
        <di:waypoint x="1440" y="365" />
        <di:waypoint x="1440" y="260" />
        <di:waypoint x="1500" y="260" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_1k5xysm_di" bpmnElement="SequenceFlow_1k5xysm">
        <di:waypoint x="1600" y="260" />
        <di:waypoint x="1714" y="260" />
        <di:waypoint x="1714" y="365" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_1eag7yq_di" bpmnElement="SequenceFlow_1eag7yq">
        <di:waypoint x="1465" y="390" />
        <di:waypoint x="1500" y="390" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge id="SequenceFlow_07nvgk7_di" bpmnElement="SequenceFlow_07nvgk7">
        <di:waypoint x="1600" y="390" />
        <di:waypoint x="1689" y="390" />
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNShape id="Group_1bfu1nn_di" bpmnElement="Group_1bfu1nn">
        <dc:Bounds x="220" y="160" width="390" height="553" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Group_1v4zn2e_di" bpmnElement="Group_1v4zn2e">
        <dc:Bounds x="640" y="160" width="420" height="553" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Group_11llgzu_di" bpmnElement="Group_11llgzu">
        <dc:Bounds x="1080" y="180" width="830" height="636" />
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape id="Group_153xl6j_di" bpmnElement="Group_153xl6j">
        <dc:Bounds x="1445" y="827" width="465" height="303" />
      </bpmndi:BPMNShape>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</bpmn:definitions>
```