Lisana Berberi

Analysis of Process Variants with a Process Warehouse Approach

DISSERTATION

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften

Doktoratsstudium der Technichen Wissenschaften Fach Informatik
(L786880)

Alpen-Adria-Universität Klagenfurt
Fakultät für Technische Wissenschaften

Begutachter: O.Univ.–Prof. Dipl.–Ing. Dr. Johann Eder
Begutachter: Prof. Dr. hab. Tadeusz Morzy

Institut für Informatik-Systeme
Information and Communication Systems

Affidavit

I hereby declare in lieu of an oath that

- the submitted academic paper is entirely my own work and that no auxiliary materials have been used other than those indicated,
- I have fully disclosed all assistance received from third parties during the process of writing the thesis, including any significant advice from supervisors,
- any contents taken from the works of third parties or my own works that have been included either literally or in spirit have been appropriately marked and the respective source of the information has been clearly identified with precise bibliographical references (e.g. in footnotes),
- to date, I have not submitted this paper to an examining authority either in Austria or abroad and that
- when passing on copies of the academic thesis (e.g. in bound, printed or digital form), I will ensure that each copy is fully consistent with the submitted digital version.

I am aware that a declaration contrary to the facts will have legal consequences.

Lisana Berberi m.p. *Klagenfurt, December 2020*

Keywords

Process variants analysis, process variants, process warehouse, meta-model, data warehouse, key performance indicators(KPI), business process management, reference model, variability, process measures, abstract process, consolidated hierarchy, generic model, BPM, generic activity, specialization of process, generalization of process, specialized activity, OLAP operation, process pattern analysis.

Zusammenfassung

Prozessmodellvarianten sind Sammlungen ähnlicher Prozessmodelle, die sich herausbilden da im Laufe der Zeit an einem bestimmten Prozess in einem bestimmten Bereich Anpassungen vorgenommen wurden z. B. in dem Order-to-Cash- oder Procure-to-Pay Prozess im Reseller oder Beschaffungsbereich.

Diese Anpassungen führen zu einigen Abweichungen zwischen Prozessmodellen, die hauptsächlich identisch sein sollten, sich dadurch jedoch geringfügig unterscheiden. Diese Abweichungen sind auf neue Verfahren, gesetzliche Bestimmungen in verschiedenen Ländern, Abweichungen aufgrund unterschiedlicher Entscheidungshistorien und organisatorischer Verantwortlichkeiten sowie auf unterschiedliche Anforderungen für verschiedene Unternehmenszweige zurückzuführen.

In dem bestehenden Ansatz zur Erfassung und Analyse von Prozessvarianten sind zwei Richtungen zu unterscheiden: im Geschäftsprozessmanagement und im Bereich Business Intelligence/Data Warehousing.

Gegenwärtige Ansätze zur Verwaltung von Prozessvarianten unter Verwendung eines Geschäftsprozessmanagementsystems (BPMS) weisen drei Mängel auf die sich auf ihre praktische Verwendbarkeit auswirken. Erstens führen diese Prozessmodelle zu Datenredundanz, wenn alle diese Varianten separat aufbewahrt werden da Modellvarianten häufig für die meisten Teile ähnlich oder identisch sind. Zweitens kann das Modellieren und Verwalten dieser Prozesse zu einer zeitaufwändigen und fehleranfälligen Aufgabe für Geschäftsdesigner führen. Drittens kann es sein das einige Optimierungstechniken auf eine bestimmte Variante angewendet werden, ohne die anderen damit verbundenen zu berücksichtigen.
Ansätze zur Analyse dieser Varianten mithilfe einer Data Warehouse Lösung sind durch die Abstraktion und Konsolidierung aller Varianten zu einem einzigen generischen Prozessmodell begrenzt, wodurch die Möglichkeit verhindert wird, verschiedene Teile verschiedener Varianten zu unterscheiden und zu vergleichen. Dieser Mangel wirkt sich auf die Entscheidungsfindung von Geschäftsanalysten für einen bestimmten Prozesskontext aus. Infolgedessen erwies sich die Analyse und der Vergleich dieser verschiedenen Varianten innerhalb eines gemeinsamen IT-Systems als alles andere als trivial.

In Anlehnung an eine designwissenschaftliche Forschungsmethode werden in dieser Arbeit die oben genannten Mängel behoben, indem ein Rahmen für die Analyse von Prozessvarianten vorgeschlagen wird.
Das Framework besteht aus drei ursprünglichen Beiträgen: (i) einem neuartigen Metamodell von Prozessen als allgemeines Datenmodell zur Erfassung und Konsolidierung von Prozessvarianten in einem Referenzprozessmodell; (ii) ein Prozesslagermodell zur Durchführung typischer OLAP Operationen an verschiedenen Variationsteilen, um so die Entscheidungsfindung zu unterstützen; (iii) ein Vergleich zwischen bestehenden Metamodellierungsansätzen der Forschungsgemeinschaft auf der Grundlage verschiedener Kriterien aus der Literaturrecherche. Die Framework-Konzepte wurden formal definiert und anhand von

zwei verschiedenen Szenarien validiert. Darüber hinaus wird ein Prototyp implementiert, um die Validierung des Frameworks zu unterstützen.

Abstract

Process model variants are collections of similar process models evolved over time because of the adjustments that were made to a particular process in a given domain, e. g.,order-to-cash or procure-to-pay process in reseller or procurement domain. These adjustments produce some variations between these process models that mainly should be identical but may differ slightly. These variations are due to new procedures, law regulations in different countries, variations due to different decision histories and organizational responsibilities and to different requirements for different branches of an enterprise.

Existing approaches for capturing and analysing process variants diverge in two directions: in business process management and in business intelligence/data warehousing.

Current approaches for managing process variants using BPMS suffer from three shortcomings that affect their usability in practice. Firstly, these process models result in data redundancy as often model variants are similar or identical for most parts if all these variants are kept separately. Secondly, to model and maintain these processes may result in a time-consuming and error-prone task for business designers. Thirdly, some optimization techniques might be applied to a specific variant without considering the other ones related to it.
Whereas, approaches to analyse these variants using a data warehouse solution suffer from adequately abstracting and consolidating all variants into one generic process model, to provide the possibility to distinguish and compare among different parts of different variants. This shortcoming affects decision making of business analysts for a specific process context. As a consequence, analysing and comparing these multiple variants within a common IT system proved far from trivial.

Following a design science research method, this thesis addresses the above shortcomings by proposing a framework to analyse process variants.
The framework consists of three original contributions: (i) a novel meta-model of processes as a generic data model to capture and consolidate process variants into a reference process model; (ii) a process warehouse model to perform typical OLAP operations on different variation parts thus providing support to decision-making; (iii) a comparison between existing meta-modeling approaches by the research community based on different criteria from literature review. The framework concepts were formally defined and validated using two different scenarios. Moreover, a prototype is implemented to support the validation of the framework.

Acknowledgments

Thank God this journey is over! It has been quite a long one. First and foremost, I would like to express my immense and sincere gratitude to my supervisor Prof. Johann Eder. I really appreciate his continuous suggestions, comments and patience during my research work. I have to thank him as well for his support to cover my accommodation expenses during my short stays in Klagenfurt (two months/year) in the last two years and my tuition fees for two semesters. Without his help I wouldn't be able to complete my thesis in time. Additionally, I gratefully acknowledge the subsidy received from *ÖH Social Fund* towards my PhD for seven months.
A sincere thanks go to Prof.Ass. Julius Köpke for his valuable feedback on my work and to my ex advisor Dr. Christian Koncilia for his support.

I owe the merits of pursuing a research to Prof. Andreas Bollin who made possible the renewal cooperation between my home university and Klagenfurt university on behalf of the staff of University of Shkodra to achieving further academic qualifications.

This thesis wouldn't have been possible without the support I received from my colleagues and friends Fatjona and Sidita for their help on managing many of my administrative tasks as a lecturer during my absence. A special thanks go to my friend Lulzim for his help during one of my implementation task. Special thanks go to Elisa and Sara as well, for their friendship during my stays in Klagenfurt, which helped me not feel lonely during the time I was away from my family.

My sincere thanks also go to my noble mother Bukurije and my beloved father Skender for their constant love, support, and for instilling me the desire to learn. They took care of my daughter years ago when she was a newborn, and my little son during my short visits in Klagenfurt. My careful mother-in-law Minire and father-in-law Behxhet deserve special thanks as well. They really love me as a daughter and took care of my children during my absence.

Last but most importantly, I'd like to thank my husband Ardi, my son Seldi and my daughter Ajla - to whom I dedicate this achievement. They deserve a special thank for their love, understanding, endurance, encouragement, trust, and admirable support over these years, without them I wouldn't have made it (they are my everything). I would like to dedicate this achievement to the memory of my unique brother Elvis as well. He was so precious to me! I miss him a lot!

to my family and to the memory of my unforgettable brother Elvis.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Chapter 1 starts by introducing two problem areas i. e., Business Process Management and Process-oriented data warehouse related to process variants analysis.

1.1 Problem Area

Process model variants, as collections of similar process models, may evolve over time because of the adjustments made to the same business process in a given domain, e. g.,order-to-cash or procure-to-pay process in reseller or procurement domain. These adjustments produce some variations between these process models. Surely, between business processes across department of the same organization, or across companies in a given industry many common activities are frequently found. For example, typical process *procure-to-pay* often consists of a business process that starts from the moment a procure invoice is received from a vendor after a customer places an order and fulfilled if the vendor has received the corresponding payment. All these *procure-to-pay* processes include activities related to receiving, invoicing and payment. They may look the same but they slightly differ from each other. For example, a procure-to-pay process if customer decides to pay cash (e. g., only a verification of cash amount is needed) is different from a procure-to-pay process if customer decides to pay by bank transfer (e. g., a confirmation by a billing specialist is needed). Additionally, when it comes to re-design and analyse their *procure-to-pay* process a company should consider the common practices of other companies as they have a lot to learn from each other. Especially, of great importance is having an information on management of the work progress between different parts of different variants and then select the most efficient one. Dedicated technologies lack on effectively manage the information on processes encoded in process models and process execution records (Polyvyanyy et al., 2017).

1.1.1 Business Process Management Systems

BPM as a mature discipline is acknowledged by practitioners (analysts, consultants, software developers and managers) and academics. This is illustrated by the availability of many BPM systems and a range of BPM-related conferences (Van Der Aalst, 2013). Furthermore, companies currently undertaking various Business Performance Management (BPerM) initiatives, have come to the conclusion that the best way to link their strategic and operational levels is via BPs. In fact, correct mapping between strategic and business process KPIs (Key Performance Indicators) will ensure that BPs are used to implement organizational strategies (Melchert et al., 2004). At the same time, this mapping enables organizations to detect possible process-related problems at the operational level and then address them, within the most appropriate BPs.

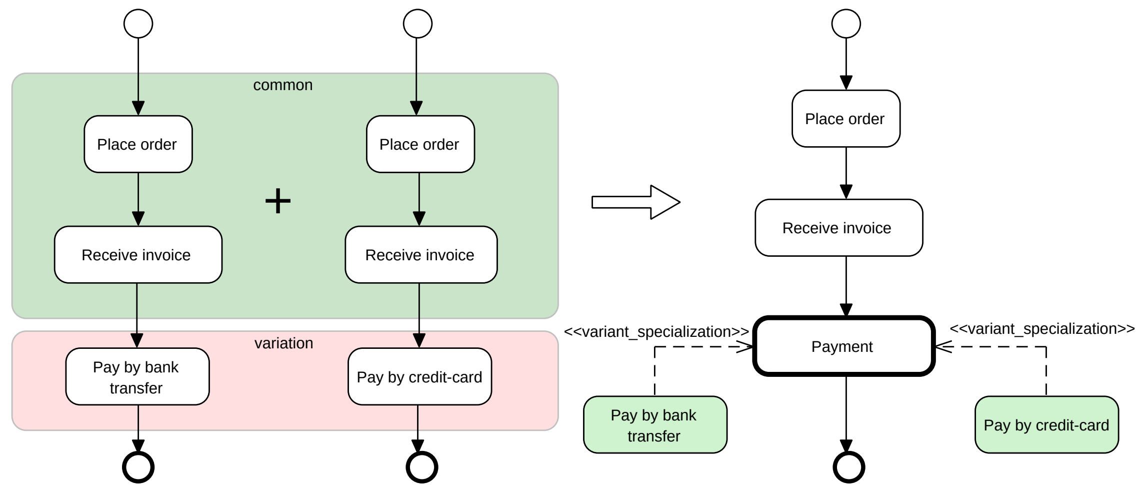


Figure 1.1: An example of reference(global) process model abstracting multiple process variants

To understand how a reference or global process model is constructed, let us consider a concrete example. This simple example refers to customer invoice payments after ordering his/her goods or services. Figure 1.1 shows two variants of the order-to-pay process represented Business Process Modelling Notation (BPMN)(BPMN, Spec.). These variants reflect two possibilities to pay: the first pay by bank transfer (filling a bank statement), the other, pay by credit-card (check customer balance). We show how a reference can be constructed by identifying the commonalities and variability among them. The choice between pay by credit-card or pay by bank transfer represents a variability in this process: depends on different drivers such as type of invoice, type of goods etc. The two variant activities are integrated to a new generic (abstract) activity named *Payment* as shown on the right-hand side of the figure. We use a stereotype named *«variant specialization»* assigned to the generic connector between the generic activity and the specialized activities.

1.1.2 Process-oriented Data Warehouse

Process Warehouses (Eder et al., 2002; Pau et al., 2007; Benker, 2016) are an appropriate means for analysing the performance of business process execution using well established data warehouse technology and on-line analytical processing (OLAP) tools. Data that stems from process executions is analysed using some typical dimensions such as process, time, geographic location and resource. In particular, they allow the definition, computation and monitoring of key performance indicators along several dimensions. Typical dimensions in process warehouses are process, time, actor, geographic location. While most of the dimensions are organized in hierarchies supporting roll-up and drill-down operations, the process dimension usually is relatively flat, often comprising just two levels: activity and process, sometimes augmented with a part-of hierarchy but typically without a generalization hierarchy. This structure has some shortcomings: frequently processes exist in several different variants or versions in the same enterprise. These variants mainly arise due to process evolution and the arising differences add additional complexity to modelling data warehousing. Thus making it difficult to provide aggregate management information of activities if many variants of the same business process are present.

1.2 Problem Statement

Analysing a family of process variants is cumbersome when using current process managing tools and current process warehouse technology. Firstly, a correct way to capture design-time of process model variants should be established. Subsequently, a process warehouse model is necessary to analyse the execution logs during run-time of respective variants. However, the process dimension of a PW usually is relatively flat, often comprise only two levels: process and activity (e. g., activity *Charge credit* within process *PayInvoice*) and sometimes augmented with a part-of hierarchy (e. g., subprocess *Make billing inquiry* within process *ReceiveInvoice*) but a generalization hierarchy is missing which is essential in establishing and organizing the structure of dimension attributes. As a consequence, is not possible to perform typical OLAP operations, e. g., roll-up or drill-down and isn't possible to compute (calculate) key-performance indicators (KPIs) (e. g., average duration/cost of customer payments using credit-cards for all orders received via online shop) between different variants at different levels of genericity.
In this context, the main research question is:
RQ: How can a family of process variants be effectively and efficiently analysed using a process warehouse approach?

This research question specifies the interoperability between business process modelling, enactment and data warehouse research areas with the aim of analysing different variants in a multidimensional perspective.
From this research question the following three sub questions are derived:

- *RQ 1: How to develop a method for a comprehensive analysis of process variants?* We aim to develop a method to analyse process variants to have the possibility to distinguish between similar and different parts of the same business process type.

- *RQ 2: How can a family of process variants be consolidated in a process dimension hierarchy of a PW schema?* We aim to define a consolidation hierarchy between process variants to fit the dimension structure of a process dimension of a PW schema. In so doing, we can perform typical OLAP operations such as roll-up (view data by decreasing the level of detail) or drill-down (view data by increasing the level of detail).

- *RQ 3: How to compute KPIs to compare between different variants at different hierarchy levels?* We aim to compute KPIs along different variants and to different levels of process dimension hierarchy in order to be able to compare these variants and select the most efficient one.

1.3 Objectives and Approach

After we identified shortcomings, the core objectives of this research are to:

- offer business analysts to consolidate these variants into one hierarchy and do drill-down and roll-up in different levels of the hierarchy
- compute different aggregated management in different variation parts
- offer business analysts to compare between variants to select the most efficient one as a best proven practice.

To achieve these objectives in providing a comprehensive analysis of process variants this thesis proposes a process warehouse approach. A process performance analysis is crucial to identify how effective (e. g. measure customer satisfaction for a product or process) and efficient (e. g. measure time, cost and resource utilization) a business process is. Moreover it helps in estimating process improvement efforts.

Process variants are defined as sets of similar process models that may evolve over time because of the adjustments made to the same business process in a given domain. During our study we considered two different scenarios of these variants: the customer invoice payment process variants as our motivation example and the building permit application process variants as a real-life case study. In practice dozens of process variants exist increasing thus modelling and maintaining efforts which is both time-consuming and error-prone task. Current business process management systems and traditional process warehouses lack on adequately abstracting and consolidating all variants into one generic process model, to provide the possibility to distinguish and compare among different parts of different variants. This shortcoming affects decision making of business analysts for a specific process context. As a consequence, analysing and comparing these multiple variants within a common IT system becomes quite hard for process designers.

To overcome these shortcomings we establish first a correct way to capture design-time of process model variants and secondly built a process warehouse model to analyse the event logs during runtime execution of the respective variants.

The approach consists of a core of a process warehouse model which allows to express a generalization hierarchy of processes to adequately capture process variants. This generalization hierarchy can be generated from a meta-model of business process models which introduces the notion of generic activities which generalize a set of activities (e. g., pay by credit card, by check, or by third-party (PayPal) could all be generalized to an activity payment). Based on these given hierarchies of activities we can define generalization hierarchy of processes for the "process" dimension of a process warehouse. This hierarchy can then be used to roll-up or drill down when analysing the logs of the executions of the various process variants and it makes it much easier to compare key-performance indicators between different variants at different levels of genericity.

1.4 Outline

This thesis is organized as follows. Chapter 2 presents a running example of processing customer invoice payments of different branches of a financial agency. Chapter 3 reviews current literature on meta-modelling of process variants and introduces our proposed meta-model to capture process variants. Chapter 4 presents related work of existing process warehouse models to analyse business processes(without considering variants). Chapter 5 exposes our process warehouse model to analyse these variants. Chapter 6 discusses about process mining techniques available for process variants. Chapter 7 discusses the prototype implemented as a proof-concept to test and verify the design and functionalities of the framework concepts. Chapter 8 evaluates the approach based on two case studies. Finally, Chapter 9 concludes this thesis by summarizing the work presented and discussing possible further improvements.

CHAPTER 2

Motivating Example

To motivate our approach we start by focusing our attention on the modelling and consolidating of different process variants into a generic customized process as a benchmark example. We introduce different simplified variants of processing customer invoice payments in section 2.1. Next, we argue over shortcomings of current process warehouse approaches to analyse processes with variants, pose some typical queries related to process variants that are of main interest to business analysts.

2.1 Variants of processing customer invoice payments

Let's assume we have an illustrative example of a core business process, e. g., *Processing Customer Invoice Payments* of a financial administration agency that is modeled as a collaboration between two processes named *ReceiveInvoice* and *PayInvoice*. *ReceiveInvoice* process consists of a set of activities that sends an invoice (either e-invoice or hard-copy) to a customer (buyer) with/without requesting a payment apriori for ordering its goods or services. Whereas, *PayInvoice* process consists of a set of other activities that submit or complete with the payment (either by cash, bank transfer, credit-card or paypal) after a customer invoice is received. We express these variants using BPMN notation [1] which is now standardized by (ISO/IEC-19510:2013) to bridge the gap between business process design and process implementation. We use this process with the set(family) of its variants as a running example throughout the thesis. Usually, dozens up to thousands of variants may exist of the same business process depending on different factors. For example, in our running example variability is caused by the order customer choice (e. g., either via online shops or call centers) and/or method of invoice payment (e. g., either cash or credit-card) or invoice type (e. g., either hard-copy or e-invoice) of the designed process models in different branches of different cities. Therefore, we represent these variants as shown in Figure 2.1, Figure 2.2 and Figure 2.3. These five variants share some similarities highlighted with light gray, but they show differences, too. A detail description about these variants is as follows.

[1]OMG: Business Process Model and Notation http://www.bpmn.org/

All variants start with activity *Place order* by a customer for ordering his/her goods and services.

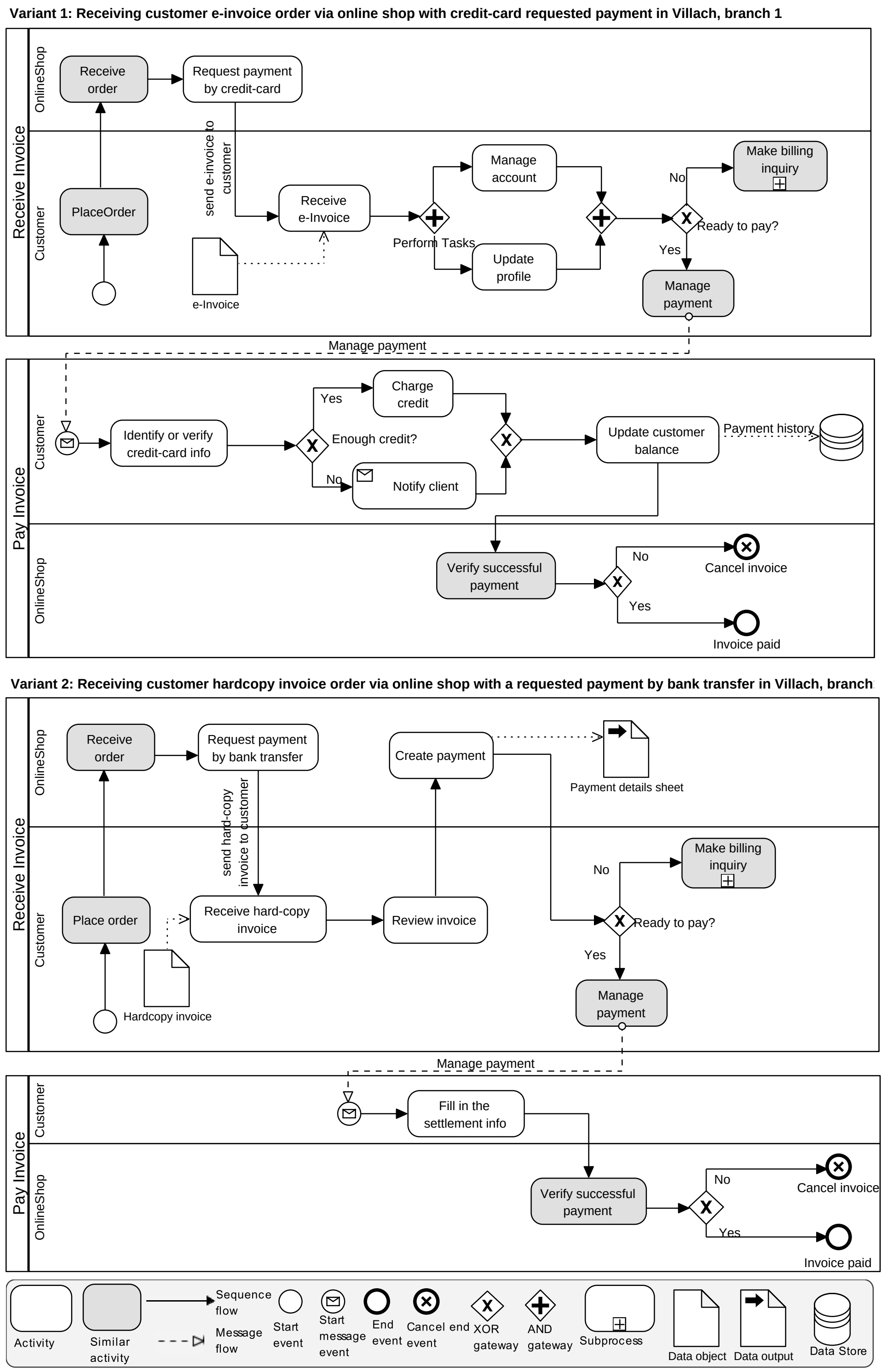
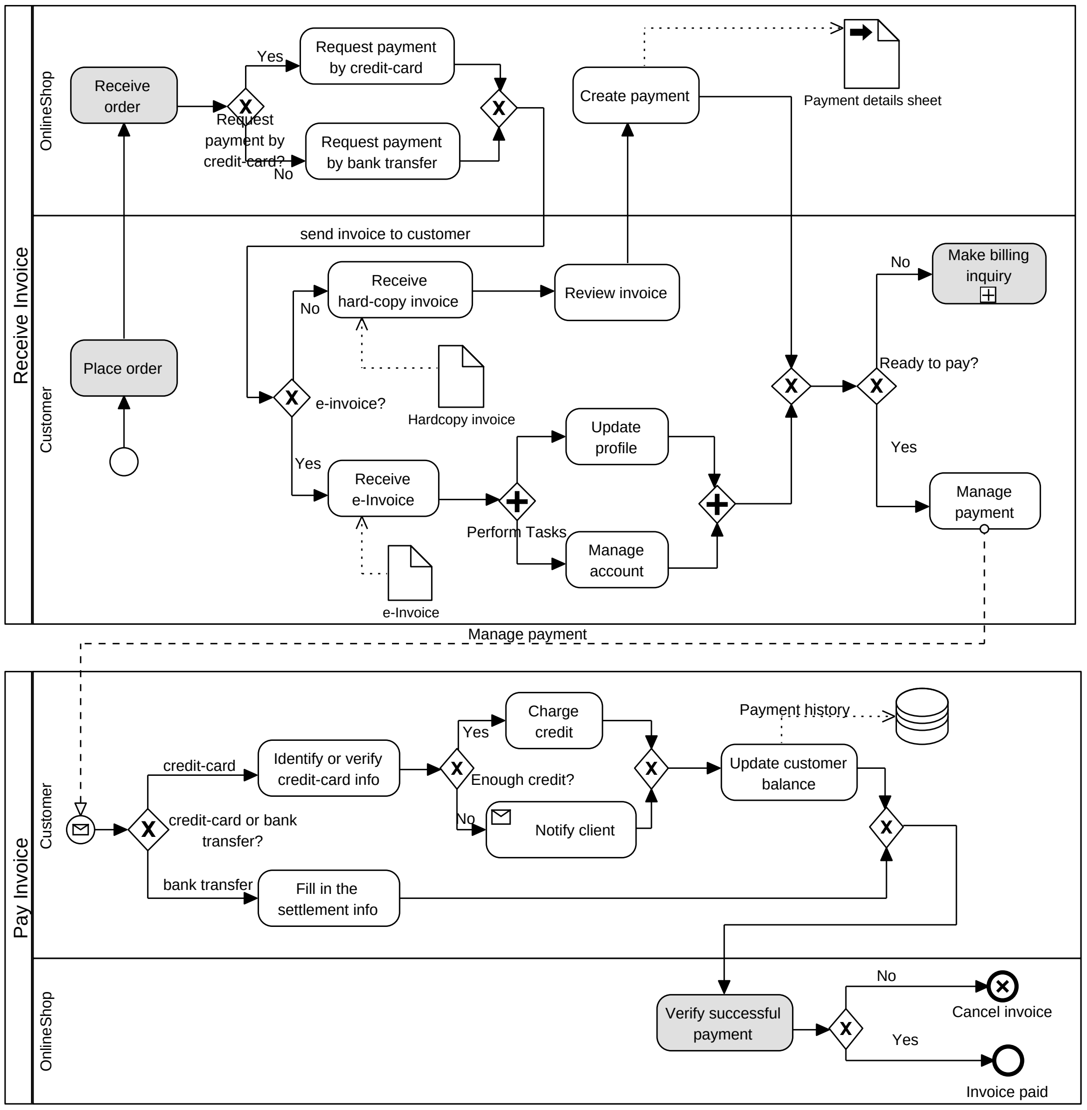


Figure 2.1: Process Variants (1)

In variants 1-4 the order is received (i. e., activity *Receive order* depicted with a rounded rectangle) via an online shop, whereas in variant 5 via a call center. In the first three variants the processing of customer invoice payments is shown in different branches of the same city Villach, whereas variants four and five show how these processes are modeled in an agency located in a different city, e. g., in Klagenfurt. After the order is received, activity *Request payment by credit-card* is followed in variant 1 Figure 2.1. Customer receives an e-invoice (i. e., *Receive e-invoice*, then, two parallel activities should be performed: *Manage account* and *Update profile* followed by a decision point (depicted with an X diamond) where subprocess *Make billing inquiry* is executed if customer is not ready to pay otherwise activity *Manage payment* (which is a common activity among all variants) is performed and the the flow is shifted to the second process *PayInvoice* via a message start event (depicted with an envelope inside a circle). This subprocess deals with two types of inquiry by the customer: *self-service* or via a *place call* for further investigation related to invoice. If no billing adjustment are needed then the invoice can be paid executing the intermediate event (depicted with a right arrow in a double circle outline) to shift the control to process *PayInvoice* otherwise activity *Make billing adjustment* is performed by a billing specialist of the online shop to adjust billing items and afterwards the altered invoice is sent back to the customer. An expanded view of this subprocess is modeled as a separate business process diagram depicted in the bottom of Figure 2.2. In process *PayInvoice* after identifying credit-card info (i. e., *Identify or verify credit-card info* activity) the credit is charged if customer has enough credit to its account, followed by activity *Update customer balance* which updates payment history data store (depicted with a cylinder) and finishes by *Verify successful payment* activity. Otherwise customer receives a notification about its insufficient credit amount following the cancellation of its invoice order (depicted with an x circle).
In variant 2 (Figure 2.1) after customer order is received activity *Request payment by bank transfer* is performed. Afterwards, a hard copy invoice is sent to the customer (*Receive hard-copy invoice*) generating a data object *Hardcopy invoice* which serves as input object for activity *Review invoice.* Then, a payment sheet named *Payment details sheet* is generated as an output data object of activity *Create payment* performed by an employee of the Online shop. In contrary to variant 1, the payment should be done by bank transfer as requested in the process *ReceiveInvoice.* Accordingly, (i. e.,*Fill in the settlement info*) is executed followed by activity *Verify successful payment* otherwise the invoice is canceled if the bank transfer isn't settled with the right information.
In variant 3 (Figure 2.2) a request payment (either by credit-card or bank transfer) is required by the agent of the online shop to send the invoice(either e-invoice or hard copy invoice) modeled via a decision point to evaluate the choice. Based on the chosen invoice either payment by credit-card or payment by bank transfer is possible as expressed via the (i. e., decision point named *Pay method?*). Again, after the verification of the payment the process completes with the event *Invoice paid* for successful payments or event *Cancel invoice* for unsuccessful ones.
In variant 4 (Figure 2.3) a pre-request payment is not required by the online shop, which means the decision is left to the customer after receiving an e-invoice or hard copy invoice. Here, possible payment methods are by credit-card, bank transfer or by third-party such as Paypal.

Variant 3: Receiving customer invoice order via online shop with the option of request payment either by credit-card or bank transfer in Villach, branch 2



Make billing inquiry subprocess as a separate business process diagram

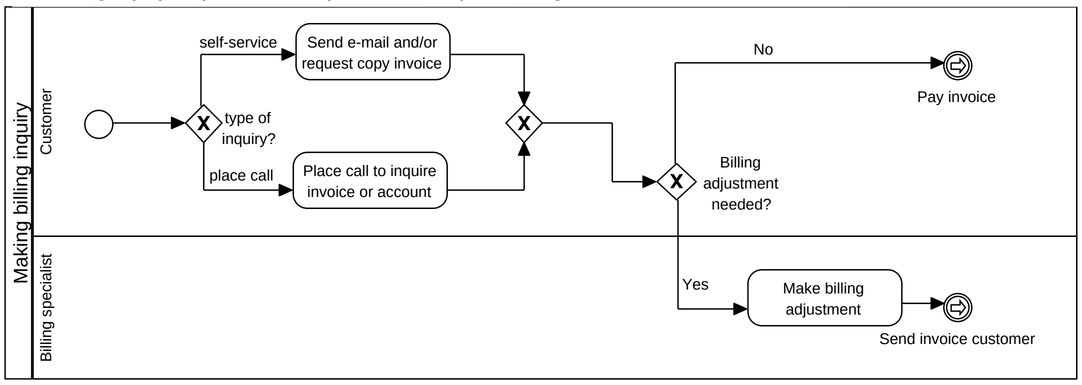
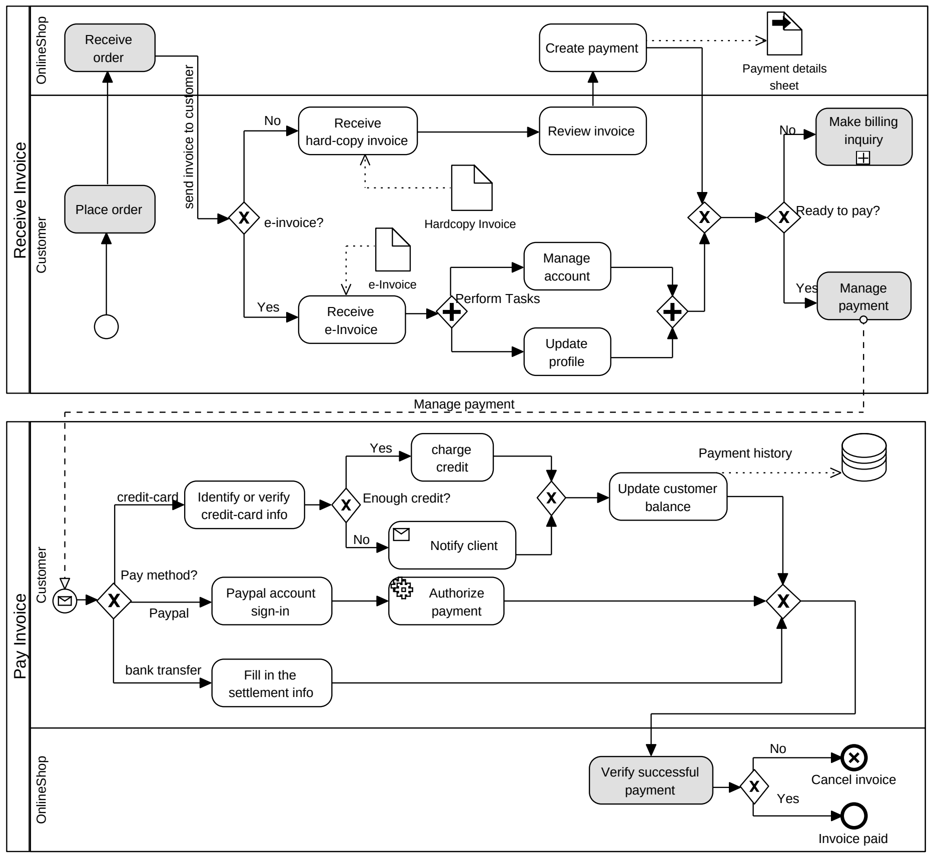


Figure 2.2: Process Variants (2)

Variant 4: Receiving customer invoice order via online shop in Klagenfurt without a pre-request payment



Variant 5: Receiving customer invoice order via a Call Center in Klagenfurt and pay either by credit-card or cash

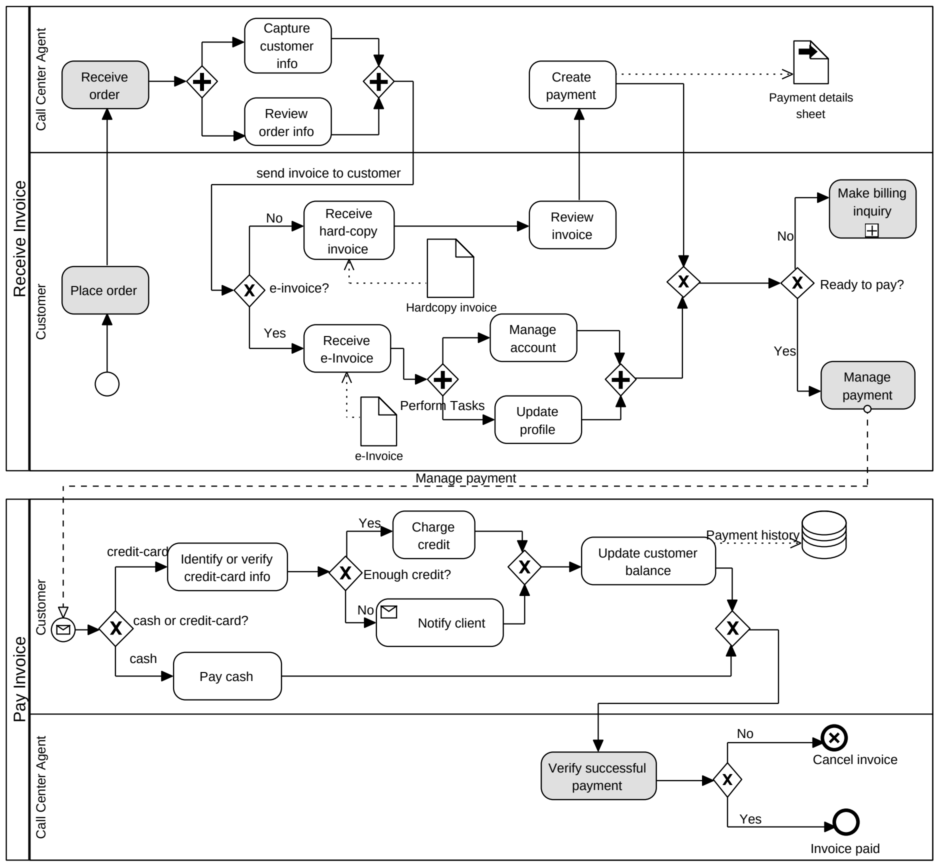


Figure 2.3: Process Variants (3)

If customer chooses to pay by PayPal then after the customer signs-in (i. e., *Paypal account sign-in* activity) and authorize the payment (i. e., *Authorize payment* system activity by a PayPal processor), it completes with a verification activity if the payment is successful or not. And in the last variant (Figure 2.3), customer places its order via a call center and after receiving its order the agent of the call center performs two simultaneous activities (i. e., *Capture customer info* and *Review order info*) and then sends invoice (either e-invoice or hard-copy) to the customer. Afterwards, two options are possible for the invoice payment, either by credit-card or cash. The process finally completes with events *Invoice paid* or *Cancel invoice* based on the payment verification step if successful or not.

These variants exist due to new or changed law regulations on different countries, variations due to different decision histories and organizational responsibilities and to different requirements for different branches of an enterprise. We assumed these variants are modeled using the *multi-model* approach as classified by (Hallerbach et al., 2008), which means that they are designed and kept separately resulting in data redundancy as often model variants are similar or identical for most parts. Furthermore, is far from trivial to combine existing variants to a new one (semi-)automatically. This solution is feasible only if few variants exists or if they differ significantly from each other.

However, in practice a large number of variants occurs increasing thus modeling and maintaining efforts which is both time-consuming and error-prone task. As processes may evolve over time, some optimization techniques might be applied to a specific variant without considering the other ones related to it. As a consequence, analysing and comparing these multiple variants within a common IT system becomes quite hard for process designers.
A typical case is when two companies merge their activities and benchmarking companies' processes based on performance gives the opportunity to select the best and most frequent variant. Furthermore, efficient paths of a specific variant can be applied to the rest of the processes to improve performance whereas inefficient ones can be substituted or removed. Another way of managing these variants is using the *single-model* (see Figure 2.4) approach where these variants might be expressed in a single process definition with the excessive use of XOR-Splits. The resulting processes are large, difficult to understand and to communicate and overloaded, and new process definitions still comprise of all the past processes definitions they should replace (Berberi et al., 2018). Moreover, it isn't possible to distinguish between normal and variant-specific branchings (e. g., our *PayInvoice* process includes a decision to pay by bank transfer, i. e., perform activity *Fill in the settlement info* if bank transfer choice is selected and if activity *Request payment by bank transfer* is either performed or skipped, whereas in the model-side it's ambiguous and mixed with the "normal" process logic), unless these variant-specific conditions are marked and represented explicitly using special conventions (Hallerbach et al., 2008).

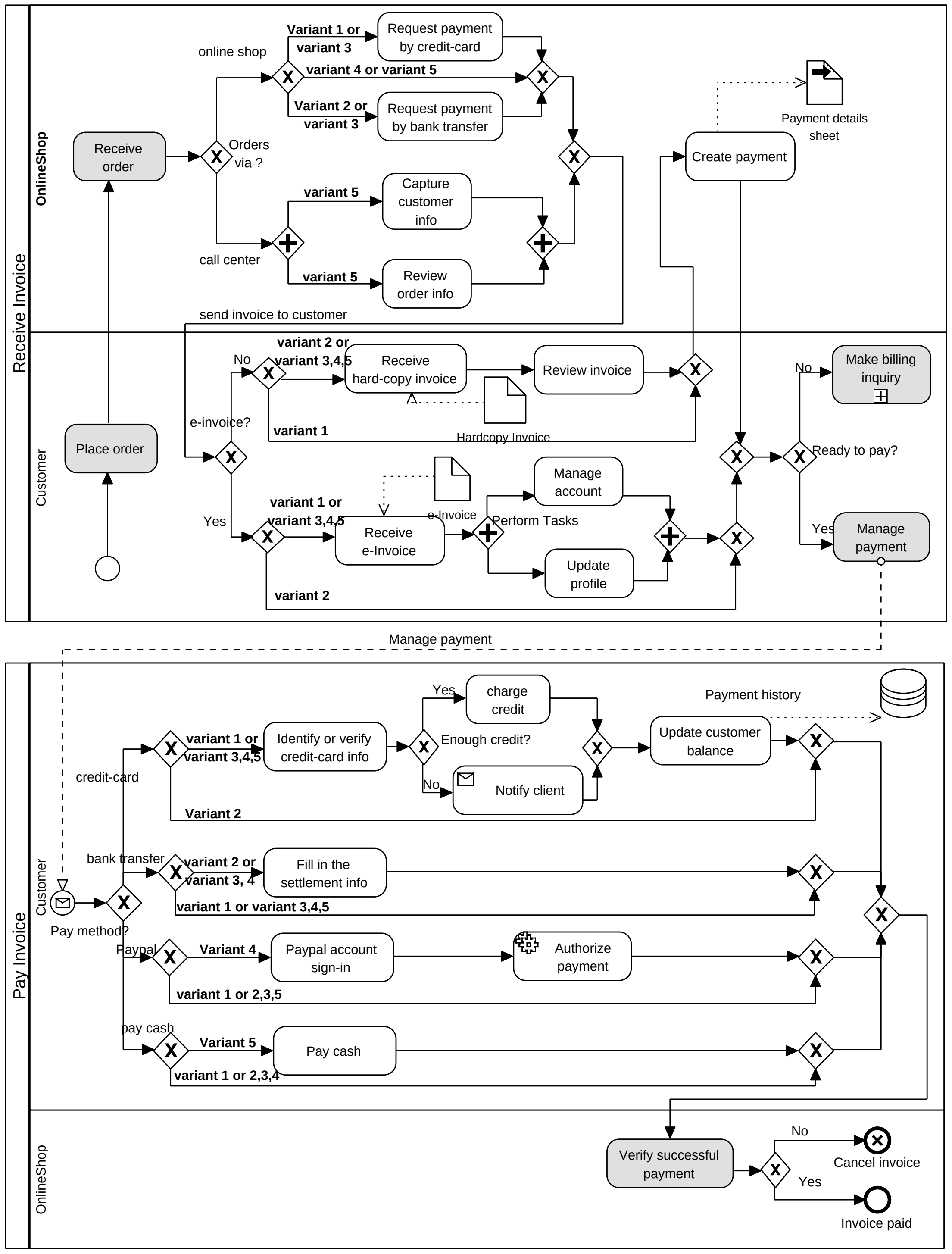


Figure 2.4: Invoice payment process variants realized by means of conditional branches

Some interesting queries based on time- and cost-related process performance that might be of interest for business analysts are as follows.

User Queries:

- Display the minimum exe time of payments among different variants for those orders received via online shop?
- Display the cycle time of processed payment after an e-invoice is received from an online-shop agent?
- Display the average exe time of orders if the payment is requested by cash?
- Display the average process duration of payments by credit-card performed by a specific employee of an online shop?
- Display the minimum exe time of payments using only PayPal for those orders received via online shop?
- Display the average unit cost for employees if payment by credit-card were processed?
- Display the cycle time of patterns "P_X, $*$ " from all variants, where P_X = *C: Capture customer info or D: Review order info or E: Request payment by credit-card or F: Request payment by bank transfer* ? i. e., all processed payments after an order is processed either by an online shop or call-center?
- Display the cycle time of patterns "P_Y, $*$ " from all variants, where P_Y = *G: Receive hard-copy invoice or H: Receive e-invoice* i. e., all processed payments after customer receives his invoice?
- Display the average process duration of payments with patterns "P_Z, $*$ " performed by an organization unit, where P_Z = *O: Pay cash*, *P: Fill in the settlement info Q: Paypal account sign-in R: Identify or verify credit-card info* , i. e., of different payment options?

In the next section we introduce our meta-model to model process variants and compare our approach among state-of-the-art of existing ones.

CHAPTER 3

Process Variants

Chapter 3 gives a background of the basic notions, reviews current literature relevant to the topic of business process variants modelling and describe our meta-model of process variants.Section 3.2 provides an overview of meta-models of business processes with variants. Specifically, the focus is on those meta-models that have been extended in order to capture variations of business processes. In section 3.3 we present a full meta-model capturing both business processes design-time and run-time aspects. Next, in section 3.4 we give a comparative analysis of the current meta-modelling approaches. And finally in section 3.5 we summarize and discuss.

3.1 Introduction

In many process-aware information systems (PAIS) during design-time phase, many variants of the same process often have to be specified. Here, we introduce the basic notion of a process variant as follows.
A ***process model variant*** or shortly named *process variant* is an adjustment of a particular process to specific requirements building the process context (Hallerbach et al., 2008). A process context is directly related to all the elements that comprise a business process. This includes several contextual properties, such as e. g., process domain properties, control flows, goals specified, resources assigned, organizational units associated, etc. Depending on the process context type, different variants of our process are required, whereas the context is described by country-specific, order-specific, invoice-specific, and payment-specific variables.

3.2 State-of-the-art of process variants meta-modelling

This section provides an overview of cutting-edge meta-modelling approaches to capture variability of business process models for modelling and/or managing configurable processes. Instead of visualizing all proposed meta-models we show how our running example is represented in their solutions or through their case studies. Therefore, we exclude from our analysis approaches that do not propose a meta-model for process variants. Furthermore, in the following subsections, from sections 3.2.1 to 3.2.4 we classify proposed approaches realized by means of different ≪*variability mechanisms*≫ suitable for business processes. A variability mechanism is defined as a technique for the derivation of process model variants from existing process models.

3.2.1 Inheritance and parameterization variability mechanism

As discussed by (Puhlmann et al., 2005) these two variability mechanisms introduce the following:

- *Inheritance* allows for the replacement or addition of a model element, e. g., activity by the specialized one.

- *Parameterization* allows for controlling the behaviour of single execution step in a process by configuring the process with corresponding parameter values.

To introduce variability and configuration modelling to the processes in PESOA domains, (Bayer et al., 2005) proposed a conceptual model with variation points where fixed activities are marked with stereotypes applied in both UML ADs (Activity Diagrams) and BPMN. Their approach is called variant-rich process modelling. The stereotype *«VarPoint»* is assigned to activities of a process model in which variability can occur. An *abstract* activity is represented by a variation point, such as "Customer info" that is specialized with one or more of the concrete variants (variants are inclusive). For example, "Review order info" and "Capture customer info" assigned with the stereotype *«Variant»* are specializations of "Customer info". With the stereotype *«Abstract»* are marked also abstract activities "Invoice type" and "Payment method" in our example where variation points are resolved by selecting only one of the concrete variants.

Figure 3.1 (a) shows the process model for processing invoice payments in PESOA-BPMN where activities have been marked as variation points with their variants, e. g., "Identify or verify credit-card info" marked as default activity of "Payment method", as being the most common choice in this process. In this case, a variation point with the stereotype *«Default»* represents the default variant. Whereas, activity "Processing order using" is assigned with the stereotype *«NULL»* to indicate its optional behaviour to one of the specialized activities annotated with *«Optional»* stereotype. During customization variability in this point can be resolve by selecting one of its specialized variants such as "Request payment by credit-card" or "Request payment by bank transfer" or may completely be dropped from the process model. Accordingly, Figure 3.1 (b) shows an excerpt fragment of the configurable BPMN process model for the derived process variant of ordering e-invoice via online shops with a pre-request payment by credit-card.
To abstract from the configurable process model and its variation points during configuration (Schnieders and Puhlmann, 2006) propose to use feature models in contrary to PESOA where (Bayer et al., 2005) stated that the abstraction and transformation to derive process variants from a configurable process is out of their scope. A feature model is represented graphically by one or more feature diagrams. Figure 3.2 (a) depicts the feature diagram for our invoice payments example and (b) depicts the configured feature diagram for receiving hard-copy invoices with a pre-requested payment by bank transfer. A domain

property for each process variants is captured as a feature such that when a feature is disabled in a feature model configuration, the corresponding variant is removed from the process model. There are features related to the options available for ordering via online shops or call centers, type of invoices and methods of payment.

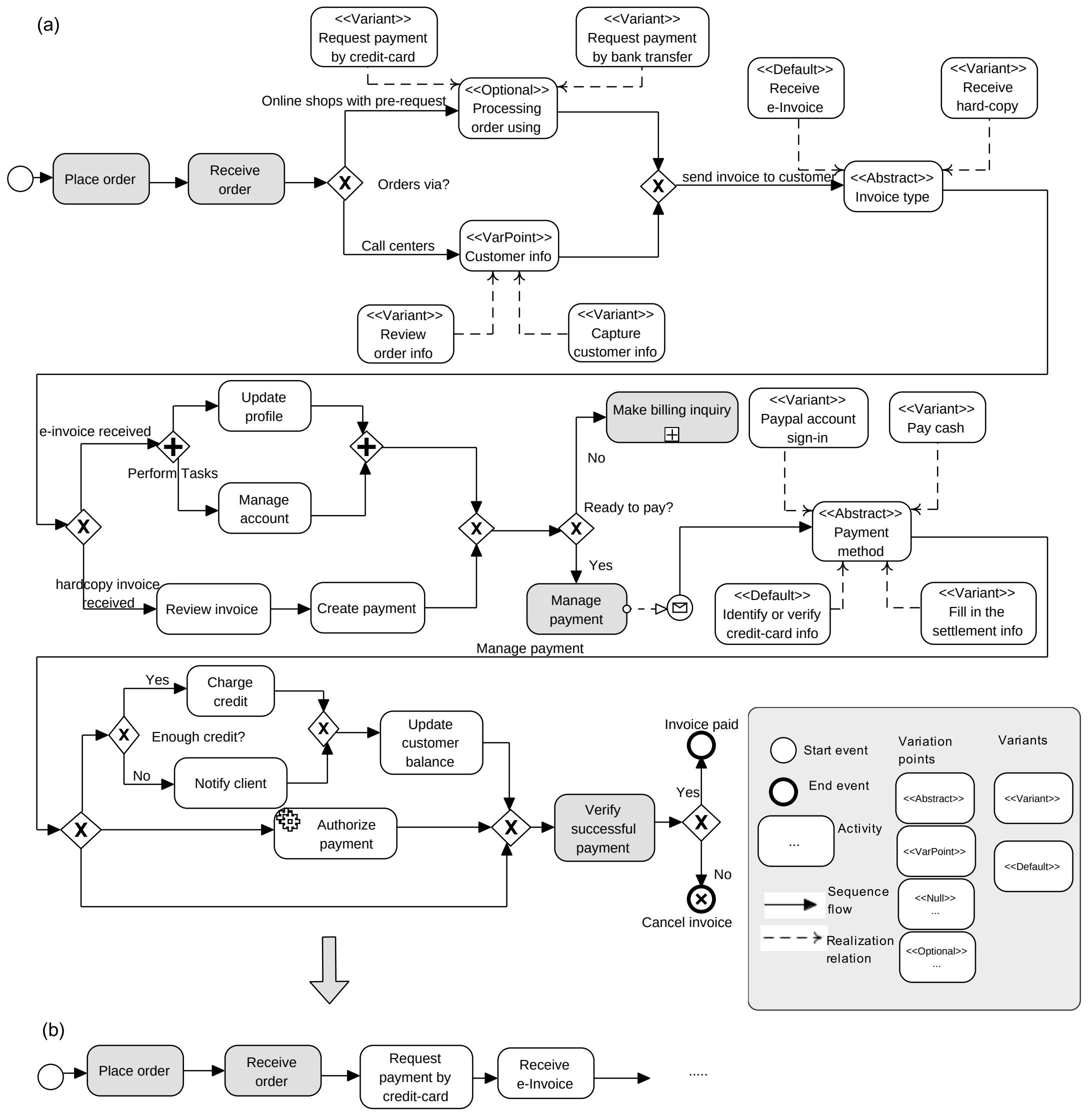


Figure 3.1: (a) Processing invoice payment example in PESOA-BPMN (b) A customized process model

A feature can be mandatory or optional (i. e., it can be deselected). It can be bound to other features via constraints (i. e., propositional logic expressions over the values of features). For example, the subfeature "Request payment by bank transfer" of "Pre-request payment" must be deselected if the subfeature "e-invoice" of "Invoice type" of "Order" is not selected. The relation between features and subfeatures is modeled using XOR (only one subfeature must be selected), AND (all the subfeatures must be selected), and OR (one or more might be selected). Even a feature is modeled as mandatory (arcs depicted with oval arrows) it can still be excluded if it has an XOR/OR relation with its sibling features. This is the case of the subfeatures of "e-invoice" which is mandatory and still is excluded in the configured feature diagram when subfeature "Pre-request by bank transfer" is selected. However, a guidance is missing on how to perform the selection of a suitable set of features.

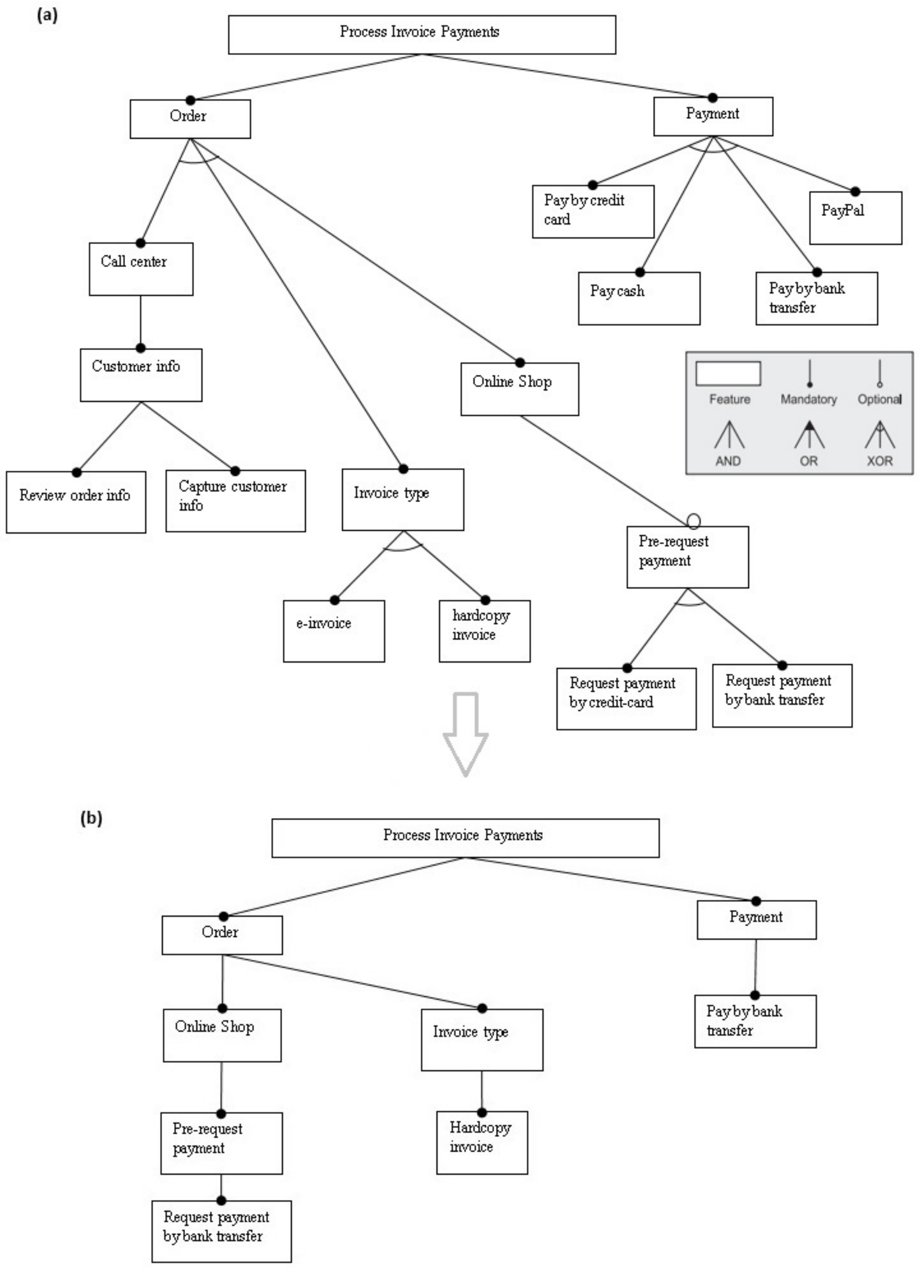


Figure 3.2: (a) A feature diagram for invoice payments (b) A possible feature configuration for invoice payments feature diagram

BPMN attributes can be parameterized to support optional, alternative or range variation points. An association marked with the stereotype *«Parameterization»* is used instead if a misinterpretation exist between the attribute and its corresponding element (Puhlmann et al., 2005). The associations are used also to link data objects that contain the possible parameters to the grouping box that surrounds the attribute, see Figure 3.3. This figure shows the parameterization of two different attribute where the lower one offers an alternative for the *Datetime* attribute of the intermediate timer event. The alternative behaviour triggers the event at the end of each month whereas the default behaviour triggers the event at the end of each quarter. In the upper side of this figure the *ConditionExpression* attribute of a sequence flow is parameterized. The default parameterized attribute *Amount* of an invoice order serves as a sentinel that activates the sequence flow if order amount is greater than a value (in this case, greater than €150). Accordingly, the sequence flow is activated and a bonus is calculated for the customer. An alternative parameterization changes this attribute to activate the sequence flow if the order amount is greater then €500.

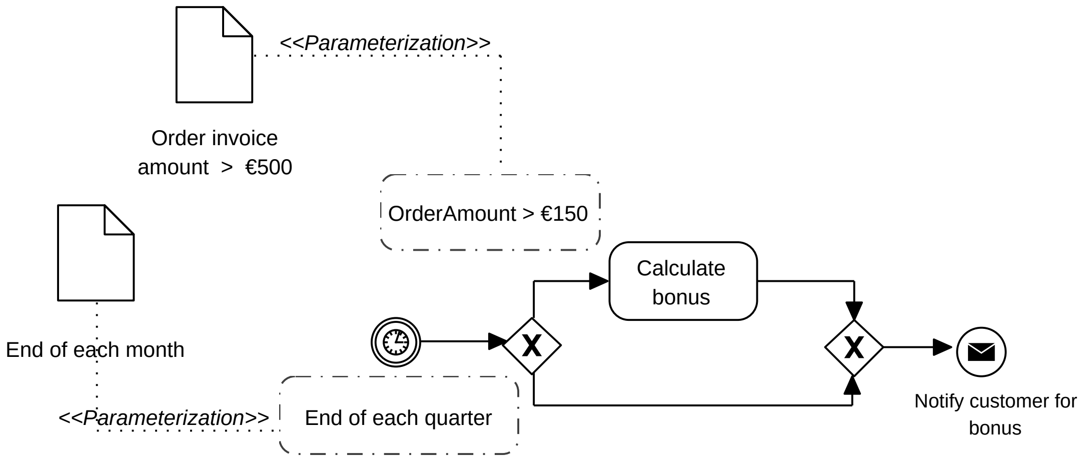


Figure 3.3: Parameterization of two different attributes in BPMN

Another BPFM (Business Process Family Model) approach classified in this group is proposed by authors (Moon et al., 2008) as a two-level approach to capture customizable process models using an extended version of UML ADs. They claim to have systematically conduct the realization of the variability in processes in different abstract levels in comparison with PESOA research project. They represent variability using not only variation point and variant but also variation point type, boundary, and cardinality. At the first level, an activity can be defined as *common* if it cannot be customized or *optional* if it can be omitted during customization. The second level selects one of the specialized variants, i. e.,concrete activities, which is represented by a variation point, i. e.,abstract activity. Variation points can be assigned only to activities. (Moon et al., 2008) identified three types of variation points (*vpType*):

(i) Boolean, exactly one variant is selected from specialization

(ii) Selection, at least one variant is selected from a number of variants denoted with a cardinality (e. g., 1..2). When more than one variant is selected the control-flow relation between selected variants should be specified (called *flow pattern*). Then, there exists different possibilities to execute the selected variants, such as: they are ordered sequentially, in in an OR decision between a XOR-split and a XOR-join, or in parallel between an AND-split and an AND-join.

(iii) Flow, a set of activities (expressed in a variant region) without a specified flow relation. To restrict the behaviour of activities in a variant region a flow pattern should be added to organize them according to the pattern, even though the precise order has to be decided by the user during customization time. Furthermore, a boundary of a variation point can be classified as either open or closed. An open boundary allows to introduce new variants during customization. In contrast, a closed boundary allows to select only the identified variants during the asset development (Moon et al., 2008).

In the following Figure 3.4 (a) the first activity "Process order" indicate an open variation point of type flow with a decision pattern between activities "Request payment by credit-card", "Request payment by bank transfer", "Review order info", and "Capture customer info" in the associated variants region (depicted with a double rectangular). This means, that during customization the arrangement of activities is left entirely to the user as it is an open boundary. Moreover, they are modeled as optional activities after applying the first-level step of this method to express the fact that they might be dropped from the customized model, as this is the case of deriving variant three in Figure 2.2. Whereas the second activity "Invoice type" indicate a boolean variation point, where only one of the variants can be selected. And finally activity "Payment method" is of type boolean. Four variants are assigned to it, i. e., "Identify or verify credit-card info", "Pay cash", "Paypal account sign-in", and "Fill in the settlement info" and only one can be selected during customization. In our example, we don't indicate an activity of *selection* type to express the fact that at least one or more variant may be assigned to it, as is the case of an OR-decision.

Figure 3.4 (b), shows a customized model in which the first variation point has been customized to a decision between variants "Request payment by credit-card", "Request payment by bank transfer" and of another parallel execution of the other two remaining variants. While the second and the last has been customized to one of the specialized activities, i. e., "Receive e-Invoice" and "Identify or verify credit-card info".

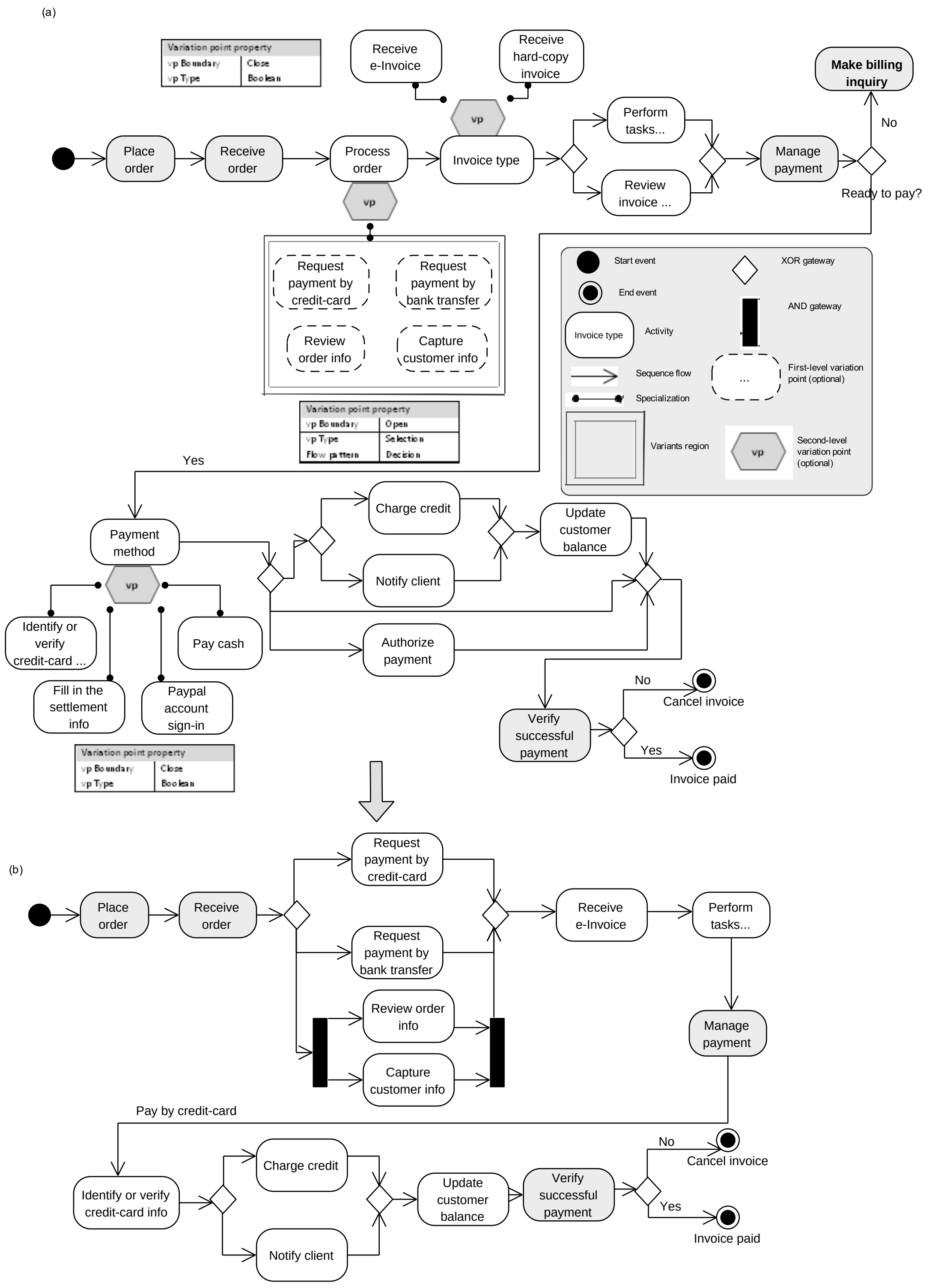


Figure 3.4: (a) Processing invoice payment example in BPFM; (b) A customized model

3.2.2 Adaptation pattern-based variability mechanism

Adapter design patterns are based on information hiding and inheritance like the Strategy design pattern(Puhlmann et al., 2005). These patterns are used to represent processes using a combination of encapsulation and inheritance between process variants. (Dőhring and Zimmermann, 2011) proposed the vBPMN(variant BPMN) approach to define the modelling of workflow variants by pattern- and rule-based adaptation in BPMN (BPMN, Spec.). Their approach consists of firstly, marking adaptive segments(variants) in a reference process, secondly, a BPMN2 adaptation pattern catalogue for realizing behaviour deviations and last, rules formulated in an event-condition-action (ECA) format applied to which adaptive segments and in what data-context. To indicate the start and end of an *"adaptive segment"* within a BPMN process definition two new nodes are introduced in vBPMN. An adaptive segment is structured as a single-entry, single exit point to facilitate the use of adaptation patterns.
Figure 3.5 shows our example of processing invoice payments with basically three adaptive segments, each of them marked between two opening/closing square brackets (depicted as intermediate events) indicating the modification of this segment using an adaptation pattern. Another annotation proposed by (Dőhring and Zimmermann, 2011) is by assigning a black diamond in the upper left corner of a single task. Each pattern consists of an implicit parameter <*AdaptationSegment*> relating to which process segment it is applied on and the workflow engine gets notified whenever an adaptive segment is entered or left by explicitly annotating tasks.

To construct new variants the connection between the values of data-context variables and process tailoring operations needs to be established. This is achieved by formulating adaptation rules in an event-condition-action (ECA) format (Dőhring and Zimmermann, 2011). Each time a token enters an adaptive segment, the context variables are evaluated and the segment potentially becomes subject to immediate adaptations before continuing through the segment.

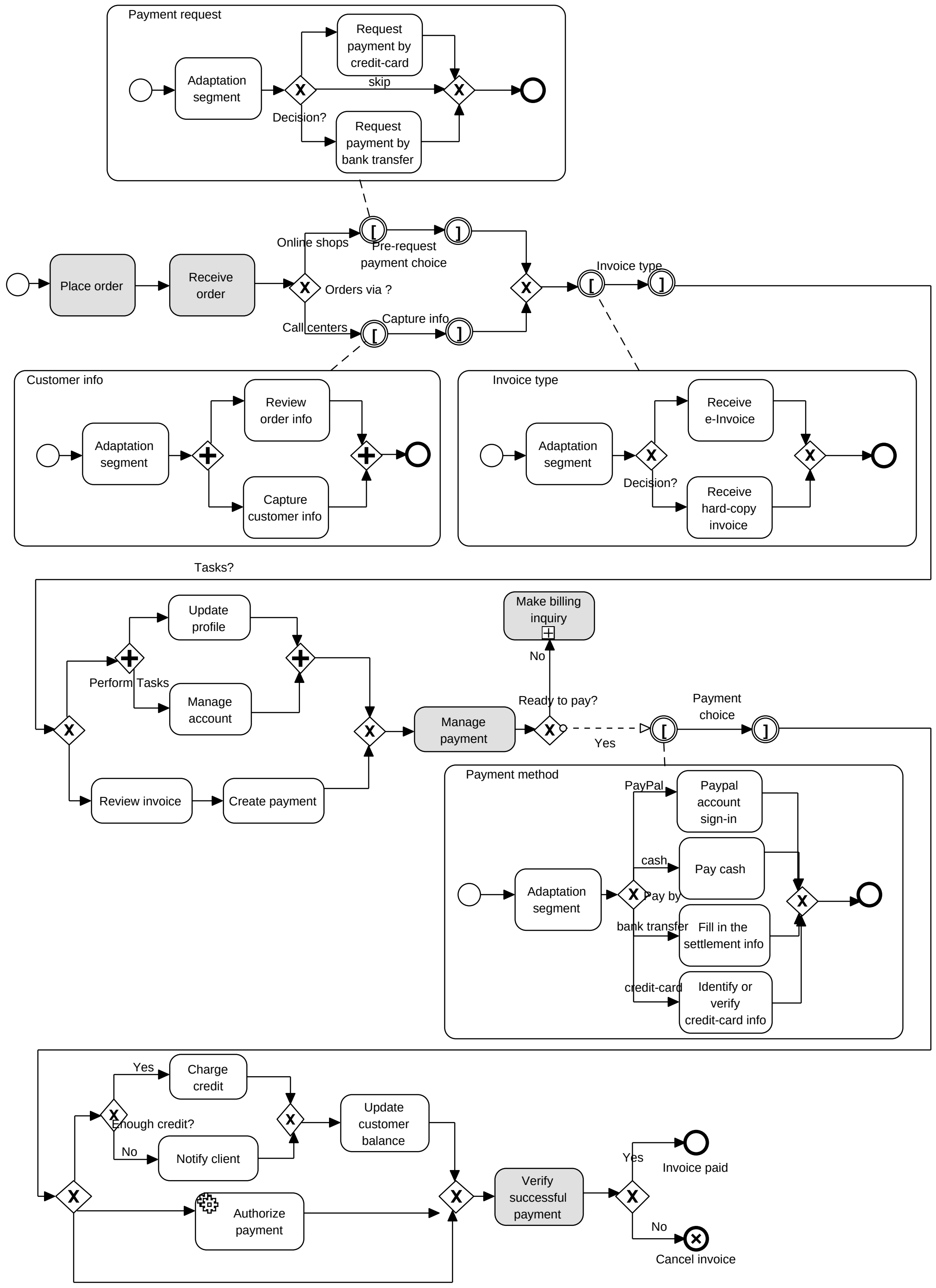


Figure 3.5: An adapted process model in vBPMN

In the following figure we show how another variant may be constructed if we annotate activity "Verify successful payment" as an "adaptive segment" to send an extra message

notification of the verified payment to "special" customers (i. e., selected with a high status) for ordering their goods/services via call centers. This is achieved by adding a time-message pattern to the adaptive segment. Another adaptation can be annotated to this adaptive segmented to add an additional task in parallel with sending a message. Parameterized patterns are applied to the adaptive segment by wrapping them around it as extensions as shown in Figure 3.6:
Rule #1: realizes the send message event contextual facet for special customers placing their orders via call centres as shown in Figure 3.6 (a).
Rule #2: inserts an additional task for example "Send advertisement" for these type of customers as shown in Figure 3.6 (b).

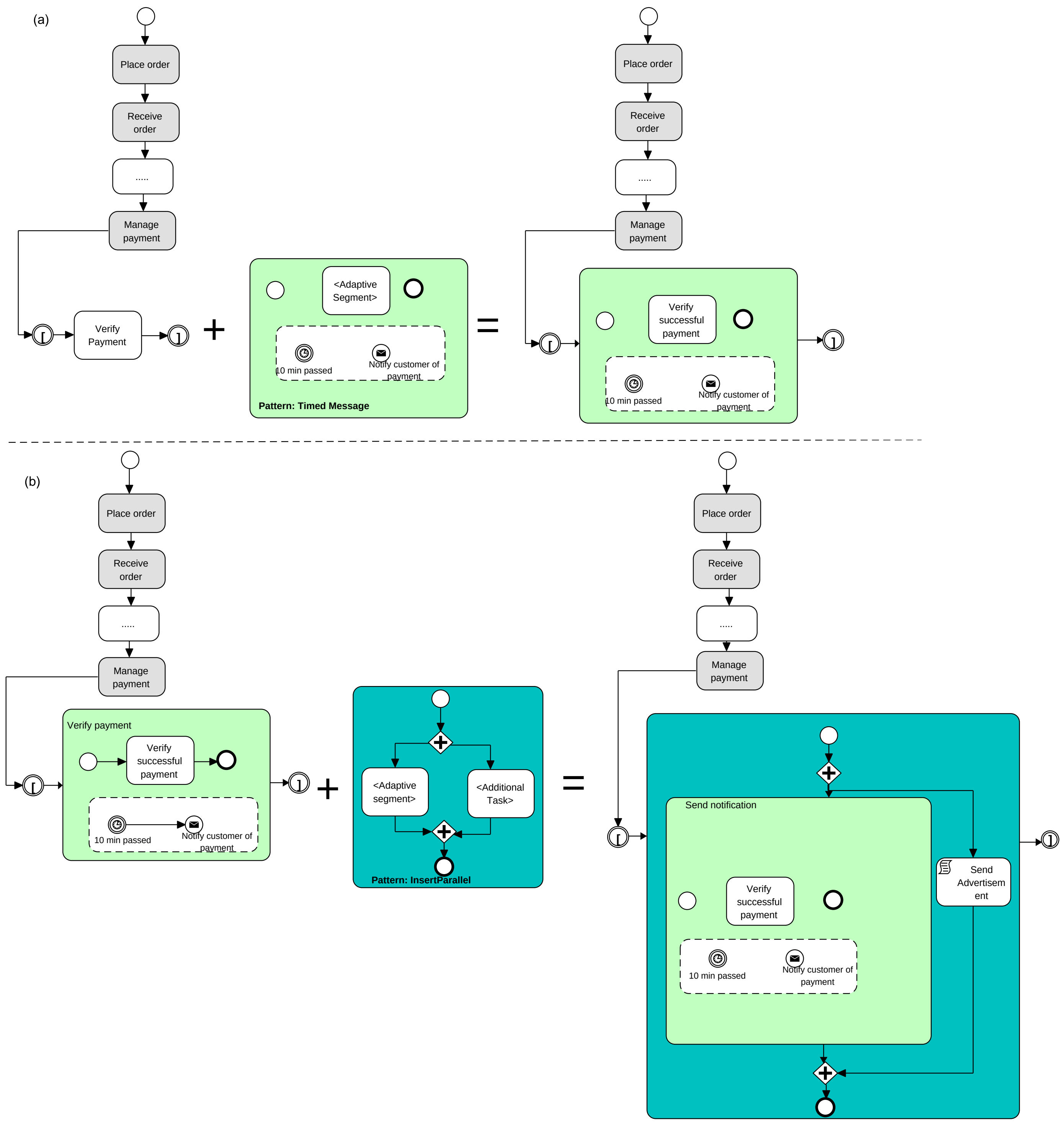


Figure 3.6: (a) Time message-pattern adaptation of a process in vBPMN; (b) Insert parallel-pattern adaptation of a process in vBPMN

RULE #1: ON verifyPayment_entry IF orderVia="CallCenter" AND customerStatus="High" THEN APPLY timedMessage(segment="Verification_entry", HandlerTask=

"VerifyPayment", time=10 min)

RULE #2: ON verifyPayment_entry IF orderVia="CallCenter" THEN APPLY insert_parallel(segment= "Verification_entry", task="Send Advertisement")

Each adaptation rule has only one context factor, which uniquely assigns the rule to a distinct process variant. However, there is no systematic way explained on how to mark these adaptive segments to capture variability on process models.

Another approach from (Hallerbach et al., 2008) is proposed, namely **Provop** (PROcess Variants by OPtions) for managing large collections of process variants in a single process. A set of change operations (i. e., insert, delete, modify and move) is used to describe the difference between *basic process model* (i. e., the most frequently executed variant of a process family or process without a specific use case) and its respective variant model. They identified requirements related to modelling of process variants, linking them to process context, executing them in WfMS, and continuously optimizing to deal with evolving needs.
Although different adaptation patterns such as: insert/delete, replace/move/swap process fragment or embedding the latter in loops, parallel or xor branches have been applied along to the entire process lifecycle (Hallerbach et al., 2010; Weber et al., 2011), they are not yet sufficient to cope with complexity of process families (Ayora et al., 2013). To this purpose, (Ayora et al., 2016) argued to address variability-specific needs of process families through change patterns that complements these adaptation patterns. Their approach namely **CP4PF** (Change Patterns for Process Families) comprise ten derived change pattern implemented in C-EPC for facilitating variability management in process families. These CPs have been grounded empirically and validated in a real scenario through a case study 'check in process' in airline industry application with the aim to considerably reduce variability management effort.

3.2.3 Template method pattern-based variability mechanism

Design patterns like 'Template Method' allows for controlling the behaviour of certain steps called 'placeholders' deferred to process runtime (Puhlmann et al., 2005). Template approach is proposed for configuring a reference process based on a set of related business process models with an a-priori known variability (Kulkarni and Barat, 2011) as well as on superimposed variants (Czarnecki and Antkiewicz, 2005). An essential BPMN meta-model is introduced by (Kulkarni and Barat, 2011) to capture the fixed behaviour (i. e., process structure) defined as a set of activities and events. A template constitutes of a control flow definition engaging fixed set of activities and events, e. g.,template1 in the following figure. A process structure is specified by multiple TKVs (i. e., a set of activity types). TKV is tuple <P, C> where P is a set of abstract activities, i. e., placeholders, and C is set of concrete activities. From TKV definition each placeholder is derived, which can be either an activity or an event. Variants assigned at a placeholder (i. e., part) are modeled explicitly using maps (i. e., a set of mappings describing fitment of a part at a placeholder) (Kulkarni and Barat, 2011)

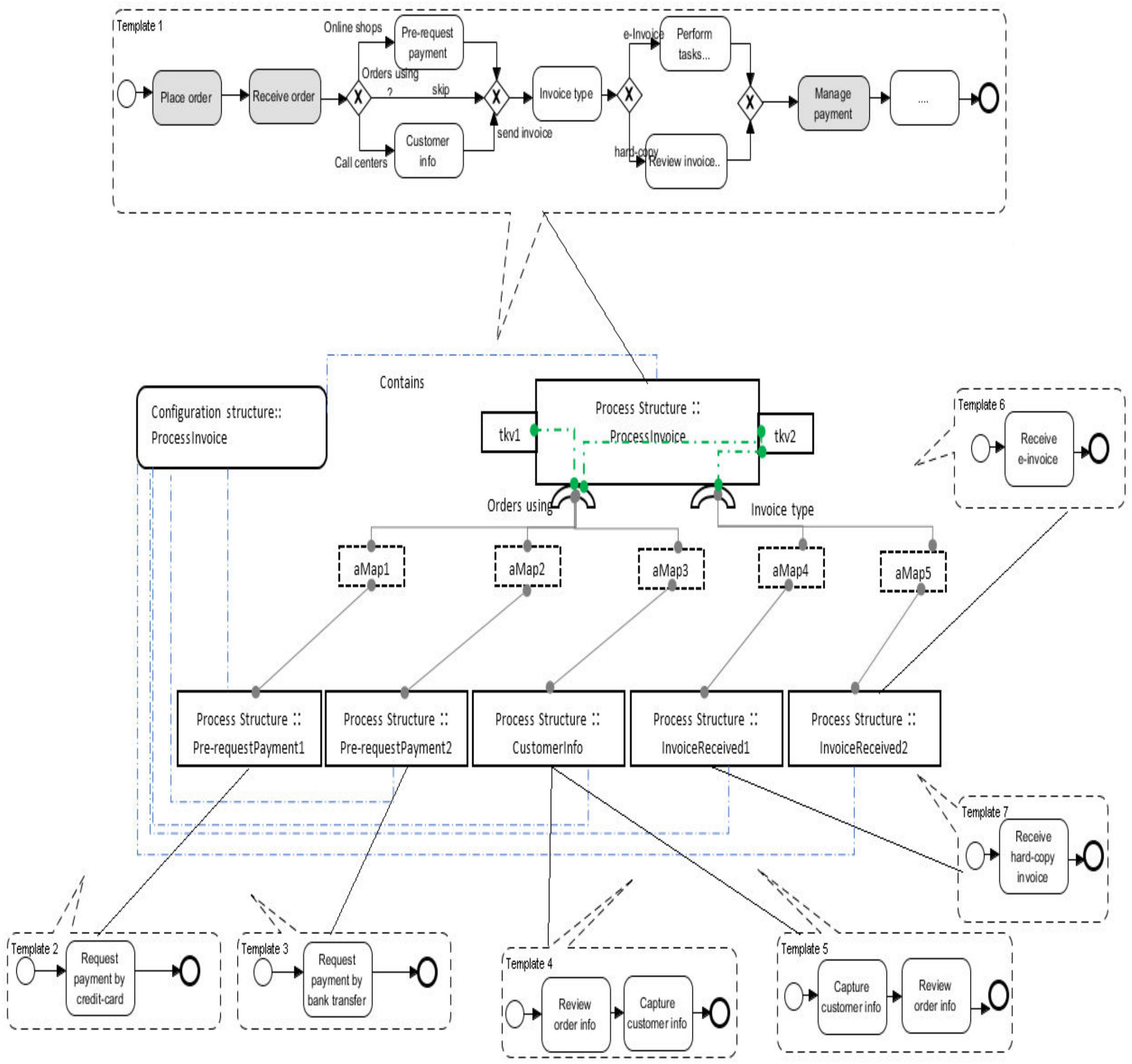


Figure 3.7: Process family of processing invoice payments

Figure 3.7 shows an excerpt of invoice payments process to illustrate variability and configurability. The process is defined as a control flow of set of activities A={Place order, Receive order, Pre-request payment, Invoice received, Perform tasks, Manage payment,..} as depicted in Figure 3.7.
A configurable process $P_{InvPayment}$ = {<E, A, template1, D, tkv1 >} of a process family PF ={< E, A, {template1}, D, TKV >} where:
A = {Place order, Receive order, Pre-request payment, Invoice type, Perform tasks, Manage payment,..},
E = {},
template1= instance of essential BPMN meta model, and
TKV = {tkv1, tkv2} where tkv1 = {P= {Orders using}, C= {A − {Orders using}}},

tkv2 = {P={Orders using, Invoice type}, C={A − {Orders using, Invoice type}}} and D=data objects.

Here, five *activity maps* are defined, e. g., aMap1, aMap2, aMap3 are specializations of process structure "ProcessInvoice" if context *Orders using* is selected. Whereas, aMap4 and aMap5 are specializations of process structure "ProcessInvoice" if context *Invoice type* is selected. Some different behavioural variances through different configurations are as follows:-

(a) configuration1=< $P_{InvPayment}$, {Pre-requestPayment1}, {aMap1}> where

Pre-requestPayment1 ={<{}, {Request payment by credit-card }, template2, D, ϕ >}

(b) configuration2 =< $P_{InvPayment}$, {Pre-requestPayment2}, {aMap2}> where

Pre-requestPayment2 ={<{}, {Request payment by bank transfer }, template3, D, ϕ >}

(c) configuration3 =< $P_{InvPayment}$, {CustomerInfo}, {aMap3}> where

CustomerInfo ={<{}, {Review order info, Capture customer info }, template4, D, ϕ >}

(d) configuration4 =< $P_{InvPayment}$, {CustomerInfo}, {aMap3}> where

CustomerInfo ={<{}, {Capture customer info, Review order info }, template5, D, ϕ >}

The same logic applies for the next context "Invoice type" as a specialization of process structure "ProcessInvoice". A Configuration structure describes the entire configuration context in terms of parts that can be fitted at placeholders. It contains different process structures, in this case six as depicted in figure below. Therefore, a configurable process with placeholders is a specialization of a template. The behaviours of configurable business process *ProcessInvoice* can differ as different parts can be fitted at defined placeholder, i. e., abstract activity *Pre-request payment* or *Invoice type*.

3.2.4 Node configuration variability mechanism

A node of a customizable process model, called configurable node is a variation point assigned to different customize options. Two main approaches fall in this group named Configurable Integrated Event-driven Process Chains (C-iEPC) and Configurable Workflows. (Gottschalk et al., 2008b; Rosa et al., 2011; van der Aalst et al., 2005; Rosemann and van der Aalst, 2007) extended the EPC language for configuring a reference process model to capture multiple process variants in a consolidated manner. Reference process models should be distinguished from so-called customizable process models. A customizable process model is a concrete process model intended for a certain context, whereas a reference process model is intended to capture common behaviour or best practices of a family of process variants(Fettke and Loos, 2003; Rosemann, 2003). In configurable workflows approach (van der Aalst et al., 2006; Gottschalk et al., 2008b) presented C-YAWL(Configurable-YAWL), an extension of the executable process modelling language YAWL[1] where variation points in a process are configured using so-called *ports*. Logic

[1]www.yawlfoundation.org

connectors (AND, XOR and OR) are integrated in each task in the form of a split (for the outgoing arcs) and a join (for the incoming arcs). In C-YAWL, like in C-EPC each feasible port variation is presented with a process fact. A C-EPC is an EPC in which functions and connectors can be marked as "configurable". A modeler can derive an individualized EPC from a C-EPC by selecting a possible variant for each configurable element(Rosa et al., 2011). As this approach doesn't present a meta-modelling solution for variants it's outside of the scope of this literature review section for a further discussion. Whereas in C-iEPC approach configurable nodes might be activities, gateways, events as well as objects and resources presented with a meta-model.

For each configurable node one customization option is selected to achieve customization. Configurable roles and configurable objects have two dimensions: optionality and specialization. If a configurable role (object) is "optional" (OPT), it can be restricted to "mandatory" (MND), or switched "OFF" to be removed from the process. If it is "mandatory" it can only be switched "OFF" (Rosa et al., 2011). There are some options for every configurable node, such as *off* option which means node(s) does not appear in the customized model or *on*, node(s) is being kept in the customized model. Therefore, configurable nodes indicate the differences between process variants. These variations in the extended notation, namely C-iEPC, are captured in the way roles and objects are assigned to activities. To maintain control-flow, resource and object perspectives synchronized is essential to prove the correctness of the individualized process model.

Our case study of processing customer invoice payments is shown in Figure 3.8 which captures all five variants modeled as separate process models (see Chapter 2) into one single process model. Here, activities and gateways (i. e., variation points) are marked as configurable with a thicker border. Configurable gateways can be customized to an equal or more restrictive gateway. A configurable OR can be restricted to an XOR or to an AND gateway or can be left as a regular OR (no restriction).

The number of an OR outgoing flows (if it is a split gateway) or the number of its incoming flows (if a join) can be restricted to any combination (e. g., two flows out of three), including being restricted to a single flow, in which case the gateway disappears (Rosa et al., 2011).

For example, we can capture the choice of processing orders via online shops or call centers by customizing the first XOR-split in Figure 3.8 or we can postpone the decision till runtime. If the choice is "online shops" we can restrict this gateway to the outgoing flow leading to the event "Invoice reviewed". As a result, the branch starting with the sequence flow "Call centers" is removed, and vice versa. Configurable activities can be kept on or switched off. If switched off, the activity is simply hidden in the customized model. In addition, they can be customized to optional. The choice of whether to keep the activity or not is deferred until runtime. For example, the function "Request payment by credit-card" and "Request payment by bank transfer" are configurable nodes in Figure 3.8; thus, we can switch them off for those orders received via online shops for which a pre-request payment is not required. Configurable elements might be resources (called roles in C-iEPCs) and objects, too. (Rosa et al., 2011) propose to use logical gateways so-called range connector (i. e., XOR, OR and AND) that allow any combination of the resources and objects connected to activities modeled by a pair of natural numbers, e. g., lower bound (2) and upper bound (5), which means at least 2 and at most 5 resources.

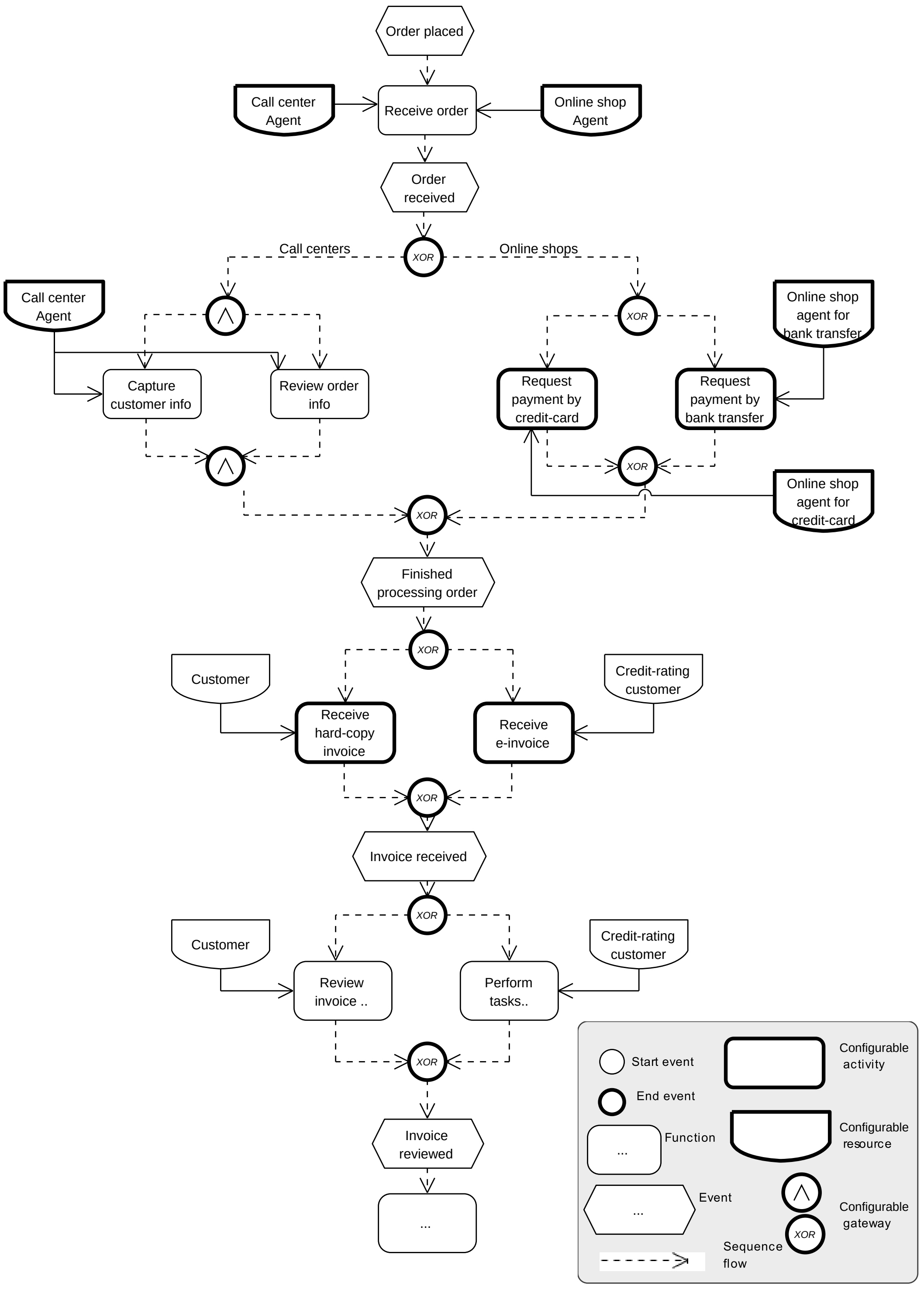


Figure 3.8: The C-iEPC model representing all invoice payment variants

For simplicity, Figure 3.8 depicts only three resources marked as configurable nodes with a thicker border meaning that during customization they can be configured to one of the

specialized resources. However, it's out of our scope to demonstrate how the specialization of resources assigned to activities is achieved.
A formalized algorithm with a proven theory is presented to guarantee the correctness of the individualized process model (i. e.,an iEPC with non relevant options being removed) derived from a configurable process model with respect to a valid configuration.

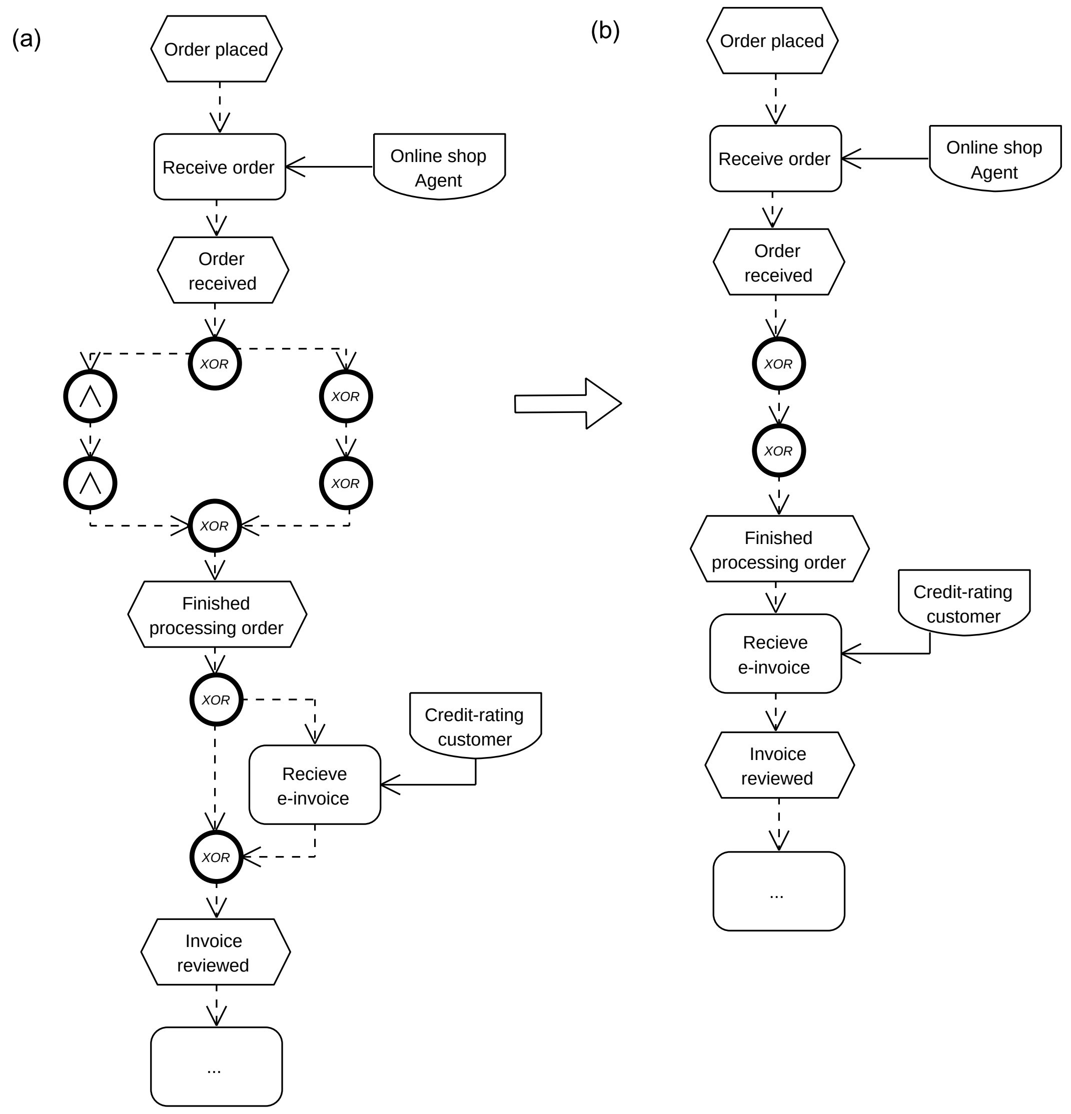


Figure 3.9: The application of the individualization algorithm to a fragment of processing customer e-invoice process model of Figure 3.8

Accordingly, functions "Capture customer info", "Review order info", "Request payment by credit-card", "Request payment by cash", "Review invoice" have been switched *OFF* and thus they have been replaced by an arc. The resulted model is shown in Figure 3.9 (a) whereas (b) is the result after applying the last step of the individualized algorithm where SESE control-flow connectors are replaced with arcs (this may result in consecutive events and consecutive functions to be interleaved with events have to be removed). C-iEPCs do not provide any execution support, they are formally defined in (Rosa et al., 2011). An iEPC model is derived from a C-iEPC using an individualization algorithm. In the customized model all nodes that are no longer connected to the initial and final events via a path are removed and the remaining nodes are reconnected to preserve (structural and behaviour) model correctness. To capture domain properties and their values a questionnaire is linked to configurable nodes of C-iEPC, supported by Synergia[2] and Apromore[3] toolsets. The resulting customized models have been validated via a case study in the film industry.

Instead, we design the questionnaire model based on their proposal to fit our example. It captures processing invoice payment properties as shown in Figure 3.10 which comprise a set of features called domain facts organized into questions. All questions and facts have been assigned a unique identifier. A domain fact has a default value which is the most common choice for that fact, e. g., *f5:e-invoice* as most of the invoice payments are e-invoice, then we can assign a false value to the other fact *f6:hardcopy invoice*. Moreover, if a domain fact needs to be explicitly set when answering the question it is marked as mandatory. Otherwise, if a fact is left unset for the corresponding question then its default value can be used to answer the question or it is skipped. In a questionnaire model an order is established for posing questions to users in contrast with the feature model. This is achieved via order dependencies. There are two types of dependencies: *full* and *partial* dependency. For example, *q2* can be posed after *q1* is answered, this is expressed via the partial dependency between *q1*, *q2* depicted with a dashed line in Figure 3.10. Whereas, a full dependency, e. g., *q4* is posed after *q1* AND *q3* is answered to capture the mandatory precedence in order to give priority to the most important questions.

[2]www.processconfiguration.com
[3]www.apromore.org

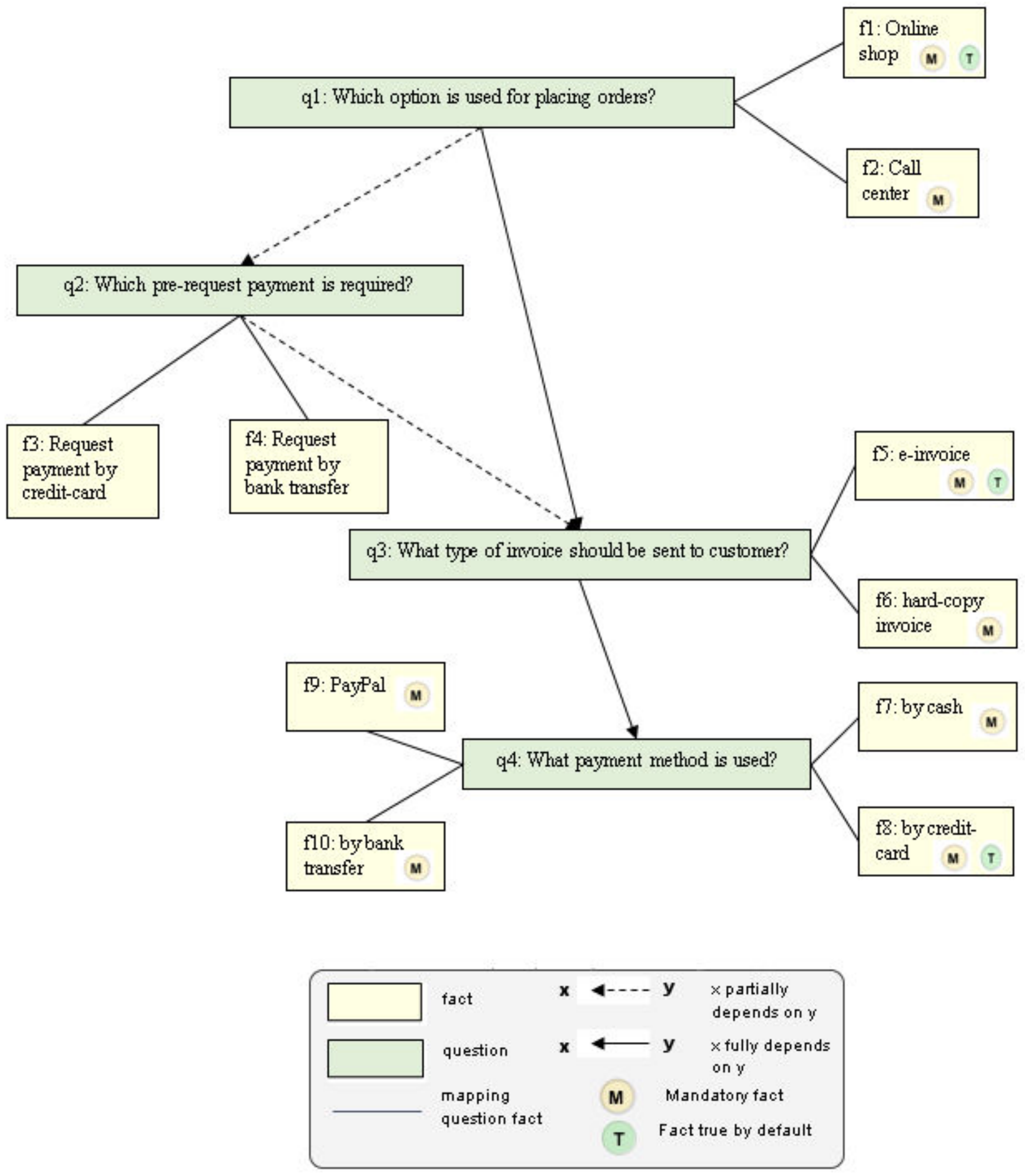

q1: Which option is used for placing orders?
f1: Online shop M T
f2: Call center M
q2: Which pre-request payment is required?
f3: Request payment by credit-card
f4: Request payment by bank transfer
q3: What type of invoice should be sent to customer?
f5: e-invoice M T
f6: hard-copy invoice M
q4: What payment method is used?
f7: by cash M
f8: by credit-card M T
f9: PayPal M
f10: by bank transfer M
fact
question
mapping question fact
x y x partially depends on y
x y x fully depends on y
M Mandatory fact
T Fact true by default

Figure 3.10: An extract of the questionnaire model for configuring e-invoice type process model

After the clear overview of current process meta-modelling approaches we propose a novel meta-model for capturing business processes with variants. In the following Section 3.3 we give a detail description about our generic meta-model.

3.3 A novel process meta-model

In this section we present our method to deal with process variants, specifically we design a meta-model to adequately capture process variants by introducing two new notions of generic activities and generic processes and to define specialization/generalization relationships between them. We depict an excerpt of the core of our meta-model using UML class diagram as shown in Figure 3.11 that captures process model elements and their variants from a design-time perspective. A detail description of all its elements is as follows.

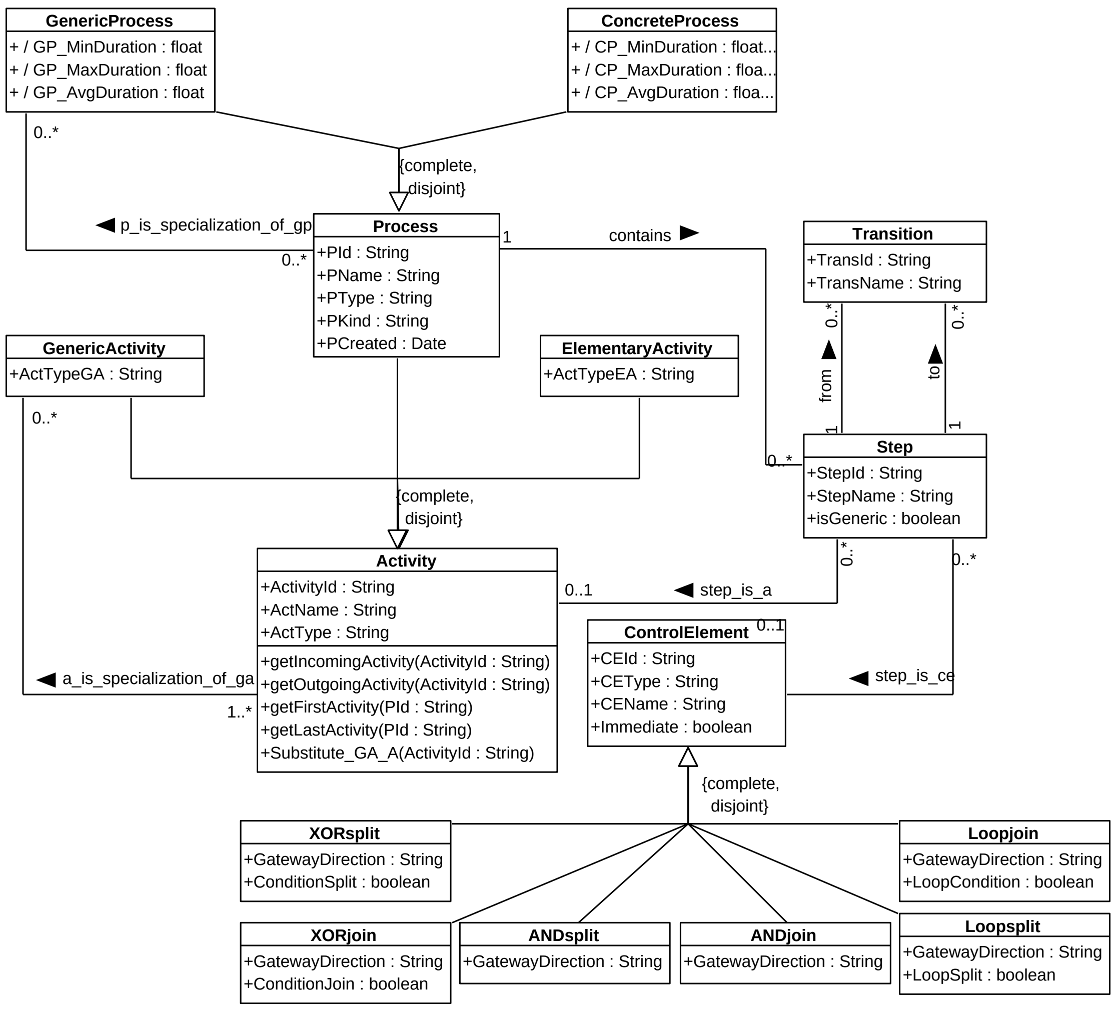


Figure 3.11: An excerpt of process meta-model capturing modelling elements

(Eder et al., 2002) defined a process as a collection of activities (i. e., individual steps in a business process), participants (i. e., software systems or users responsible for the enactment of activities), and dependencies (i. e., the order of activities and the data flow between them) between activities. We use the same concept to identify a *concrete process*(CP) in a process definition which consists of many steps logically related in order to achieve a common goal and represent it by *ConcreteProcess* class in our model. A *Step* which corresponds

to a *FlowNode* in BPMN specification (BPMN, Spec.) is a concrete invocation of activity in a process and it can be identified either as an *Activity* (e. g., activity *Place order in our running example*) or as a *ControlElement* (e. g., *Pay method* decision gateway in variant 4 of Figure 2.3 Section 2). Each step has many different predecessors (*from*-relationship) and successors (*to*-relationship), which is expressed by the association class *Transition. Control element* (i. e., gateways) step controls how the transitions interact as they converge and diverge within a process. It's represented by the superclass *ControlElement* specialized to subclasses based on different gateway types: *XORSplit*, *XORJoin*, *ANDSplit*, *ANDJoin*, *LoopSplit*, *LoopJoin*. Each of them represents a point (situation) within the process where one or several activities are chosen or completed based on a decision depending on the gateway type. For example, the control element steps *Perform tasks* and *Enough credit?* in our variants represent specific objects of the respective *AndSplit*, *XORSplit* classes as specialized ones in our model. And in process model variant 3 of Figure 2.2, *Perform tasks* has one predecessor activity *Receive e-invoice* represented by the incoming transition (depicted with a solid arrow) and two successor activities *Manage account* and *Update profile* represented by two outgoing transitions. Each transition (i. e., a sequence flow) has only one source and only one target step.
An *Activity*, the smallest unit of work that a company performs can be of the following subtypes: *elementary activity*, *generic activity* or *process* (sub-process), where each of these elements are represented through respective classes in the meta-model in a generalization relationship (Figure 3.11). We decide to model these activities in a generalization relationship to represent the fact that certain associations e. g., *a_is_specialization_of_ga* can be applied to only some of the objects (member of subclass *GenericActivity*) of super class *Activity*.
An elementary activity so-called a task is an atomic/uncompounded activity, e. g., activity *Place order* in all variants. A task can be of different types, such as manual, script, service, send task etc. A sub-process (complex activity) is composed of other activities, e. g., subprocess *Make billing inquiry* in Figure 2.2 is composed of activities e. g., *Send e-mail and/or copy invoice*, *Place call to inquire invoice or account* etc. Attribute *ActType* in this class stores such information about types of activities. Within a complex activity the same activity may appear different times, where every of those appearances can be unambiguously identified by the concept of the *Step*.

Here, we introduce two notions: generic activities and generic processes expressed by the respective classes *GenericActivity* and *GenericProcess*.
A *generic activity* (GA) is defined as a step in a process that might be realized by different activities. To identify these steps in our meta-model we add a boolean type attribute named *isGeneric* with a true value otherwise false. These activities might be single elementary activities or so-called *customized subprocess* (might contain only a few elementary activities in a specific order). For example, generic activity *GA1: Pre-request payment* as depicted with a thick border in Figure 3.17 might be realized by one of the specialized elementary activities e. g., *E: Request payment by credit-card* or *F: Request payment by bank transfer*. In addition, *GA2: Customer info* might be realized by one of the specialized *customized subprocess* e. g., *SP1: SubP_CseqD* or *SP2: SubP_DseqC* or *SP3: SubP_-CparD*. Connectors between specialized activities and generic activities are annotated with a dotted line arrowhead. In the meta-model a generic activity and its specializations are related by

a_ is_ specialization_ of_ ga relationship as shown in Figure 3.11. For example, elementary activities *H: Receive e-Invoice* and *I: Receive hard-copy invoice* in Figure 3.17 are specializations of the generic activity *GA3: Invoice type.* We annotate these specialization relationships by using a stereotype *«variant_ specialization»* in the *Generic Connector* line with the arrow-head directed to the generic step.

A formal definition about specialization of generic activities is in the following section 3.3.1.
A *generic process* (GP) is defined as a process that contains at least one generic activity, e. g., *GP_ Pay* consists of generic activity *GA4: Payment method* whereas *GP_ Receive* consists of generic activities *GA1: Pre-request payment*, *GA2: Customer info* and *GA3: Invoice type* as shown in Figure 3.17. We add an attribute *PKind* to store values for a process, such as *GP, CP, SubP* if process is generic, concrete or a subprocess. Whereas attribute *PType* stores values such as *public, private or unspecified* for a process. We model the process as a subclass of activity to express the fact that a process is a subprocess for activity, so different from the common modelling in BPMN where a subprocess is *part-of* process. CP and GP are modeled as disjoint subtypes of *Process* supertype class, thus a CP cannot contain any GA.
A process P is a specialization of a generic process G, if P can be derived from G by substituting one of the generic activities g of G with one of g's specializations. For example, the process *PayInvoice* is a specialization of the generic process *GP_ Pay.* A formal definition about specialization of generic processes is in the following section 3.3.2.

Additional elements captured from organizational perspective (i. e., resource perspective) are: Participant, User, Role, OU, OU_hierarchy as depicted in Figure 3.12 and an example of a possible structure of an agency organization is shown in Figure 3.13 to express how users responsible for specific tasks are positioned in an organization unit.
Participant consists of a resource that performs many *steps*, i. e., *activities.* Participant might be *User* (individual/registered users of the system) or *Role* (logical description of a position in an OU).

Instead of using a ternary relationship between *User*, *Role* and *OU (OrganizationUnit)* to express the fact that a user has a specific role in a specific OU, we model it as a binary relationship adding an extra class named *UserRole.* Furthermore, users, roles or users with roles can participate in different processes for different OUs. It is not only possible to have assignments of activities through roles but also a direct assignment to users.

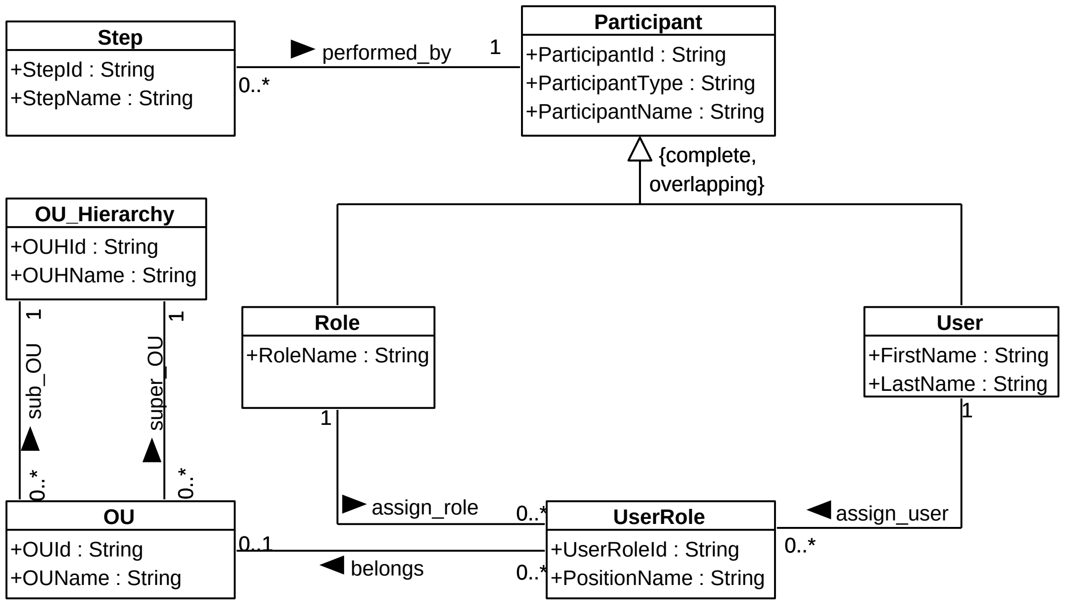


Figure 3.12: An excerpt of process meta-model capturing resource perspective elements

The hierarchy structure of an organization is defined where an organization unit (Department) can have many other sub units (other sub departments). For example, for processing e-invoices received via a call center agent named "Ana" with role "Order manager" under "Financial Management" department is assigned to perform activity "Receive order". Other participants assigned with specific roles might belong to different departments.

Other important elements from run-time perspective as shown in Figure 3.14 store information about instances of different process variants after executing the latter with a BPMS. *ProcessInstance*, *StepInstance*, *ActivityInstance*, *CEInstance* are used for the representations of the instances of *processes*, *steps*, *activities*, and their *control elements* during runtime.

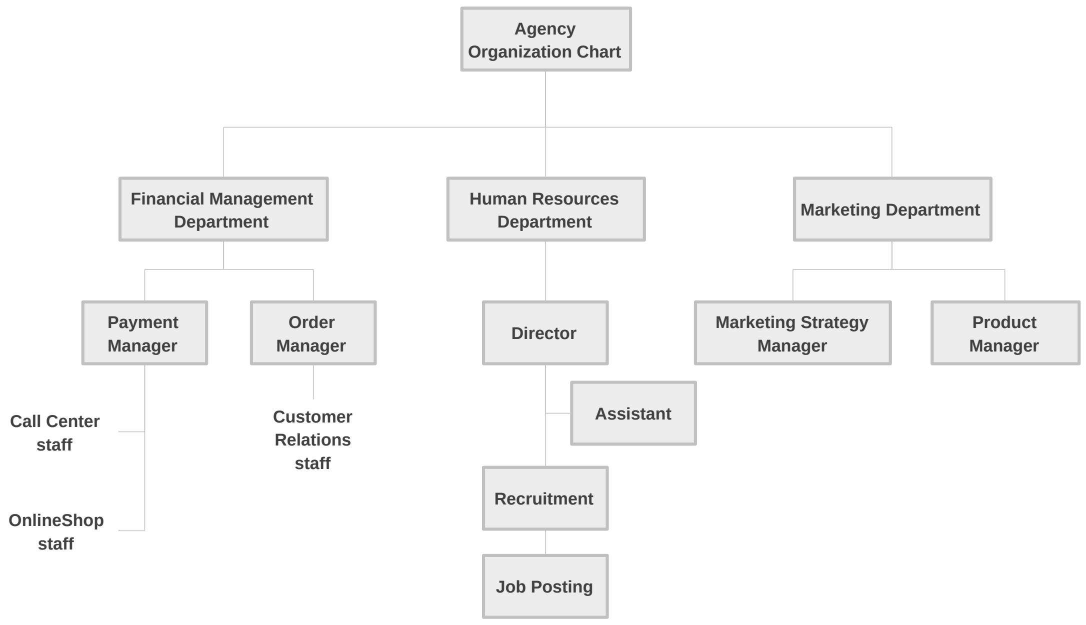


Figure 3.13: Agency organization structure example

Analogous to the step in the process model, *step instance*(see Table 3.2) consists of either *activity instances* or *control element instances*. Many specific *events* occur in a step instance, e. g., start event *Place order: assign* with specific timestamps (i. e., 2018-07-05 15:00:00.000). We focus on limiting two type of events: only *assign* (start) and *complete*(end) events. We capture all activity instances from the event logs generated from each process variants execution. For example, activity instance *Place order* of process instance "1" (variant 4) with an execution time (i. e., 20 minutes) calculated by subtracting timestamps of end event (i. e., 2018-07-05 15:20:00.000) from timestamps of start event *Place order*. Furthermore, a process instance consists of many step instances and belongs always to exactly one concrete process. We might have the same process instance id (during runtime) for different processes; to store these kind of information we set a composite primary key (*ProcessInstanceId*, *Concrete_Process_Id*) for class *StepInstance*.

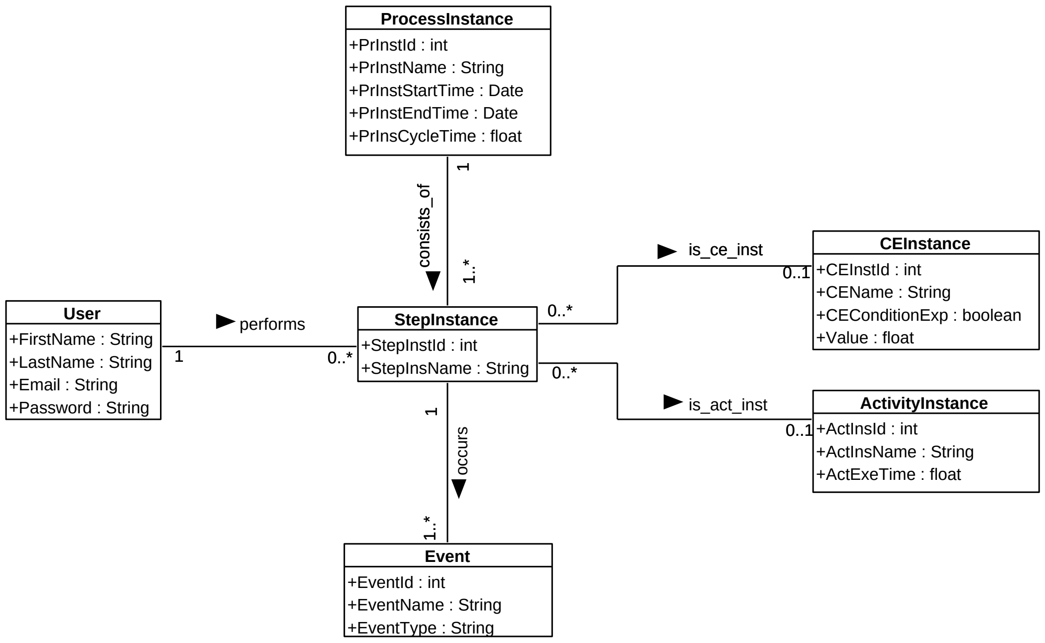


Figure 3.14: An excerpt of process meta-model capturing instance elements

We unify all above diagrams into one class diagram as shown in Figure 3.15. We model the relationships between design-time and run-time elements, e. g., many-to-one relationship between *ActivityInstance* and *Activity* to express the fact that an activity can have many instances and one activity instance belongs to exactly one activity. Analogously we model relationships between *Step, StepInstance* and *ControlElement, CEInstance*. Furthermore, we specify each process instance to which process they belongs. An example of process instance data is as follow.

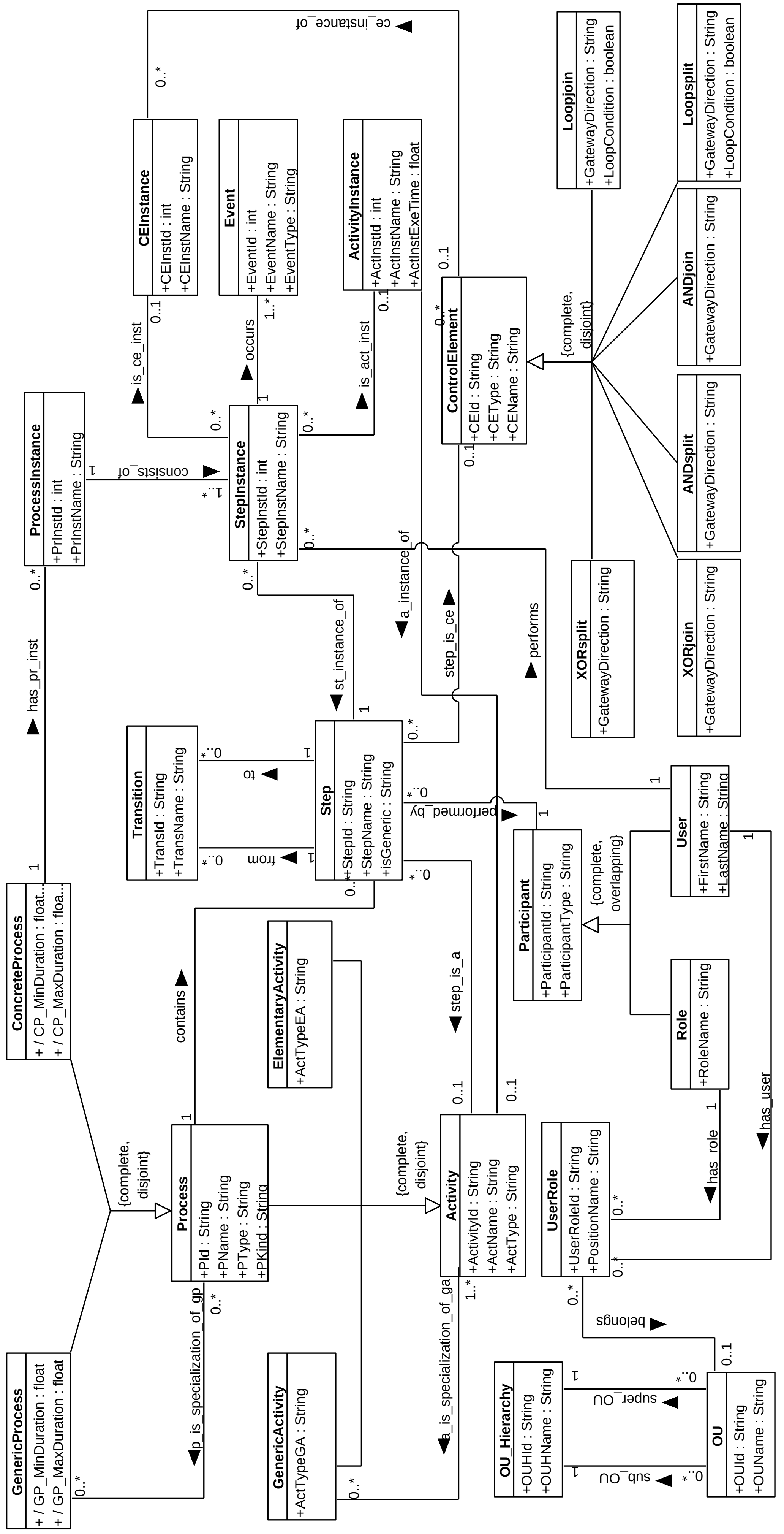


Figure 3.15: A full process meta-model capturing process variants and their instances

In the following tables we give some records on how these information from our running example is captured in our conceptual data model. In Table 3.1 and Table 3.2 we show records of events data and step instances respectively from process variants executions. Multiple occurrences of the same event are recorded but that belongs to a different step instance. And the same step instance might have two occurrences of the same event but of different types, e. g., "assign" and "complete" as displayed in Table 3.1.

Table 3.1: An excerpt of event data

EventId	EventName	EventType	EventTimestamp	StepInstanceId
1	start order	assign	2018-06-22T16:00:00+02:00	1
2	start order	complete	2018-06-22T16:00:00+02:00	1
3	Place order	assign	2018-06-22T16:00:00+02:00	2
4	Place order	complete	2018-06-22T16:02:00+02:00	2
5	Receive order	assign	2018-06-22T16:02:00+02:00	3
6	Receive order	complete	2018-06-22T16:04:00+02:00	3
7	Request payment by credit-card	assign	2018-06-22T16:04:00+02:00	4
8	Request payment by credit-card	complete	2018-06-22T16:07:00+02:00	4
9	Receive e-Invoice	assign	2018-06-22T16:07:00+02:00	5
10	Receive e-Invoice	complete	2018-06-22T16:17:00+02:00	5
11	Manage account	assign	2018-06-22T16:17:00+02:00	6
12	Update profile	assign	2018-06-22T16:17:00+02:00	7
13	Manage account	complete	2018-06-22T16:19:00+02:00	6
14	Update profile	complete	2018-06-22T16:22:00+02:00	7
[..]	[..]	[..]	[..]	[..]
661	Receive order	assign	2018-06-22T11:07:00+02:00	331
662	Receive order	complete	2018-06-22T11:09:00+02:00	331
663	Request payment by credit-card	assign	2018-06-22T11:09:00+02:00	332
664	Request payment by credit-card	complete	2018-06-22T11:12:00+02:00	332
665	Receive e-Invoice	assign	2018-06-22T11:12:00+02:00	333
666	Receive e-Invoice	complete	2018-06-22T11:22:00+02:00	333
[..]	[..]	[..]	[..]	[..]

Whereas, a process instance comprise the path of executed step instances that are traceable through the network of sequence flows, gateways and activities within a particular process. Also, there are multiple occurrences of the same step instance within different process instances of the same process. In addition, the same step instance name can be executed for different processes, e. g., *Identify or verify credit-card info* step instance in process *process-BP4752* and *process-BP4746* as records displayed in Table 3.2.

Table 3.2: An excerpt of step instances data

StepInsId	StepInsName	StepInsExeTime	StepId	ProcessIns	ProcessId	ProcessName
1	start order	0	BP4745_BP2335_BP2332	1	process-BP4745	Receive Invoice-Variant 1
2	Place order	120	BP4745_BP2335_BP2407	1	process-BP4745	Receive Invoice-Variant 1
3	Receive order	120	BP4745_BP2349_BP2407	1	process-BP4745	Receive Invoice-Variant 1
4	Request payment by credit-card	180	BP4745_BP2349_BP2409	1	process-BP4745	Receive Invoice-Variant 1
5	Receive e-Invoice	600	BP4745_BP2335_BP2279	1	process-BP4745	Receive Invoice-Variant 1
6	Manage account	120	BP4745_BP2335_BP2283	1	process-BP4745	Receive Invoice-Variant 1
7	Update profile	300	BP4745_BP2335_BP2335	1	process-BP4745	Receive Invoice-Variant 1
8	Manage payment	240	BP4745_BP2335_BP2416	1	process-BP4745	Receive Invoice-Variant 1
[..]	[..]	[..]	[..]	[..]	[..]	[..]
13	Receive e-Invoice	600	BP4745_BP2335_BP2279	2	process-BP4745	Receive Invoice-Variant 1
[..]	[..]	[..]	[..]	[..]	[..]	[..]
855	Identify or verify credit-card info	120	BP4746_BP2335_BP2279	1	process-BP4746	Pay Invoice-Variant 1
[..]	[..]	[..]	[..]	[..]	[..]	[..]
3301	Receive e-Invoice	60	BP4751_BP2335_BP2279	88	process-BP4751	Receive Invoice-Variant 3
[..]	[..]	[..]	[..]	[..]	[..]	[..]
3401	Identify or verify credit-card info	50	BP4752_BP2335_BP2279	3401	process-BP4752	Pay Invoice-Variant 3
[..]	[..]	[..]	[..]	[..]	[..]	[..]

3.3.1 Specialization and generalization of activities

In this section we argue about specialization and generalization between activities after defining generic activities. We define generic activities (depicted with bold line rounded rectangular) as typical places where variation occurs among process variants. We annotate each of these connections between generic activities and specialized activities (either elementary or subprocess) with *variant_ specialization* stereotype, to distinguish from the 'normal' sequence connector in process modelling. This identification procedure usually is assumed to be done by process analysts or designers. Here, we define ourselves where these variations occur between variants and consequently we define the generic activities for our running example. In figure Figure 3.16 we define generic activities as steps in a process that might be realized by different activities.

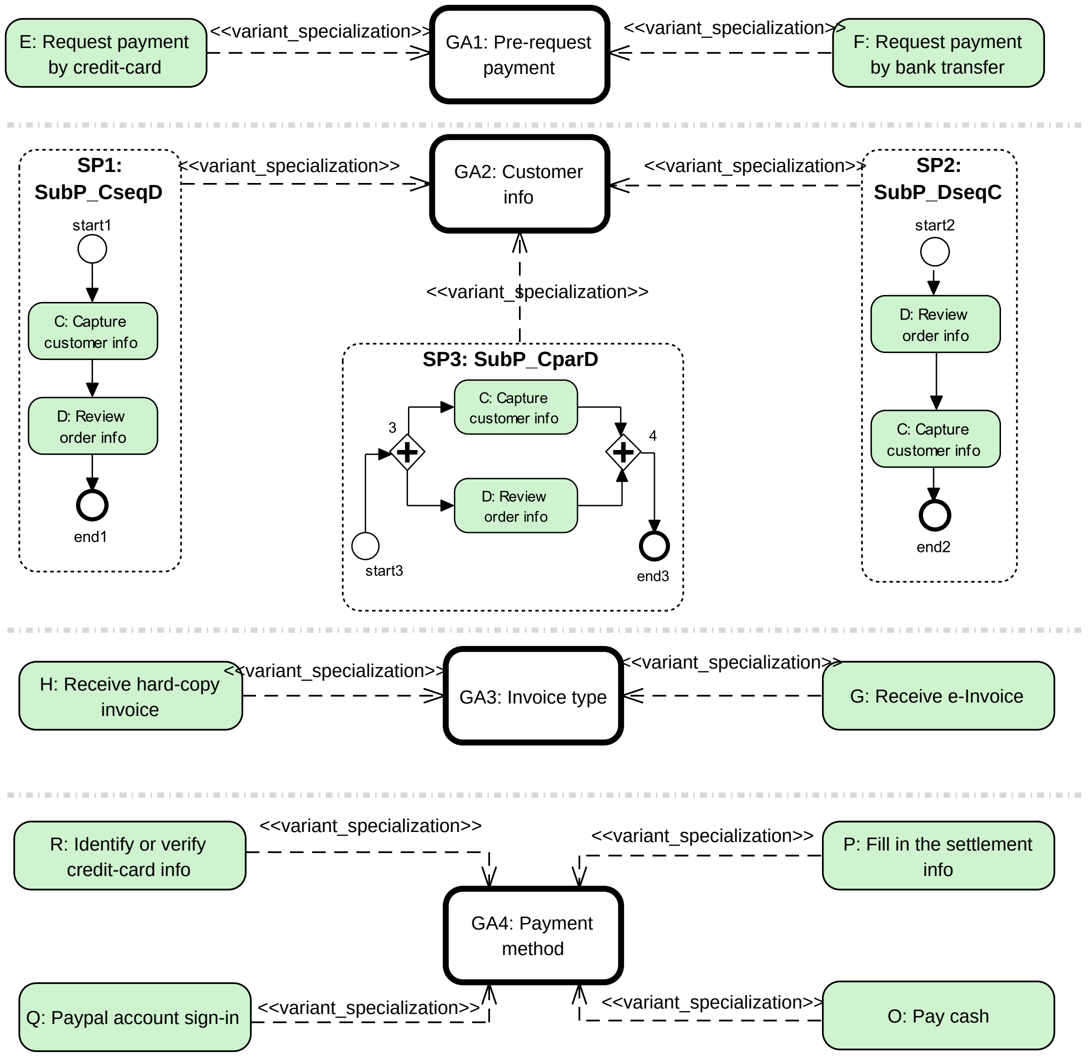

E: Request payment by credit-card
<<variant_specialization>>
GA1: Pre-request payment
<<variant_specialization>>
F: Request payment by bank transfer
SP1: SubP_CseqD
start1
C: Capture customer info
D: Review order info
end1
<<variant_specialization>>
GA2: Customer info
<<variant_specialization>>
SP2: SubP_DseqC
start2
D: Review order info
C: Capture customer info
end2
<<variant_specialization>>
SP3: SubP_CparD
3
C: Capture customer info
D: Review order info
4
start3
end3
H: Receive hard-copy invoice
<<variant_specialization>>
GA3: Invoice type
<<variant_specialization>>
G: Receive e-Invoice
R: Identify or verify credit-card info
<<variant_specialization>>
<<variant_specialization>>
P: Fill in the settlement info
GA4: Payment method
Q: Paypal account sign-in
<<variant_specialization>>
<<variant_specialization>>
O: Pay cash

Figure 3.16: Specialization of generic activities to elementary activities

Specifically, the specialization between a generic activity to its activities (either elementary activities or customized subprocesses) is as follows:

- Generic activity *GA1: Pre-request payment* might be realized by one of the specialized activities e. g., *E: Request payment by credit-card* or *F: Request payment by bank transfer.*
- Generic activity *GA2: Customer info* might be realized by one of the subprocesses that contains some activities in a sequential order or some interleaved activities between two parallel branches. Specifically, custom subprocesses named *SP1: SubP_-CseqD* (see Figure 3.16) starts by performing *C: Capture customer info* elementary activity and after its completion elementary activity *D: Review order info* is started. Whereas, the customized subprocess named *SP2: SubP_DseqC* performs the above elementary activities in the contrary order. We define these subprocesses as *Abstract* type to express the fact that they can replace another activity or process. Furthermore, *SP3: SubP_ CparD* contains interleaved activities/events between parallel branches. In this latter case, event patterns such as *C-started, D-started, C-completed, D-completed* or *C-started, D-started, D-completed, C-completed* or *D-started, C-started, C-completed, D-completed* or as a last possible combination *D-started, C-started, D-completed, C-completed* might be revealed and the choice is deferred until runtime.
- Generic activity *GA3: Invoice type* might be realized by one of the specialized activities e. g., *H: Receive hard-copy invoice* or *G: Receive e-Invoice.*
- Generic activity *GA4: Payment method* might be realized by one of the specialized activities e. g., *O: Pay cash* or *P: Fill in the settlement info* or *Q: Paypal account sign-in* or *R: Identify or verify credit-card info.*

The specialization is accomplished using substitution function as follows:

Definition 3.3.1. *A substitution* $\Theta : GA \rightarrow A \cup P$ *is a function that replaces the occurrences of a generic activity GA by the occurrences of another activity A or process P, where A is a set of activities (either elementary or generic) whereas P is a set of processes (either concrete or generic). Accordingly,* $\Theta(GA) = A \ \vee \ \Theta(GA) = P$.

For each realization of a generic activity to one of the specified activities every occurrence of a generic activity is substituted with the occurrence of the respective activity of a concrete process. Consequently, from definition of the substitution function we can define specialization relationship between activities as follows:

Definition 3.3.2. *An activity a is a specialization of a generic activity g* $\ iff \ \exists\Theta$, *such that* $g(\Theta) = a$.
Equally, a generic activity g is a generalization of an activity a $iff \, \exists\,\Theta$, *such that* $a(\Theta) = g$

For example, activity *G: Receive e-Invoice* is a specialization of the generic activity *GA3: Invoice type* if we substitute generic activity *GA3: Invoice type* we get activity *G: Receive e-Invoice.*

In Figure 3.17 we give an illustration of customizable invoice payment process as a reference (or so-called configurable) process model with generic activities (i. e., variation points). We assign each activity an uppercase letter to have a better and easy understanding of different derived process variants from the customizable invoice payment process as we demonstrate in next section. For example, activity *Receive e-invoice* is renamed by preceding letter 'G' as *G: Receive e-invoice*, where G can be bound to one of the activities *G1*, *G3*, *G4*, *G5* of respective process variant it belongs. And the number assigned to the letter indicates which variant this activity belongs.

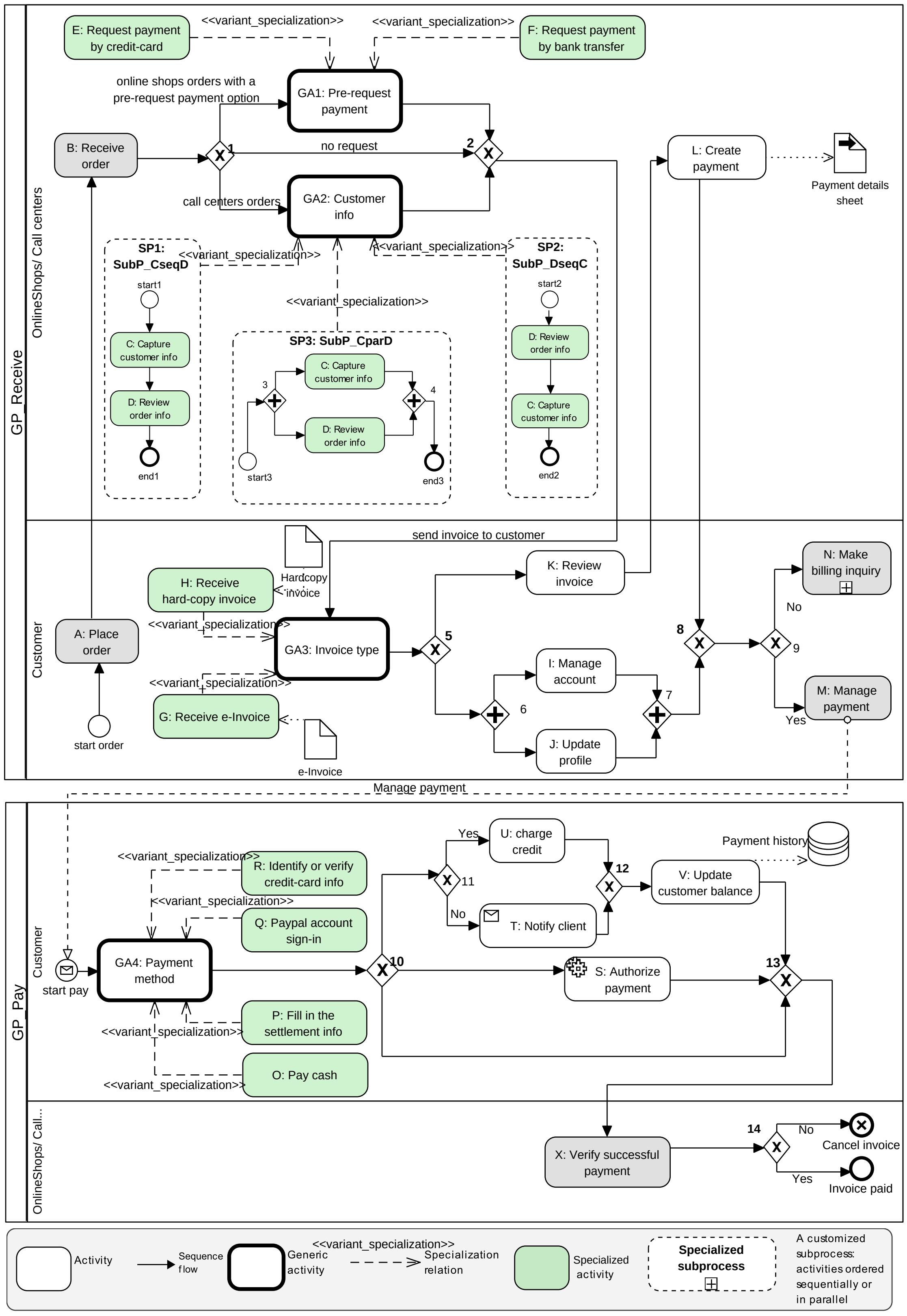


Figure 3.17: Reference invoice payment process model with generic activities

Whereas in table 3.3 we give a representation of tuples of how specialization relationships between activities and generic activities for this reference process model (contains two generic processes) is stored in our relational database.

Table 3.3: Specialization relationships between activities and generic activities of ReferencePM

ActivityName	GenericActivityName	ProcessName
E: Request payment by credit-card	GA1: Pre-request payment	GP_Receive
F: Request payment by bank transfer	GA1: Pre-request payment	GP_Receive
SP1: SubP_CseqD	GA2: Customer info	GP_Receive
SP3: SubP_CparD	GA2: Customer info	GP_Receive
SP2: SubP_DseqC	GA2: Customer info	GP_Receive
G: Receive e-Invoice	GA3: Invoice type	GP_Receive
H: Receive hard-copy invoice	GA3: Invoice type	GP_Receive
R: Identify or verify credit-card info	GA4: Payment method	GP_Pay
P: Fill in the settlement info	GA4: Payment method	GP_Pay
Q: Paypal account sign-in	GA4: Payment method	GP_Pay
O: Pay cash	GA4: Payment method	GP_Pay

If we bound each specialized activity from the generic processes to an activity of a concrete process we get some other tuples as displayed in table 3.4. In this table we show a representation of tuples of how specialization relationships between activities and generic activities for each process variant is stored in our relational database.

Table 3.4: Specialization relationships between activities and generic activities for each process variant

	Activity	GenericActivity
Process Variant 1	E1: Request payment by credit-card	GA1: Pre-request payment
	G1: Receive e-invoice	GA3: Invoice type
	R1: Identify or verify credit-card info	GA4: Payment method
Process Variant 2	F2: Request payment by bank transfer	GA1: Pre-request payment
	H2: Receive hard-copy invoice	GA3: Invoice type
	P2: Fill in the settlement info	GA4: Payment method
Process Variant 3	E3: Request payment by credit-card	GA1: Pre-request payment
	F3: Request payment by bank transfer	GA1: Pre-request payment
	G3: Receive e-invoice	GA3: Invoice type
	H3: Receive hard-copy invoice	GA3: Invoice type
	R3: Identify or verify credit-card info	GA4: Payment method
	P3: Fill in the settlement info	GA4: Payment method
Process Variant 4	G4: Receive e-invoice	GA3: Invoice type
	H4: Receive hard-copy invoice	GA3: Invoice type
	R4: Identify or verify credit-card info	GA4: Payment method
	Q4: PayPal account sign-in	GA4: Payment method
	P4: Fill in the settlement info	GA4: Payment method
Process Variant 5	C: Capture customer info	GA2: Customer info
	D: Review order info	GA2: Customer info
	G5: Receive e-invoice	GA3: Invoice type
	H5: Receive hard-copy invoice	GA3: Invoice type
	R5: Identify or verify credit-card info	GA4: Payment method
	O5: Pay cash	GA4: Payment method

As listed in the above table for process variant 5 that has three customized subprocess as specialized activities of GA2 after applying direct substitution operation this will result in only two records that are elementary activities *C: Capture customer info*, *D: Review order info*. Of course the order on which these two activities are executed (either first C->D, or D->C) can be defined after extracting the info from the respective log files. In cases where the specialized activity is a subprocess e. g., subprocess $SP2 : SubP\_D5$ seq $C5$ then formally we can give the following definition:

Definition 3.3.3. *A (sub)process P is a specialization of a generic activity g* iff $\exists\,\Theta$, *such that* $P(\Theta) = g$.
Equally, a generic activity g is a generalization of a (sub)process P iff $\exists\,\Theta$, *such that* $g(\Theta) = P$

In the following section we graphically present some examples of substitutions.

3.3.2 Specialization and generalization of processes

The notion of process specialization is not new. (Wyner and Lee, 2003) proposed process specialization specifically to state diagrams and data flows to allow it to be incorporated into existing process representations by means of a set of specializing transformations. After applying these transformations to a specific process representation, they result in a specialization of the original process thus taking full advantage of the generative power of specialization hierarchy. They defined all these transformations theoretically, proved them formally and presented an example of a restaurant information system where new state diagrams were developed based on interviews. But they lack on implementing these transformation operations in a practical tool.

After identifying generic activities and their specialization relationships to elementary activities or subprocesses for each process variant we can derive a specialization/generalization hierarchy of processes. Consequently, we can give the following definition:

Definition 3.3.4. *A process P is a specialization of a generic process G iff $\exists\,\Theta$, such that $G(\Theta) = P$.*
Equally, a generic process G is a generalization of a process P $iff \exists\,\Theta$, such that $P(\Theta) = G$

In the following Figure 3.18 we show how the customized process models are derived after applying a substitution of a single GA. Of course if these substitutions are performed to all other GAs we can get all process variants derived. For example, in section a) of this figure *GA1* is substituted with *E*, one of its specializations. Then of course this *E* will be bounded to respective activities of concrete processes. In section b) *GA2* is substituted with subprocess *SP1: SubP_ CseqD*. And in the last section c) *GA3* is substituted with *H*. Of course, if we apply all these substitutions different process behaviours will be derived from the generic process. Accordingly, based on specializations of these generic activities respective process specializations will be derived as shown in the following Table 3.5.

Table 3.5: Process specializations

ProcessId	ProcessName	PVariant	GenericPId	GenericPName
process-BP4745	Receive Invoice	Variant 1	process-BP5148	GP_Receive
process-BP4746	Pay Invoice	Variant 1	process-BP5149	GP_Pay
process-BP4747	Receive Invoice	Variant 2	process-BP5148	GP_Receive
process-BP4748	Pay Invoice	Variant 2	process-BP5149	GP_Pay
process-BP4751	Receive Invoice	Variant 3	process-BP5148	GP_Receive
process-BP4752	Pay Invoice	Variant 3	process-BP5149	GP_Pay
process-BP4753	Receive Invoice	Variant 4	process-BP5148	GP_Receive
process-BP4754	Pay Invoice	Variant 4	process-BP5149	GP_Pay
process-BP4755	Receive Invoice	Variant 5	process-BP5148	GP_Receive
process-BP4756	Pay Invoice	Variant 5	process-BP5149	GP_Pay

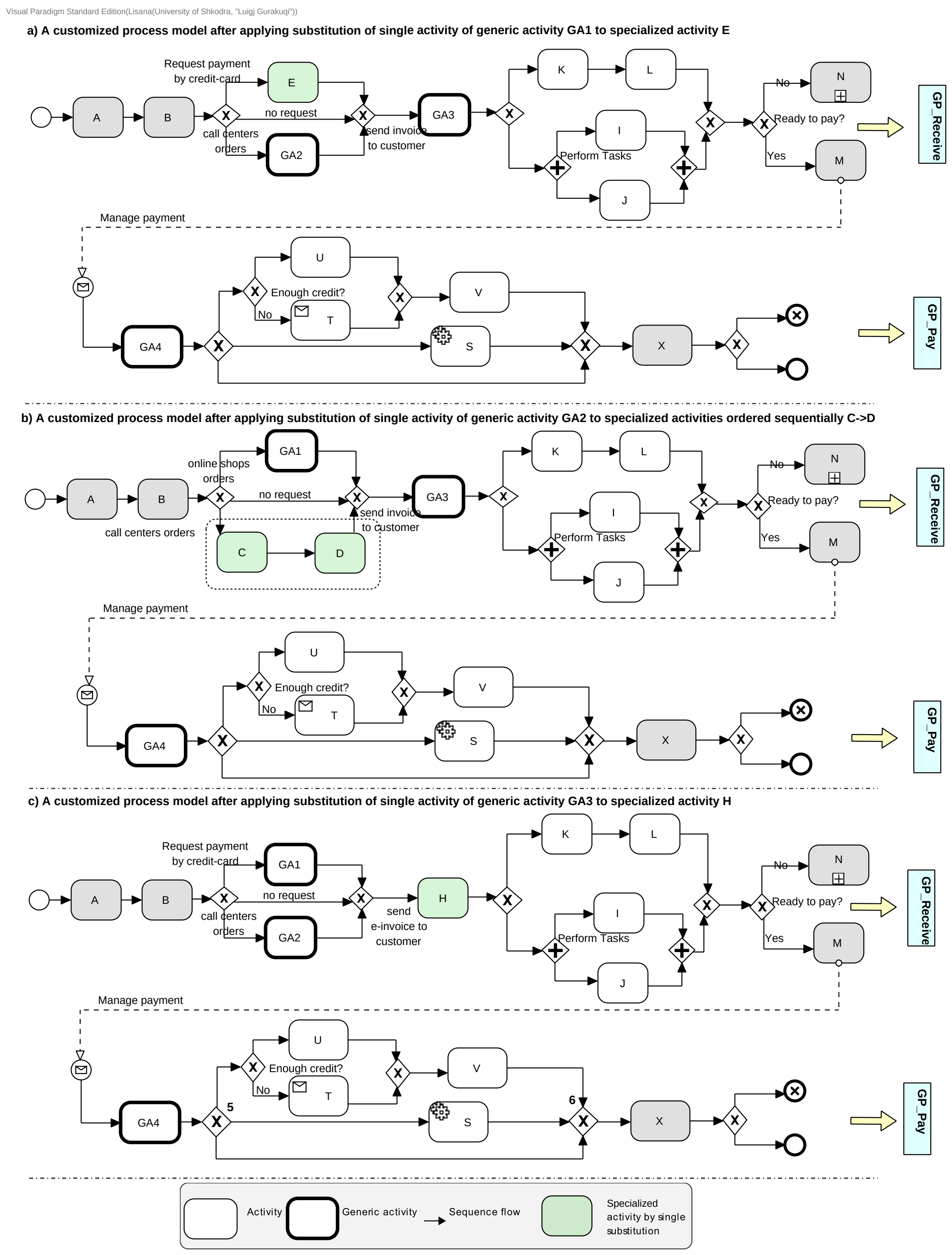

Visual Paradigm Standard Edition(Lisana(University of Shkodra, "Luigj Gurakuqi"))
a) A customized process model after applying substitution of single activity of generic activity GA1 to specialized activity E
Request payment by credit-card
call centers orders
no request
send invoice to customer
Perform Tasks
Ready to pay?
No
Yes
GP_Receive
Manage payment
Enough credit?
GP_Pay
b) A customized process model after applying substitution of single activity of generic activity GA2 to specialized activities ordered sequentially C->D
online shops orders
call centers orders
c) A customized process model after applying substitution of single activity of generic activity GA3 to specialized activity H
send e-invoice to customer
Activity
Generic activity
Sequence flow
Specialized activity by single substitution

Figure 3.18: Specialization of generic processes by substituting single activity

For example, process *ReceiveInvoice* of Variant 2 (see Figure 3.19) will be derived after applying specialization of the generic activity *GA2* to the subprocess named *SP1:*

SubP_CseqD. Then each of activities parts of this subprocess will be bound to activities of concrete processes i. e., *process variant 2*. We show how to derive different customized processes (i. e.,different possible behaviours of a process) after each substitution of GAs to the input generic process *GP_Receive*.

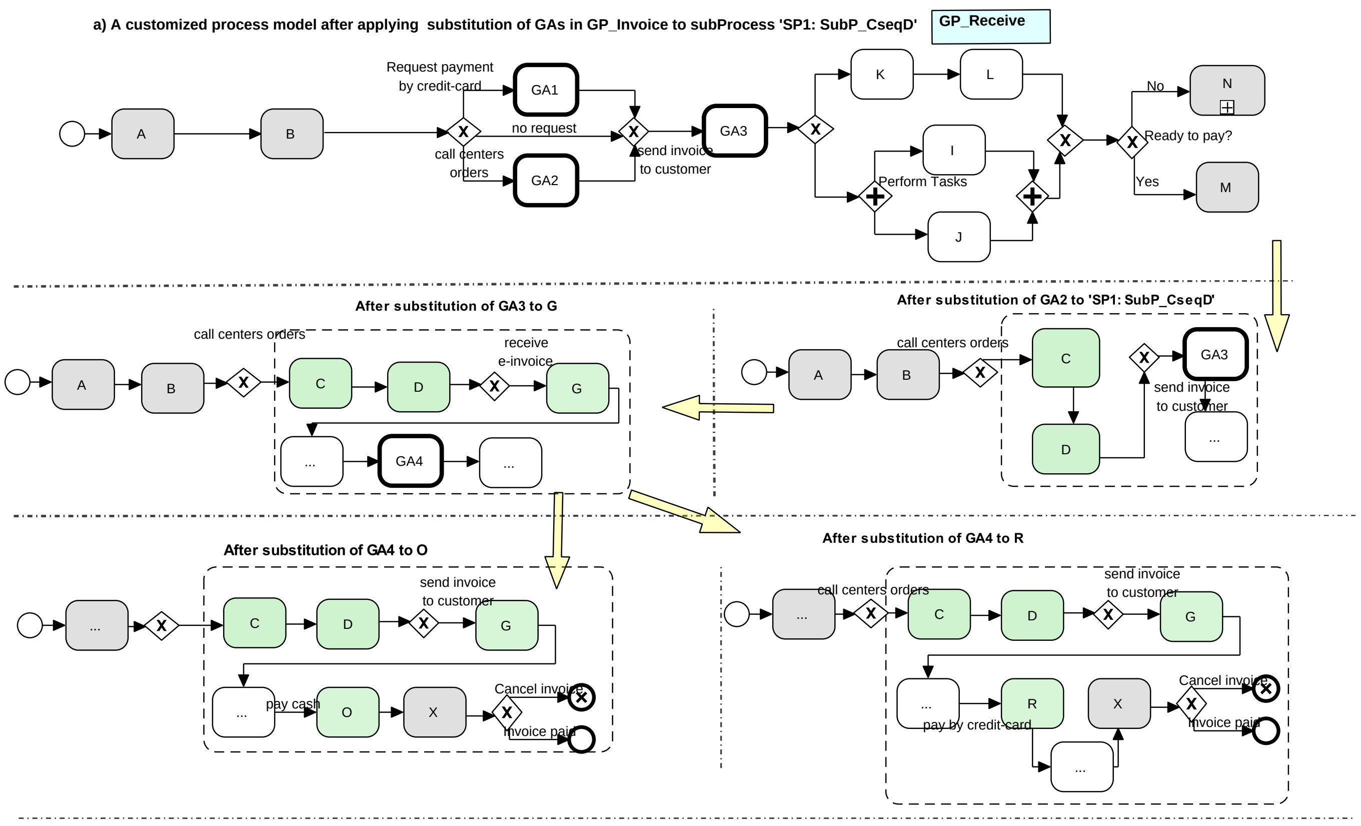

a) A customized process model after applying substitution of GAs in GP_Invoice to subProcess 'SP1: SubP_CseqD'
GP_Receive
Request payment by credit-card
no request
call centers orders
send invoice to customer
Perform Tasks
Ready to pay?
No
Yes
After substitution of GA3 to G
receive e-invoice
After substitution of GA2 to 'SP1: SubP_CseqD'
After substitution of GA4 to O
pay cash
Cancel invoice
invoice paid
After substitution of GA4 to R
pay by credit-card

Figure 3.19: Customized invoice payment process after specialization of GA to a SubProcess

3.3.3 Substitution operations and transformation sets

We discuss how we define specialization between activities and processes in the above sections. In this section, we show how to apply these transformation sets and afterwards represent them in a consolidation hierarchy. We develop an algorithm to generate all activity steps of concrete processes derived from generic processes of the Reference process model after applying substitution of each generic activities with respective activity specializations. The steps of this algorithm are as follows:

(i) Firstly, after getting the sequential order of steps from concrete and generic processes, we filter only some specific occurrences as described in Algorithm 1.

(ii) Secondly, we generate all steps of concrete processes derived from generic processes after applying direct and non-direct specializations of GAs as described in Algorithm 2. For each direct specialization of generic activities we obtain respective activities from concrete processes as bounded activities as demonstrated in Algorithm 3. Whereas, for non-direct specializations we use a breadth-first search strategy to explore other activities starting from the specialized activity up to the last activity of a concrete process. Accordingly, we derive a process variants hierarchy consolidating step activities of concrete process in different GAs positions (their absolute level in a process path of a generic process) corresponding to specific levels of the hierarchy. Therefore, we derive specializations between processes, too, e. g., *Receive Invoice-Variant 1* is a specialization of *GP_ Receive.*

(iii) Thirdly, after configuring the genericity levels of the hierarchy by ranking rows according to *lvl*-(GAs absolute level) values, we finally update step activities consolidated to respective genericity levels of the hierarchy as presented in Algorithm 4.

In the following is stated the algorithm of procedure FILTER_STEPS() which after it generates all steps of all processes in ascending order, i. e., from the very beginning of a specific process to the very end, filters only one occurrence for each step of each process ordered at a specific level, e. g., a max value. The algorithm basically starts with getting first steps of all concrete and generic processes and assigned them as level 1 (see procedure GETFIRSTSTEPS_ALL()[4] at line *15* of Algorithm 1). Then, finds all other next respective steps (both activity and control element steps) and increase the level of their detected order by one. Iteratively, we get all pairs of the result set by matching the previous steps (a self-join) [5] .
Afterwards, we filter by maximum level order (see line *10*, use ρ [6]) from the generated result set to get only those steps grouped based on these attributes (stepId, activityId, processId, isGeneric) part of the *Step* set. The reason for that is that, we can get the same step with the same id in this result set (i. e., the same step is performed in different process paths) ordered in different level. Furthermore, the number of these multiple occurrences of step activities can grow exponentially in 2^n terms, where $n-$ refers to the number of XOR-splitjoin gateways. Consequently, a combinatorial explosion might occur due to this growth of problem complexity.
Finally, we sort the result set values in ascending order.

[4]Symbol $\setminus$ is used to define *MINUS* operation over a relation

[5]We use the symbol $\times$ in relational algebra to define *CROSS JOIN* operations (i. e., to combine relations with results from functions), whereas to identify *Natural Join* operation we use $\bowtie$

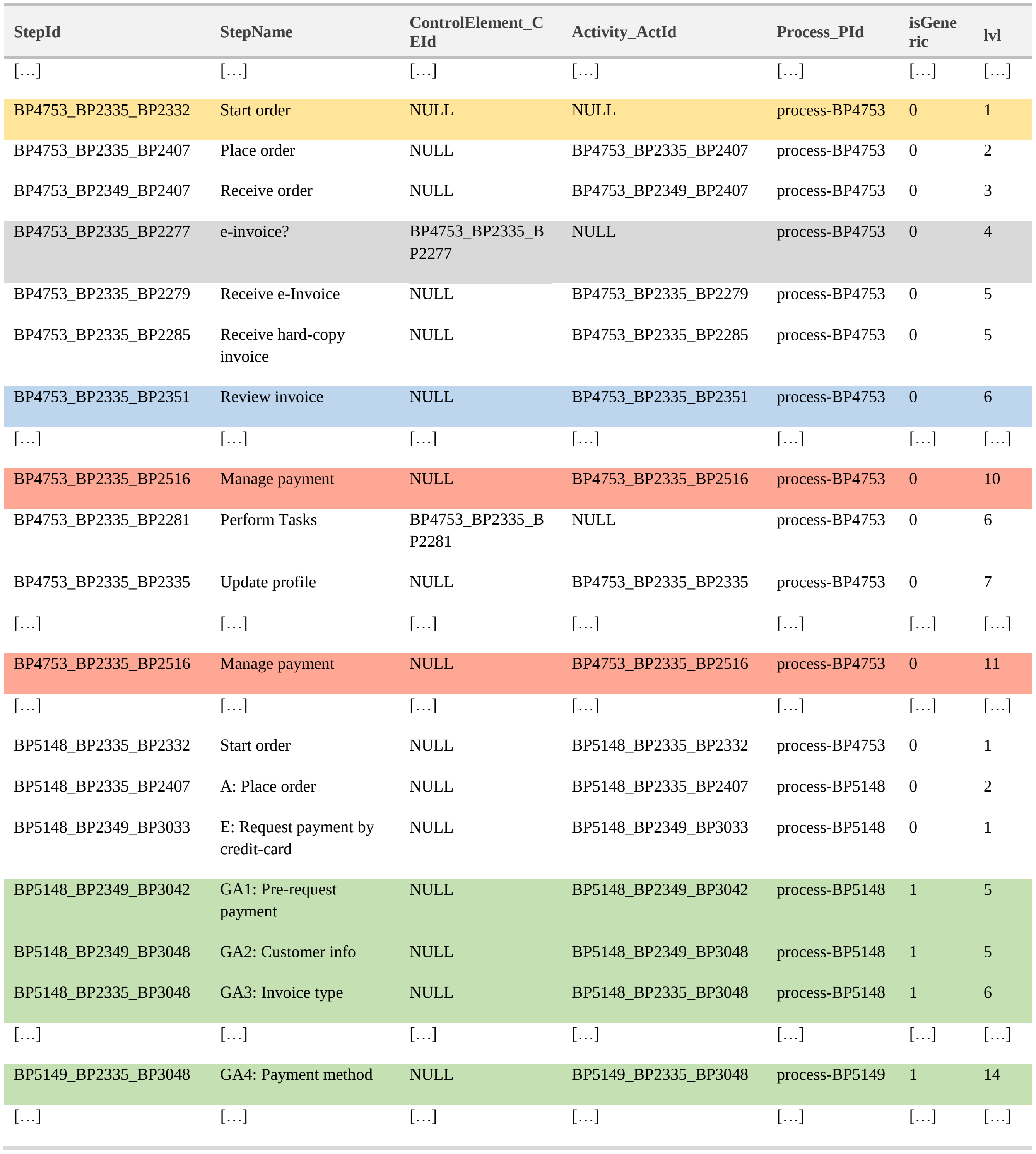
[6]ρ is used to rename attributes(e. g., rename $MAX(lvl)$ with lvl) or relations

Algorithm 1 Filter steps of multiple occurrences from all processes after sequential ordering/sorting

Input: Sets *Step*(all step elements of all processes-either generic or concrete) and *Transition*(from-to step transition elements)
Output: A set *resultSet* with tuples $\{(StepId, ActivityId, ProcessId, isGeneric, lvl)\}$

```
1: procedure FILTER_STEPS()

   Variables
2:     $lvl \leftarrow 1$                                         ▷ initialize step level order

3:     $temp \leftarrow$ GETFIRSTSTEPS_ALL()$, lvl$   ▷ alter set by assigning level 1 for each first step
4:     $count \leftarrow \pi_{count(StepId)}(temp)$                ▷ nr of first steps listed

5:     while $count > 0$ do
6:         $lvl \leftarrow lvl + 1$        ▷ generate all steps occurrences in order for each process
7:         $temp \leftarrow temp \cup \pi_{s.*,lvl}\ \sigma_{tmp.StepId=t.SourceRefStepId \wedge tmp.lvl=lvl-1}$
               $(\sigma_{stepId=TargetRefStepId}(\rho_s(Step) \bowtie \rho_t(Transition)) \bowtie \rho_{tmp}(temp))$
8:         $count \leftarrow count - 1$
9:     end while

   ▷ aggregate each step by the listed attributes to get their max level order
10:    $resultSet \leftarrow {}_{StepId, ActivityId, ProcessId, isGeneric}\mathcal{G}_{\rho_{lvl/MAX(lvl)}}(temp)$
11:    sort in asc. order $ProcessId$ and $lvl$ from $resultSet$
12:    return $resultSet$
13: end procedure

14: procedure GETFIRSTSTEPS_ALL()

   ▷ find all steps that has an incoming step in a transition
15:    $targetSteps \leftarrow \pi_{S.*}\ \sigma_{S.StepId=T.TargetRefStepId}(\rho_S(Step) \bowtie \rho_T(Transition))$

   ▷ get all steps except $targetSteps$
16:    $first\_S \leftarrow Step \setminus targetSteps$
17:    return $first\_S$

18: end procedure
```

An excerpt of result from the set *temp* used in this algorithm is shown in Figure 3.20. Here we show only steps of the concrete process *Receive Invoice* of Variant 4 and some steps of the two generic processes.



StepId	StepName	ControlElement_CEId	Activity_ActId	Process_PId	isGeneric	lvl
[…]	[…]	[…]	[…]	[…]	[…]	[…]
BP4753_BP2335_BP2332	Start order	NULL	NULL	process-BP4753	0	1
BP4753_BP2335_BP2407	Place order	NULL	BP4753_BP2335_BP2407	process-BP4753	0	2
BP4753_BP2349_BP2407	Receive order	NULL	BP4753_BP2349_BP2407	process-BP4753	0	3
BP4753_BP2335_BP2277	e-invoice?	BP4753_BP2335_BP2277	NULL	process-BP4753	0	4
BP4753_BP2335_BP2279	Receive e-Invoice	NULL	BP4753_BP2335_BP2279	process-BP4753	0	5
BP4753_BP2335_BP2285	Receive hard-copy invoice	NULL	BP4753_BP2335_BP2285	process-BP4753	0	5
BP4753_BP2335_BP2351	Review invoice	NULL	BP4753_BP2335_BP2351	process-BP4753	0	6
[…]	[…]	[…]	[…]	[…]	[…]	[…]
BP4753_BP2335_BP2516	Manage payment	NULL	BP4753_BP2335_BP2516	process-BP4753	0	10
BP4753_BP2335_BP2281	Perform Tasks	BP4753_BP2335_BP2281	NULL	process-BP4753	0	6
BP4753_BP2335_BP2335	Update profile	NULL	BP4753_BP2335_BP2335	process-BP4753	0	7
[…]	[…]	[…]	[…]	[…]	[…]	[…]
BP4753_BP2335_BP2516	Manage payment	NULL	BP4753_BP2335_BP2516	process-BP4753	0	11
[…]	[…]	[…]	[…]	[…]	[…]	[…]
BP5148_BP2335_BP2332	Start order	NULL	BP5148_BP2335_BP2332	process-BP4753	0	1
BP5148_BP2335_BP2407	A: Place order	NULL	BP5148_BP2335_BP2407	process-BP5148	0	2
BP5148_BP2349_BP3033	E: Request payment by credit-card	NULL	BP5148_BP2349_BP3033	process-BP5148	0	1
BP5148_BP2349_BP3042	GA1: Pre-request payment	NULL	BP5148_BP2349_BP3042	process-BP5148	1	5
BP5148_BP2349_BP3048	GA2: Customer info	NULL	BP5148_BP2349_BP3048	process-BP5148	1	5
BP5148_BP2335_BP3048	GA3: Invoice type	NULL	BP5148_BP2335_BP3048	process-BP5148	1	6
[…]	[…]	[…]	[…]	[…]	[…]	[…]
BP5149_BP2335_BP3048	GA4: Payment method	NULL	BP5149_BP2335_BP3048	process-BP5149	1	14
[…]	[…]	[…]	[…]	[…]	[…]	[…]

Figure 3.20: An excerpt of the output of set *temp* in Algorithm 1

To distinguish between different steps shown in the above figure we marked them with some colors as follow:

(i) step colored in light yellow indicates an event (e. g., Start order) with *NULL* values for both columns *ControlElement_ CEId* and *Activity_ ActId* of *Step* table.

(ii) step colored in light blue indicates an activity (e. g., Review invoice) with the same value for column *Activity_ ActId* and a *NULL* value for column *ControlElement_-CEId*.

(iii) step colored in light gray indicates a control element (e. g., e-invoice?) with the same value for column *ControlElement_ CEId* and a *NULL* value for column *Activity_-ActId*.

(iv) step colored in light red indicates activities with multiple occurrences (i. e., with different level's order), e. g., *Manage payment* ordered at level 10 and 11.

(v) step colored in light green indicates generic steps with value **1** for column *isGeneric* to distinguish all generic activities that can be specialized to either activities or subprocesses.

For example, step activity *Verify successful payment* of the process *Pay Invoice* of *Variant 4* is processed multiple times as a result of different process paths as shown in Figure 3.21. We display with different colors all possible paths of process enactment, i. e., with color *orange* the process path if customer chose to pay by bank transfer. As a result, activity *Verify successful payment* is performed at step level 15. Whereas, the path with *gray* color if customer chose paypal method payment causes the execution of activity *Verify successful payment* at step level 16. And last, we apply *cyan* color for the path where a customer chose to pay by credit-card. This results in processing activity *Verify successful payment* in step level 19. Of course, we supposed that start-event message of process *Pay Invoice* is performed at level "11" (if customer selects hardcopy invoice) because it can have a different value, i. e., "12" if customer selected an e-invoice order. This depends on the process path of the previous process *Receive Invoice*.

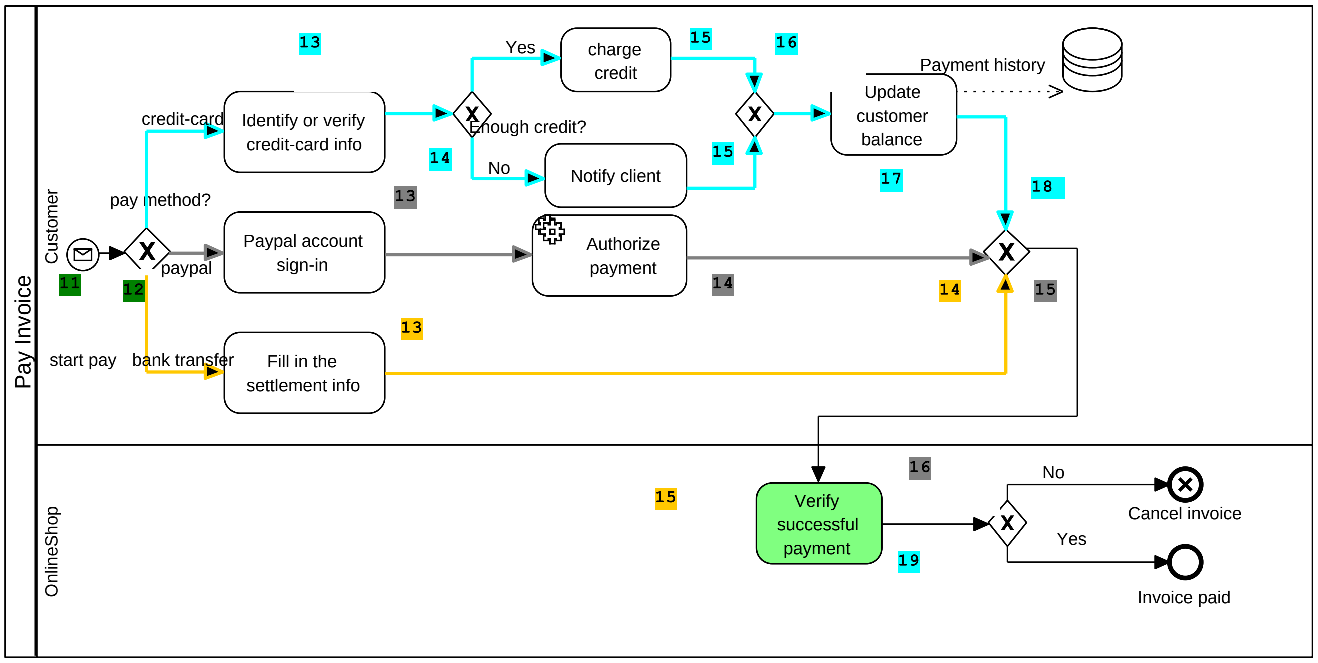


Figure 3.21: Process Pay Invoice of Variant 4 with colored process paths

Table 3.6: An excerpt of the same activity ordered at multiple levels

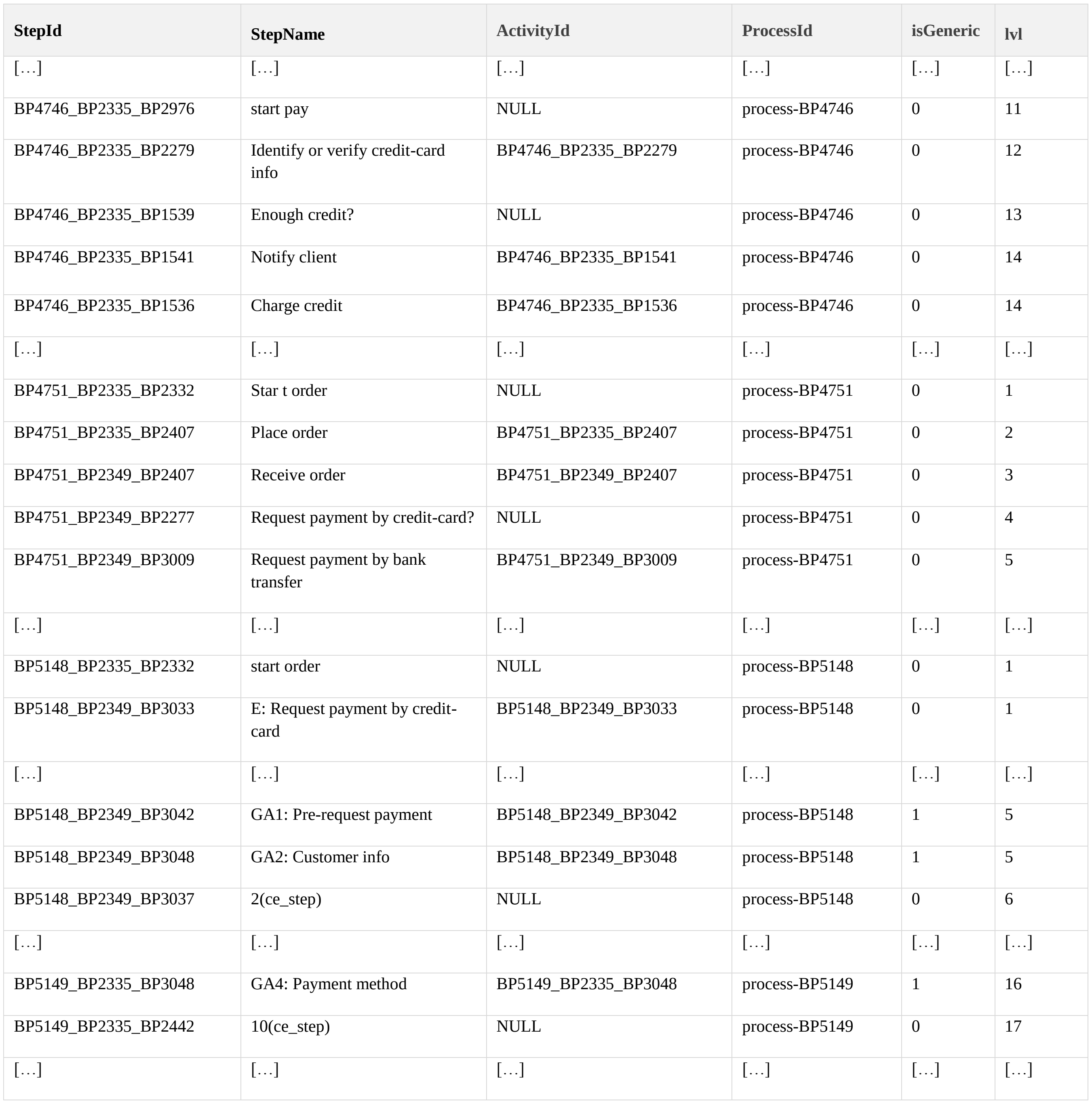
StepId	StepName	ProcessId	isGeneric	lvl
BP4754_BP2335_BP3037	start pay	process-BP4754	0	11
[..]	[..]	[..]	[..]	[..]
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	15
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	16
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	16
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	16
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	17
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	17
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	19
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	19
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	20
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	20
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	20
BP4754_BP2883_BP2448	Verify successful payment	process-BP4754	0	20
[..]	[..]	[..]	[..]	[..]

To overcome the problem of multiple occurrences of the same step ordered at different levels, we filter only those steps with a specific value for their level order, e. g., a max value. This value refers to the absolute position for each activity in a specific process path. Meanwhile, we are interested in the "relative" position of each activity with reference to a GA's position, i. e., a variation point level. For more details please refer to Table 3.6. Additionally, we give an example of the activities performed in sequence via different process paths(instances) (see Table 3.7) and store the results as a view. This is to show case that different results might be produced by writing *SQL* scripts. As shown in the table the same sequence of activities may result from different process instances.

Table 3.7: Sequence of activities in a process path of Variant 4

ProcessId	PrInsId	Activities_in_Sequence
[..]	[..]	[..]
process-BP4754	1	Paypal account sign-in\Authorize payment\Verify successful payment\Cancel invoice
process-BP4754	2	Identify or verify credit-card info\charge credit\Update customer balance\Verify successful payment\Invoice paid
process-BP4754	3	Identify or verify credit-card info\charge credit\Update customer balance\Verify successful payment\Cancel invoice
process-BP4754	4	Identify or verify credit-card info\charge credit\Update customer balance\Verify successful payment\Invoice paid
process-BP4754	5	Identify or verify credit-card info\Notify client\Update customer balance\Verify successful payment\Invoice paid
process-BP4754	6	Identify or verify credit-card info\charge credit\Update customer balance\Verify successful payment\Cancel invoice
process-BP4754	7	Fill in the settlement info\Verify successful payment\Invoice paid
process-BP4754	8	Fill in the settlement info\Verify successful payment\Cancel invoice
process-BP4754	9	Identify or verify credit-card info\charge credit\Update customer balance\Verify successful payment\Invoice paid
process-BP4754	10	Paypal account sign-in\Authorize payment\Verify successful payment\Invoice paid
[..]	[..]	[..]

We show the output of algorithm 1 in the Figure 3.22 . As it is displayed, column *ActivityId* has the same value of *StepId* if this step is an activity, otherwise it might be an *event* or a *control element* as we already highlighted in Figure 3.20.



StepId	StepName	ActivityId	ProcessId	isGeneric	lvl
[…]	[…]	[…]	[…]	[…]	[…]
BP4746_BP2335_BP2976	start pay	NULL	process-BP4746	0	11
BP4746_BP2335_BP2279	Identify or verify credit-card info	BP4746_BP2335_BP2279	process-BP4746	0	12
BP4746_BP2335_BP1539	Enough credit?	NULL	process-BP4746	0	13
BP4746_BP2335_BP1541	Notify client	BP4746_BP2335_BP1541	process-BP4746	0	14
BP4746_BP2335_BP1536	Charge credit	BP4746_BP2335_BP1536	process-BP4746	0	14
[…]	[…]	[…]	[…]	[…]	[…]
BP4751_BP2335_BP2332	Star t order	NULL	process-BP4751	0	1
BP4751_BP2335_BP2407	Place order	BP4751_BP2335_BP2407	process-BP4751	0	2
BP4751_BP2349_BP2407	Receive order	BP4751_BP2349_BP2407	process-BP4751	0	3
BP4751_BP2349_BP2277	Request payment by credit-card?	NULL	process-BP4751	0	4
BP4751_BP2349_BP3009	Request payment by bank transfer	BP4751_BP2349_BP3009	process-BP4751	0	5
[…]	[…]	[…]	[…]	[…]	[…]
BP5148_BP2335_BP2332	start order	NULL	process-BP5148	0	1
BP5148_BP2349_BP3033	E: Request payment by credit-card	BP5148_BP2349_BP3033	process-BP5148	0	1
[…]	[…]	[…]	[…]	[…]	[…]
BP5148_BP2349_BP3042	GA1: Pre-request payment	BP5148_BP2349_BP3042	process-BP5148	1	5
BP5148_BP2349_BP3048	GA2: Customer info	BP5148_BP2349_BP3048	process-BP5148	1	5
BP5148_BP2349_BP3037	2(ce_step)	NULL	process-BP5148	0	6
[…]	[…]	[…]	[…]	[…]	[…]
BP5149_BP2335_BP3048	GA4: Payment method	BP5149_BP2335_BP3048	process-BP5149	1	16
BP5149_BP2335_BP2442	10(ce_step)	NULL	process-BP5149	0	17
[…]	[…]	[…]	[…]	[…]	[…]

Figure 3.22: An excerpt of the tuples output in Algorithm 1

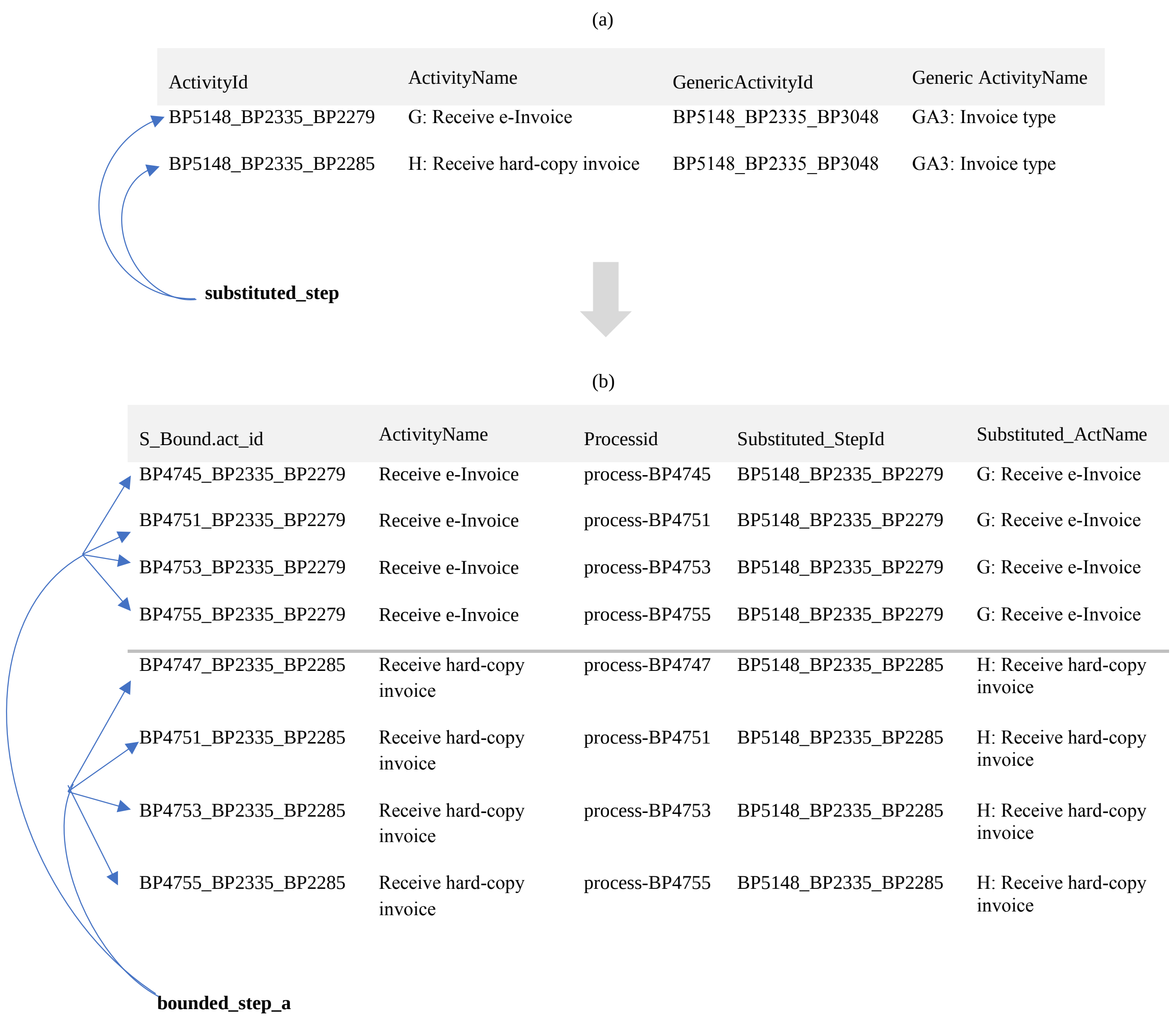
Whereas algorithm 2 as described below generates all steps of concrete processes derived from generic processes after applying direct and non-direct specializations of GAs. As a result, from all these process specializations after substitution operations we derive process variant hierarchy. Firstly, after skipping all non activity steps (i. e., control elements, line *4* of this algorithm) we fetch[7] each row iteratively (line *3* of this algorithm) (from the result set of Algorithm 1) till the last one.
While fetching we distinguish between two kind of steps, i. e., *generic* (line *5* and *non-generic steps* (line *14*). For the generic steps found we pair(combine using cross join operations annotated by symbol $\times$) each of direct (line *6, 7*) and non-direct specialized activities (after performing bound operation on matching activity labels) that substitutes a generic activity to this generic step id (i. e., refers to a GA). To get all indirect specializations (line *9*) after performing a substitution of a GA we use a breadth-first-search strategy starting from the specialized activity up to end of last activity of a process variant. In so doing, we traverse and explore all of the neighbor steps at the present depth prior to moving on to the steps at the next depth level. There are two types of specialized activities: a *custom sub-process* or an *elementary activity.* Basic bounded operations are described in Algorithm 3. We do this operation (replacement) for all remaining generic activities defined and iteratively we get all other direct and indirect specializations of respective GAs. To substitute a GA with its direct specialization it's trivial because we know which activity is a specialization of a which GA (see Table 3.4). Whereas, for the substitution of a GA with a subprocess we firstly list step activities of the specialized subprocess and then for each of them we get the bounded activity based on the longest common string(matching label activities). We give details on these bounds operations in the output of the Algorithm 3.

[7]We use *DECLARE CURSOR FOR SELECT* in *TSQL* to fetch each row separately

Algorithm 2 Derive process variants hierarchy after applying direct and non-direct specializations of GAs from generic processes

```
Input: $all\_Filtered\_Steps \leftarrow$ FILTER_STEPS()
Output: A multiset $PV\_Hierarchy$ with tuples $\{(act\_id, ga\_id, process\_id, lvl)\}$
    Variables
    $step\_ID = \pi_{StepId}\ (all\_Filtered\_Steps)$
    $process\_ID = \pi_{ProcessId}\ (all\_Filtered\_Steps)$
    $activity\_ID = \pi_{ActivityId}\ (all\_Filtered\_Steps)$
    $lvl = \pi_{lvl}\ (all\_Filtered\_Steps)$          ▷ step level
    $isGeneric = \pi_{isGeneric}\ (all\_Filtered\_Steps)$   ▷isGeneric=1 i.e. step is a GA

 1: procedure DERIVE_PV_HIERARCHY()
 2:     $PV\_Hierarchy \leftarrow \emptyset$       ▷ multiset of process variants specializations

 3:     foreach $step\_ID \in all\_Filtered\_Steps$ do
 4:         if $activity\_ID$ is not null then              ▷ skip control element steps
 5:             if $isGeneric = true$ then   ▷ check if $step\_ID$ is a GA
              ▷ get direct GA's specialization i. e., an EA or SP
 6:                 $substituted\_step \leftarrow \pi_{ActivityId}\ \sigma_{GenericActivityId=step\_ID}(a\_is\_spec\_of\_ga)$
 7:                 $bounded\_step\_a \leftarrow \pi_{S\_Bound.act\_id,\mathbf{step\_ID},\mathbf{process\_ID},lvl}(\rho_{Sub\_S}(substituted\_step) \times$
                              $\rho_{S\_Bound}($GETBOUNDEDSTEP_OF_A$(Sub\_S.ActivityId)))$
      ▷ insert into multiset $PV\_Hierarchy$ with current tuples
 8:                 $PV\_Hierarchy \leftarrow PV\_Hierarchy \cup bounded\_step\_a$
      ▷ get indirect GA's specialization using breadth-first-search strategy
 9:                 $indirect\_step\_a \leftarrow \pi_{BFS.act\_id,\mathbf{step\_ID},\mathbf{process\_ID},lvl}(\rho_{S\_Bound}(bounded\_step\_A) \times$
                              $\rho_{BFS}($BREADTH_FIRST_SEARCH$(S\_Bound.act\_id)))$
      ▷ insert into multiset $PV\_Hierarchy$ with current tuples
10:                 $PV\_Hierarchy \leftarrow PV\_Hierarchy \cup indirect\_step\_a$
        ▷ derive and store process specializations
11:                 get $concretePId\_of\_a$ for $bounded\_step\_a$
12:                 get $concretePId\_of\_ind\_a$ for $indirect\_step\_a$
13:                 $p\_is\_spec\_of\_gp \leftarrow p\_is\_spec\_of\_gp \cup$
                              $\{(concretePId\_of\_a, process\_ID)\} \cup$
                              $\{(concretePId\_of\_ind\_a, process\_ID)\}$
14:             end if
15:         end if
16:         get next $step\_ID$ from $all\_Filtered\_Steps$
17:     end foreach
        ▷ rename $PV\_Hierarchy$ attributes set
18:     $PV\_Hierarchy \leftarrow \pi_{act\_id,\rho_{ga\_id/step\_ID},\rho_{process\_id/process\_ID},lvl}(PV\_Hierarchy)$
19:     return $PV\_Hierarchy$

20: end procedure
```

In the following figure we give a demonstration of how we perform substitutions. Let's start fetching one of the tuples from the set *all_ Filtered_ steps* (line *3* in this algorithm), example step activity: *GA3: Invoice type*, its step_id:*BP5148_ BP2335_ BP3048* part of generic process *GP_ Receive*. As a generic step value 1 is stored to the local attribute named *isGeneric* that corresponds with the same attribute name of step class.



(a)

ActivityId	ActivityName	GenericActivityId	Generic ActivityName
BP5148_BP2335_BP2279	G: Receive e-Invoice	BP5148_BP2335_BP3048	GA3: Invoice type
BP5148_BP2335_BP2285	H: Receive hard-copy invoice	BP5148_BP2335_BP3048	GA3: Invoice type

substituted_step

(b)

S_Bound.act_id	ActivityName	Processid	Substituted_StepId	Substituted_ActName
BP4745_BP2335_BP2279	Receive e-Invoice	process-BP4745	BP5148_BP2335_BP2279	G: Receive e-Invoice
BP4751_BP2335_BP2279	Receive e-Invoice	process-BP4751	BP5148_BP2335_BP2279	G: Receive e-Invoice
BP4753_BP2335_BP2279	Receive e-Invoice	process-BP4753	BP5148_BP2335_BP2279	G: Receive e-Invoice
BP4755_BP2335_BP2279	Receive e-Invoice	process-BP4755	BP5148_BP2335_BP2279	G: Receive e-Invoice
BP4747_BP2335_BP2285	Receive hard-copy invoice	process-BP4747	BP5148_BP2335_BP2285	H: Receive hard-copy invoice
BP4751_BP2335_BP2285	Receive hard-copy invoice	process-BP4751	BP5148_BP2335_BP2285	H: Receive hard-copy invoice
BP4753_BP2335_BP2285	Receive hard-copy invoice	process-BP4753	BP5148_BP2335_BP2285	H: Receive hard-copy invoice
BP4755_BP2335_BP2285	Receive hard-copy invoice	process-BP4755	BP5148_BP2335_BP2285	H: Receive hard-copy invoice

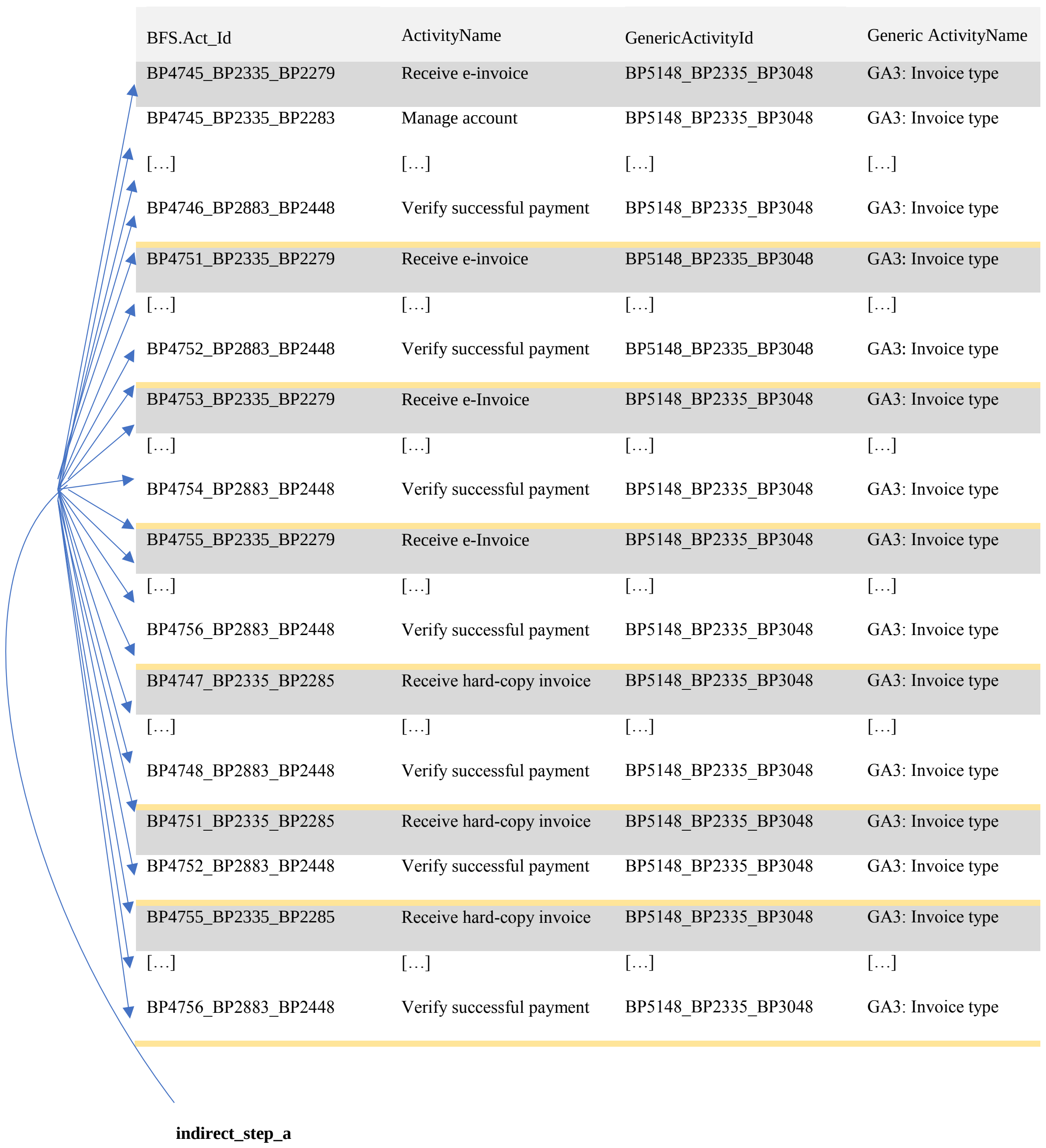
Figure 3.23: Direct specialized activities of the generic activity *GA3: Invoice type* and their respective bounded activities

This generic activity has two specialized activities (elementary ones) stored at temporary set variable named *substituted_ step* (line *6* of the following algorithm). We compute this by applying a project and restrict operation (in Relational Algebra) to relation *(a_-is_ spec_ of_ ga)*. This relation consists of two attributes *ActivityId, GenericActivityId*

(primary keys from the both classes its connects to in a many-many relationship between *Activity* and *GenericActivity*); generated after switching from conceptual data modelling to physical data model. It stores data about specializations of each generic activities defined in the generic processes as already explained in Section 3.3.1.
As shown in the above figure (part *a*), the generic activity *GA3: Invoice type* of process GP_Receive has two activity specializations *G: Receive e-Invoice* and *H: Receive hard-copy invoice*. Then, we find respective bounded activities of each concrete processes (Figure 3.23 part *b*), i. e., *G1: Receive e-invoice*, *Receive e-invoice* of variant 1, *Receive e-invoice* of variant 4 and *Receive e-invoice* of variant 5. We do the same for the other specialized activities. And then we combine each bounded step activities with this generic activity by using *cross join* operations.

We store these values at a temporary set variable named *bounded_step_a* as stated in line *7* of this algorithm.
To get the indirect specializations of a generic activity we start from the bounded step and perform a breadth first search strategy[8] to get iteratively all other sequence step activities part of a concrete process.
These values are stored at another temporary set variable named *indirect_step_a* as written in line *9* of this algorithm. We combine each indirect step of the result with the specific generic activity by using *cross join* operations.
We display some of the results as shown in Figure 3.24.

[8]For more details of the typical algorithm of breadth first search interested reader can read the table-valued user-defined function in *SQL* at Appendix A



BFS.Act_Id	ActivityName	GenericActivityId	Generic ActivityName
BP4745_BP2335_BP2279	Receive e-invoice	BP5148_BP2335_BP3048	GA3: Invoice type
BP4745_BP2335_BP2283	Manage account	BP5148_BP2335_BP3048	GA3: Invoice type
[…]	[…]	[…]	[…]
BP4746_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type
BP4751_BP2335_BP2279	Receive e-invoice	BP5148_BP2335_BP3048	GA3: Invoice type
[…]	[…]	[…]	[…]
BP4752_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type
BP4753_BP2335_BP2279	Receive e-Invoice	BP5148_BP2335_BP3048	GA3: Invoice type
[…]	[…]	[…]	[…]
BP4754_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type
BP4755_BP2335_BP2279	Receive e-Invoice	BP5148_BP2335_BP3048	GA3: Invoice type
[…]	[…]	[…]	[…]
BP4756_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type
BP4747_BP2335_BP2285	Receive hard-copy invoice	BP5148_BP2335_BP3048	GA3: Invoice type
[…]	[…]	[…]	[…]
BP4748_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type
BP4751_BP2335_BP2285	Receive hard-copy invoice	BP5148_BP2335_BP3048	GA3: Invoice type
BP4752_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type
BP4755_BP2335_BP2285	Receive hard-copy invoice	BP5148_BP2335_BP3048	GA3: Invoice type
[…]	[…]	[…]	[…]
BP4756_BP2883_BP2448	Verify successful payment	BP5148_BP2335_BP3048	GA3: Invoice type

Figure 3.24: Non-direct specialized activities of the generic activity *GA3: Invoice type*

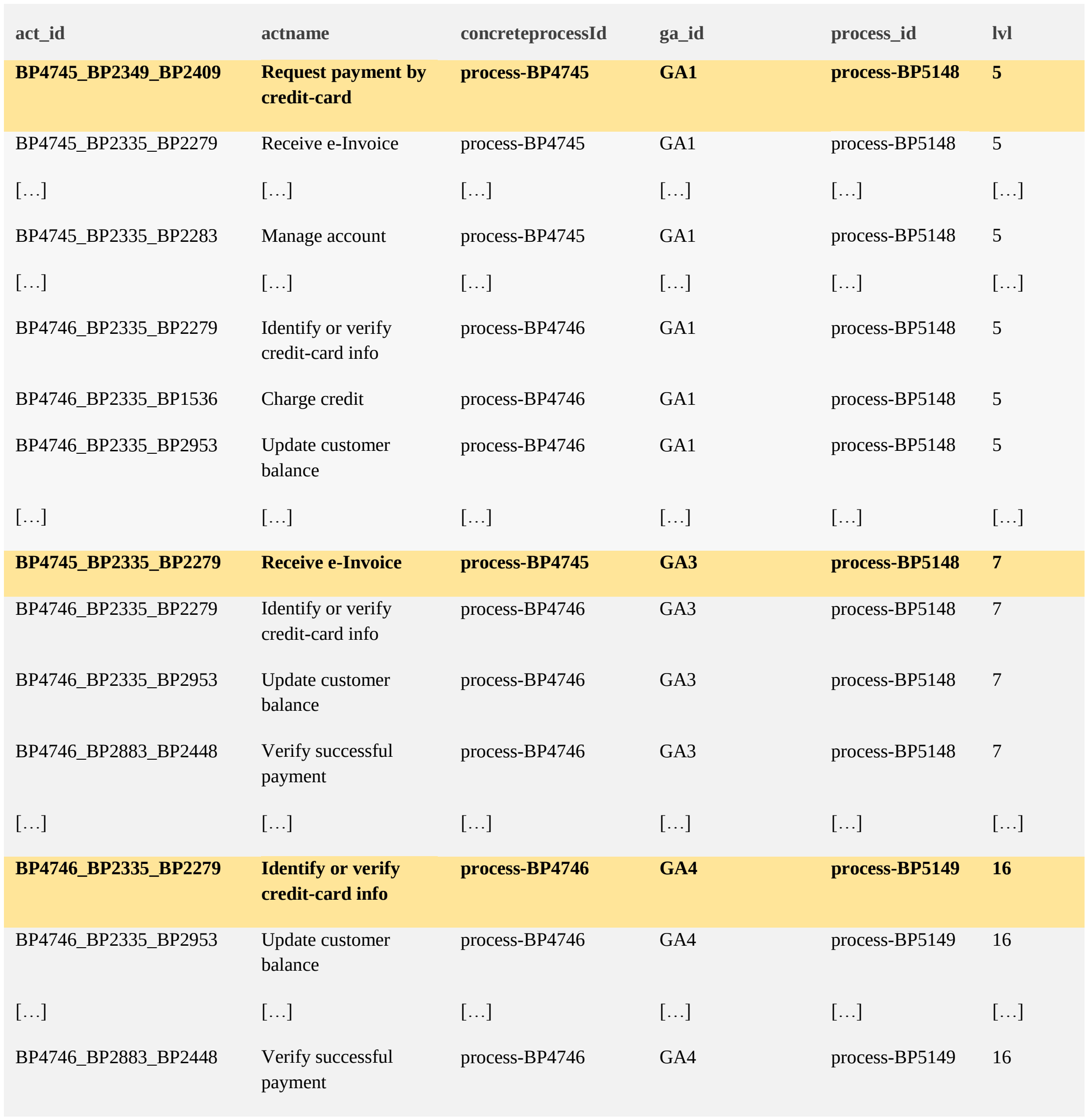
After we get the *process_id* of the *bounded_step_a* we derive the specializations between processes. For example, activity *Receive e-invoice* with id:*BP4745_BP2335_-BP2279* belongs to concrete process with id: *process-BP4745* of variant 1. Consequently, we derive the fact that concrete process *process-BP4745* named *ReceiveInvoice* is a specialization of generic process *process-BP5148* named *GP_Receive*(it contains generic activity

GA3: Invoice type). The same for all other concrete process specializations of generic processes. We store these tuple values to relation named *p_is_spec_of_gp* as a relation generated from a many-many relationship between *Process* and *GenericProcess* entity sets (see line *8* of this algorithm).
We don't need to check for other activities part of a generic process and found their specific bounded activities from concrete processes because they are already explored as non-direct specialized activities after performing BREADTH_FIRST_SEARCH() approach.

Finally, we store each processed results in the multiset called *PV_Hierarchy* (lines *8 and 10*) as tuples i. e., $\{(act\_id, ga\_id, process\_id, lvl)\})$, where:

- *act_id* represent specialized (direct and non-direct) activities of concrete processes
- *ga_id* represent generic activities
- *process_id* represent generic processes
- *lvl* represent the step GA level (absolute level) in a generic process path

An excerpt of the example output of this algorithm for process Variant 1 is captured in Figure 3.25. In this figure we show some of the specialized direct(non-direct)activities of process variant 1 combined(paired) to a generic activity of a generic process. In this figure, we added two columns *stepname* and *concreteProcessId* to make the results easy understandable; moreover, we use short value names for column *act_ID* such as: GA1, GA3, GA4 instead of the IDs of each generic activities.



act_id	actname	concreteprocessId	ga_id	process_id	lvl
BP4745_BP2349_BP2409	**Request payment by credit-card**	**process-BP4745**	**GA1**	**process-BP5148**	**5**
BP4745_BP2335_BP2279	Receive e-Invoice	process-BP4745	GA1	process-BP5148	5
[…]	[…]	[…]	[…]	[…]	[…]
BP4745_BP2335_BP2283	Manage account	process-BP4745	GA1	process-BP5148	5
[…]	[…]	[…]	[…]	[…]	[…]
BP4746_BP2335_BP2279	Identify or verify credit-card info	process-BP4746	GA1	process-BP5148	5
BP4746_BP2335_BP1536	Charge credit	process-BP4746	GA1	process-BP5148	5
BP4746_BP2335_BP2953	Update customer balance	process-BP4746	GA1	process-BP5148	5
[…]	[…]	[…]	[…]	[…]	[…]
BP4745_BP2335_BP2279	**Receive e-Invoice**	**process-BP4745**	**GA3**	**process-BP5148**	**7**
BP4746_BP2335_BP2279	Identify or verify credit-card info	process-BP4746	GA3	process-BP5148	7
BP4746_BP2335_BP2953	Update customer balance	process-BP4746	GA3	process-BP5148	7
BP4746_BP2883_BP2448	Verify successful payment	process-BP4746	GA3	process-BP5148	7
[…]	[…]	[…]	[…]	[…]	[…]
BP4746_BP2335_BP2279	**Identify or verify credit-card info**	**process-BP4746**	**GA4**	**process-BP5149**	**16**
BP4746_BP2335_BP2953	Update customer balance	process-BP4746	GA4	process-BP5149	16
[…]	[…]	[…]	[…]	[…]	[…]
BP4746_BP2883_BP2448	Verify successful payment	process-BP4746	GA4	process-BP5149	16

Figure 3.25: An excerpt of Algorithm 2 for variant 1

So, all these specialized activities will be consolidated to a generic activity. The same for all other process variants. The results of Algorithm 2 will be shifted as a process dimension of the process warehouse schema explained in Section 5.2. Therefore, typical operations such as drill-down and roll-up can be performed on this dimension.

Next, we introduce the algorithm that finds the bounded steps of specialized activities (either elementary activity or subprocess) from a generic activity (part of a generic process).

Algorithm 3 Find bounded steps of (non)specialized activities (both EA, SP) from a GP

```
Input: Sets Step(steps of all processes-either generic or concrete),
    a_is_spec_of_ga(specialized activities of a GA) and GenericActivity(generic
    activities with respective types, i. e. either Abstract or Non-Abstract)

Output: A set steps_Bounded with $(stepId, stepName, process\_Pid)$ attributes

 1: procedure getBoundedStep_of_A($ActId$)

    ▷ get the type of the specialized activity with id Actid
 2:     $a\_type \leftarrow \pi_{ActType}\, \sigma_{GA.ActivityId=ActId\, \wedge\, GA.GenericActivityId=S.GenericActivityId}$
                $(\rho_{GA}(GenericActivity) \bowtie \rho_S(a\_is\_spec\_of\_ga))$

    ▷ get the activity name of the Actid argument in the procedure
 3:     $spec\_actname \leftarrow \pi_{stepName}\, \sigma_{stepId=ActId}(Step)$

    ▷ get only steps from concrete processes
 4:     $steps\_concrete \leftarrow \pi_{stepId,stepName,Activity\_ActId,process\_Pid}\, \sigma_{S.Process_Pid=CP.Process_Pid}$
                $(\rho_S(Step) \bowtie \rho_{CP}(ConcreteProcess))$

 5:     if $a\_type\ is\ 'Non-Abstract'$ then          ▷ specialized activity is an EA
 6:         $step\_ea \leftarrow \pi_{S.stepId,S.stepName,S.Process\_Pid}\, \sigma_{Label.matchLength=\mathbf{length}(S.stepname)}($
                $\sigma_{S.stepId=S.Activity\_ActId}(\rho_S(steps\_concrete) \bowtie (\rho_{S1}(spec\_actname)))$
                $\times (\rho_{Label}$MatchLabelString$(S.stepName, S1.stepName)))$

 7:     else if $a\_type\ is\ 'Abstract'$ then          ▷ specialized activity is a SP

    ▷ get steps of the specialized subprocess
 8:         $SubP\_steps \leftarrow \pi_{S.stepId,S.ActivityId,S.ProcessId}\, \sigma_{S.processId=ActId}(\rho_S(\text{FILTER\_STEPS}()))$

 9:         $step\_sp \leftarrow \pi_{S.stepId,S.stepName,S.Process\_Pid}\, \sigma_{Label.matchLength=\mathbf{length}(S.stepname)}($
                $\sigma_{S.stepId=S.Activity\_ActId}(\rho_S(steps\_concrete) \bowtie (\rho_{S1}(SubP\_steps)))$
                $\times (\rho_{Label}$MatchLabelString$(S.stepName, S1.stepName)))$
10:     end if
11:     $steps\_Bounded \leftarrow step\_ea \cup step\_sp$

12:     return $steps\_Bounded$

13: end procedure
```

There exists two types of specializations of activities from a generic activity (GA):

- elementary activity e. g., *E: Request payment by credit-card*, *F: Request payment by bank transfer*, *R: Identify or verify credit-card info*, and all other activities which has a value *Non-abstract* for attribute named *ActTypeGA* as shown in Figure 3.26. Therefore, in this case GA is defined as an activity of type *Non-abstract*.

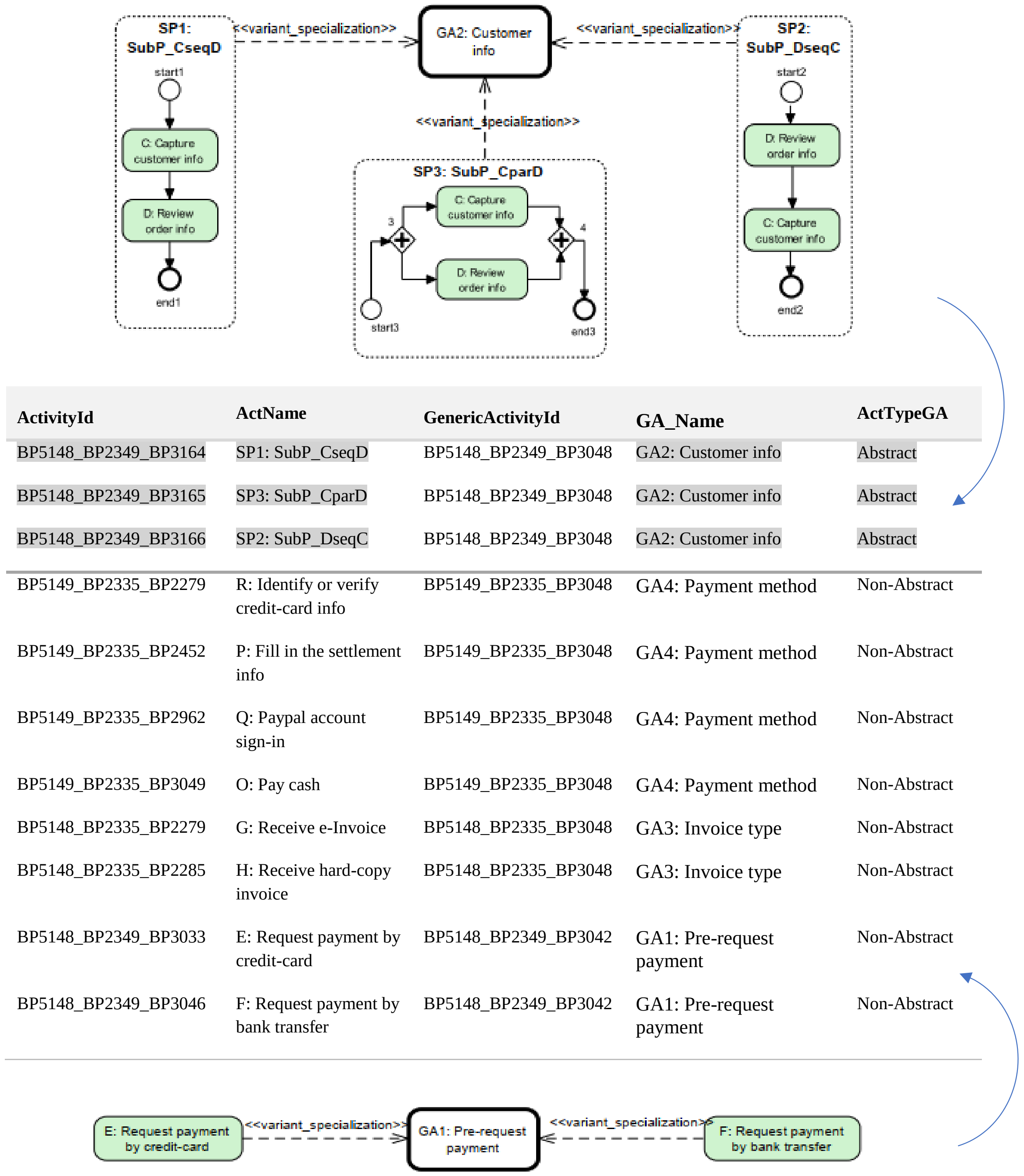
- custom subprocess e. g., *SP1: SubP_ CseqD*, *SP3: SubP_ CparD*, *SP2: SubP_ DseqC* as displayed in the first three rows of the following figure (highlighted in gray). Each of the subprocesses is composed of other step activities. Here, GA is defined as an *Abstract* activity type as displayed in the top of this figure.

Consequently, we distinguish between these specializations in Algorithm 3 to find the bounded steps activities of these specialized activities.

Firstly, we get the type and the name (statements in lines *2* and *3* respectively) of the specialized activity with id of the argument in the above procedure. We store all concrete steps (their ids, names and the concrete process id they belongs to) to a temporary set variable named *steps_ concrete*.

Secondly, we check if the type of the specialized activity is an EA(elementary activity) or a SP (subprocess) (see line *5* and *7* respectively of this algorithm) . For each specialized activities we join the steps from concrete processes that are activities and not control elements or events. For an EA specialized activity (see line *6*) we combine activities from concrete processes that match with the names (label) of the specialized activity by using procedure MatchLabelString()[9]. This label matching procedure is based on the longest common substring between two strings given as arguments. We show an example of this procedure in part (b) of Figure 3.27.
Whereas for a SP specialized activity (see line *9*) we find step activities of the custom subprocess by using the procedure FILTER_STEPS() that filters only the steps of current subprocess. Afterwards, we combine activities from concrete processes that match with the names of the steps part of the subprocess. We call the same procedure MatchLabel-String() to find the longest common substring between two strings.
For example, as shown in the following figure calling the procedure with two arguments: MatchLabelString('C: Capture customer info','Capture customer info') will result in finding the common substring *Capture customer info* which consists of 21 characters length.

[9]More information on this procedure to find the longest common substring is in (Factor, 2014)



ActivityId	ActName	GenericActivityId	GA_Name	ActTypeGA
BP5148_BP2349_BP3164	SP1: SubP_CseqD	BP5148_BP2349_BP3048	GA2: Customer info	Abstract
BP5148_BP2349_BP3165	SP3: SubP_CparD	BP5148_BP2349_BP3048	GA2: Customer info	Abstract
BP5148_BP2349_BP3166	SP2: SubP_DseqC	BP5148_BP2349_BP3048	GA2: Customer info	Abstract
BP5149_BP2335_BP2279	R: Identify or verify credit-card info	BP5149_BP2335_BP3048	GA4: Payment method	Non-Abstract
BP5149_BP2335_BP2452	P: Fill in the settlement info	BP5149_BP2335_BP3048	GA4: Payment method	Non-Abstract
BP5149_BP2335_BP2962	Q: Paypal account sign-in	BP5149_BP2335_BP3048	GA4: Payment method	Non-Abstract
BP5149_BP2335_BP3049	O: Pay cash	BP5149_BP2335_BP3048	GA4: Payment method	Non-Abstract
BP5148_BP2335_BP2279	G: Receive e-Invoice	BP5148_BP2335_BP3048	GA3: Invoice type	Non-Abstract
BP5148_BP2335_BP2285	H: Receive hard-copy invoice	BP5148_BP2335_BP3048	GA3: Invoice type	Non-Abstract
BP5148_BP2349_BP3033	E: Request payment by credit-card	BP5148_BP2349_BP3042	GA1: Pre-request payment	Non-Abstract
BP5148_BP2349_BP3046	F: Request payment by bank transfer	BP5148_BP2349_BP3042	GA1: Pre-request payment	Non-Abstract

Figure 3.26: Specialized activities either as elementary or as subprocess

And finally, we store the results of both bounded steps (by using *UNION* operation between union-compatible relations), either EA or SP to a new set named *bounded_ Steps* with value pairs (stepId, stepName, process_PId). We give a concrete example of the above algorithm in Figure 3.27. This figure consists of two parts:

- part (a) gives examples of bounded operations for each specialized activities, either

EA or SP

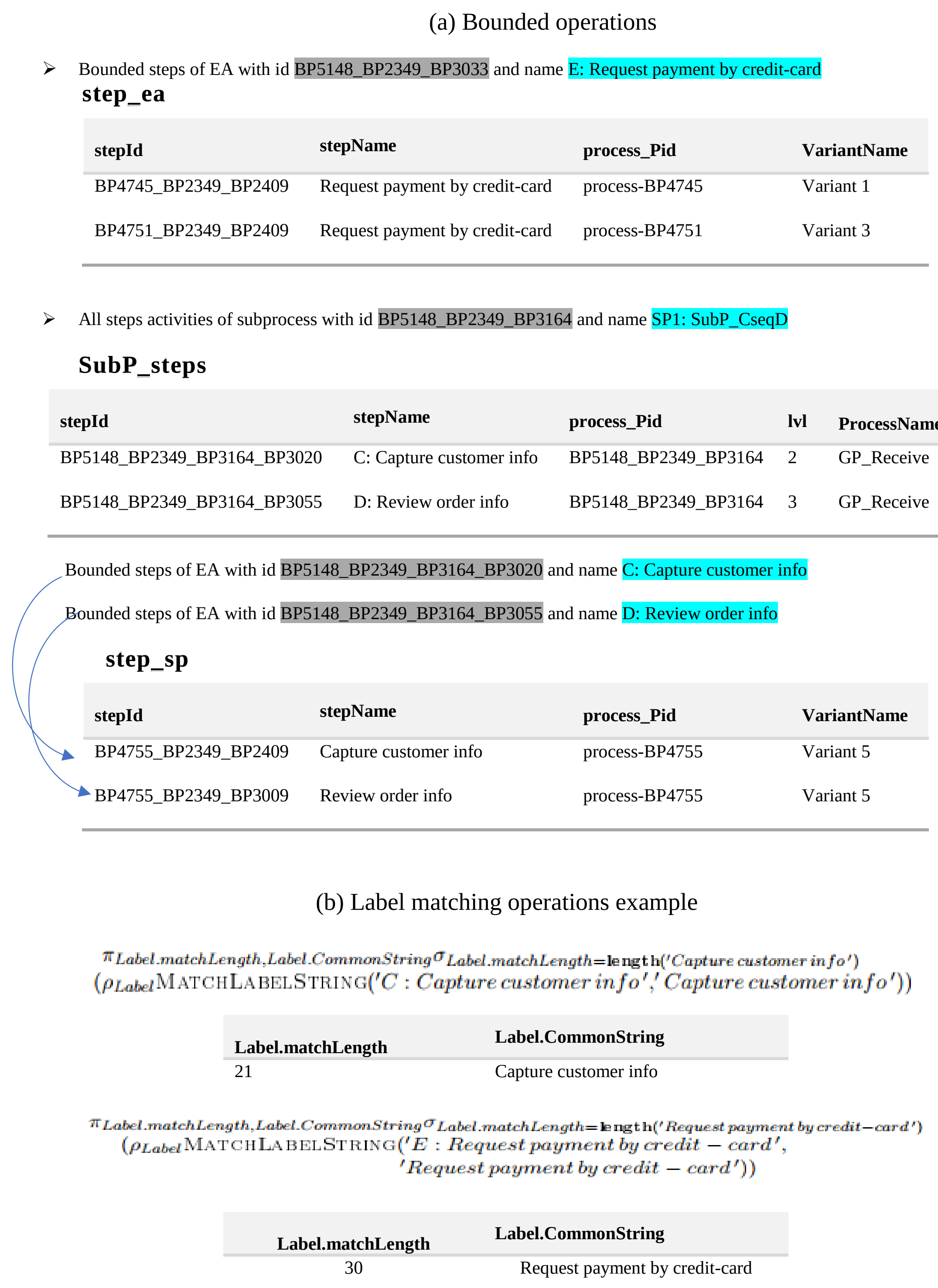
- part (b) gives examples of activity-label matching operations which basically compares two strings and outputs the longest common substring between them.

In part (a) EA and subprocess specialized activities are divided in two bullet sections.

In first section we demonstrate an example of an EA specialized activity named *E: Request payment by credit-card* and its respective bounded activities part of concrete process, i. e., activity with id: *BP4745_BP2349_BP2409* of Variant 1 and *BP4751_BP2349_-BP2409* of variant 3 both named *Request payment by credit-card*. We store these values in the set named *step_ea*. Of course, we get only the stepid, stepname, and processid values eventhough in the figure we show also variant name just to make it more clear for the reader. We restrict in listing only one example from EA specialized activity as another one is already explained in algorithm 2.

In second section we demonstrate an example of a subprocess specialized activity named *SP1: SubP_CseqD* which consists of two steps activities i. e., *C: Capture customer info* and *D: Review order info*. The other two event steps such as *start1* and *end1* are omitted from the result because we are interested in only activity steps. Moreover, we indicate the level of performing these activities sequentially after executing the procedure FILTER_-STEPS(). We enter these two rows to a temporary variable set named *SubP_steps*. Then, for each of them we find the respective bounded steps i. e., *Capture customer info* and *Review order info* (part of the concrete process in variant 5) and store them to the set named *step_sp*.

In part (b) we give two examples of procedure MatchLabelString() with two different case arguments. The first instance gives the longest common substring between an activity (i. e., a step from a subP specialized activity) and its bounded activities. Whereas, the second instance gives the longest common substring between an activity (i. e., a step from an EA specialized activity) and its bounded activities.



(a) Bounded operations

➢ Bounded steps of EA with id BP5148_BP2349_BP3033 and name E: Request payment by credit-card

step_ea

stepId	stepName	process_Pid	VariantName
BP4745_BP2349_BP2409	Request payment by credit-card	process-BP4745	Variant 1
BP4751_BP2349_BP2409	Request payment by credit-card	process-BP4751	Variant 3

➢ All steps activities of subprocess with id BP5148_BP2349_BP3164 and name SP1: SubP_CseqD

SubP_steps

stepId	stepName	process_Pid	lvl	ProcessName
BP5148_BP2349_BP3164_BP3020	C: Capture customer info	BP5148_BP2349_BP3164	2	GP_Receive
BP5148_BP2349_BP3164_BP3055	D: Review order info	BP5148_BP2349_BP3164	3	GP_Receive

Bounded steps of EA with id BP5148_BP2349_BP3164_BP3020 and name C: Capture customer info

Bounded steps of EA with id BP5148_BP2349_BP3164_BP3055 and name D: Review order info

step_sp

stepId	stepName	process_Pid	VariantName
BP4755_BP2349_BP2409	Capture customer info	process-BP4755	Variant 5
BP4755_BP2349_BP3009	Review order info	process-BP4755	Variant 5

(b) Label matching operations example

$$\pi_{Label.matchLength, Label.CommonString}\sigma_{Label.matchLength=\mathtt{length}('Capture\ customer\ info')}$$
$$(\rho_{Label}\textsc{MatchLabelString}('C: Capture\ customer\ info', 'Capture\ customer\ info'))$$

Label.matchLength	Label.CommonString
21	Capture customer info

$$\pi_{Label.matchLength, Label.CommonString}\sigma_{Label.matchLength=\mathtt{length}('Request\ payment\ by\ credit-card')}$$
$$(\rho_{Label}\textsc{MatchLabelString}('E: Request\ payment\ by\ credit-card', 'Request\ payment\ by\ credit-card'))$$

Label.matchLength	Label.CommonString
30	Request payment by credit-card

Figure 3.27: Examples of bounded activities from specialized activities (a) and examples of matching label's activities (b)

In the following section we configure abstracted activities of different process variants to specific levels positions of the hierarchy derived from Algorithm 2.

3.3.4 Consolidation hierarchy of activities and processes

After having the clear definition of when a process is a specialization of another one we can implement a process hierarchy. Such hierarchies can contribute to analysis of process variants by providing aggregation measures at different levels of this generic hierarchy. These hierarchies can contribute also to software (and design) reuse by providing a taxonomy of previous designs that can be searched easily.

Algorithm 2 determines basic operations how to derive the consolidation hierarchy of process variants steps. Whereas, Algorithm 4 after configuring the relative levels of each GAs, updates activities paired to GAs consolidated in the hierarchy to respective levels.

Accordingly, it starts altering the set derived from Algorithm 2, named *PV_Hierarchy_Level* where activities from concrete processes are consolidated(abstracted) to respective GA (generic activity). This alter consists of adding a new attribute to the set named *H_Level* (initially assigned a *NULL* value) to identify the genericity (relative) levels of the hierarchy. We should distinguish between *lvl* and *H_Level* attributes; the former one refers to the absolute level order of each GA in a specific generic process (GP). The latter one refers to a value assigned by generating a unique rank number for each **distinct** row according to a specified attribute value. Therefore, we use *DENSE_RANK()* function to assign rank to each row within a partition without gaps. Basically, starting at 1 for the first row according to a value (i. e., *lvl* as shown in line 3), then the ranks are assigned in consecutive manner, i. e., equal values are assigned with the same rank number, and next rank value will be one greater then the previous rank assigned. The result of performing the assignment of line 3 of this algorithm (after a group by operation) stored in a temporary variable set is as follows:

Table 3.8: An example of executing *DENSE_RANK()* function over *lvl*

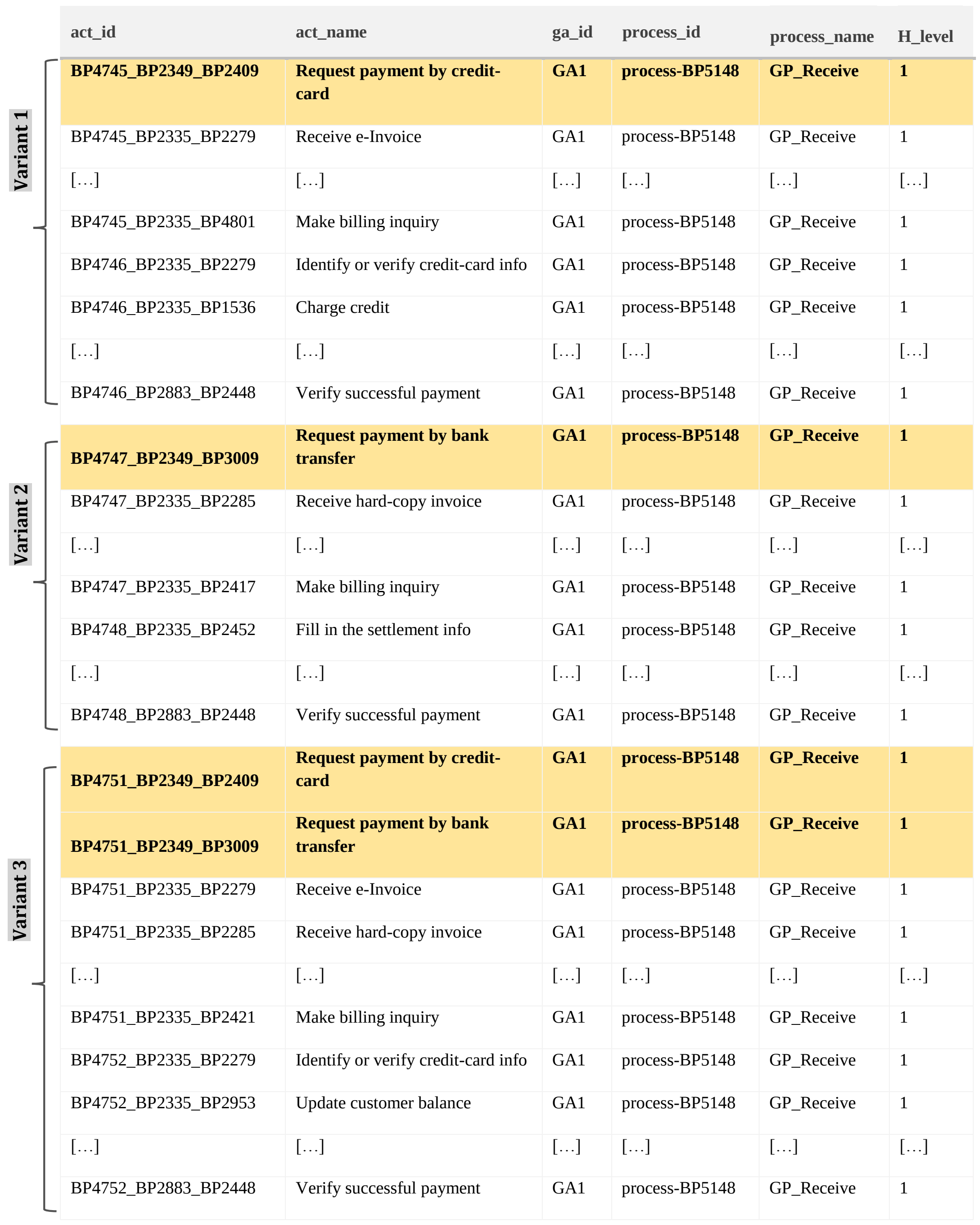
ga_id	ga_name	lvl	H_Level
BP5148_BP2349_BP3042	GA1: Pre-request payment	5	1
BP5148_BP2349_BP3048	GA2: Customer info	5	1
BP5148_BP2335_BP3048	GA3: Invoice type	7	2
BP5149_BP2335_BP3048	GA4: Payment method	16	3

As we can see from the Table 3.8 there are only $(1, 2, 3)$ ranking values instead of fourth because **GA1** and **GA2** are positioned at the same *lvl* order, i. e., level 5, ranked both with a value 1 (highlighted with light yellow color). After configuring the genericity levels of the hierarchy based on a generating ranking numbers, we update *H_Level* attribute values for each activity of *PV_Hierarchy_Level* that match with the *H_Level* values of *GA_order* set, i. e., with the ranking value of each GA. As a summary, step activities that are specialized (directly or indirectly) by a GA, are consolidated to the corresponding GA position of the hierarchy.

Algorithm 4 Configure relative levels of the derived process variants hierarchy and update activities according to these levels

```
Input: $PV\_Hierarchy\_Level \leftarrow \pi_{act\_id,ga\_id,process\_id,lvl}\ (derive\_PV\_Hierarchy())$

1: procedure ProcessVariants_Hierarchy_setLevel()
       ▷ alter set by adding $H\_level$ attribute for each step activ., init. a $NULL$ value
2:     $PV\_Hierarchy\_Level \leftarrow \{(act\_id, ga\_id, process\_id, lvl, H\_level)\}$
       ▷ update H_level by ranking rows according to $lvl$ values
3:     $GA\_Order \leftarrow ga\_id,\ lvl\ , \rho_{H\_level/\ DENSE\_RANK()\ OVER(\tau_{lvl})}$
                $\mathcal{G}_{ga\_id,lvl}(PV\_Hierarchy\_Level)$
4:     foreach $(ga\_id, H\_level) \in GA\_Order$ do
           ▷ update activity records to the corresponding $H\_level$ of each GA
5:         $PV\_Hierarchy\_Level \leftarrow \pi_{V.act\_id,V.ga\_id,V.process\_id,V.lvl,G.H\_Level}\ \sigma_{V.ga\_id=G.ga\_id}$
                $(\rho_G(GA\_Order) \bowtie \rho_V(PV\_Hierarchy\_Level))$
6:         get next value-pairs $(ga\_id, H\_level)$ from $GA\_Order$
7:     end foreach
8:     return $PV\_Hierarchy\_Level$
9: end procedure
```

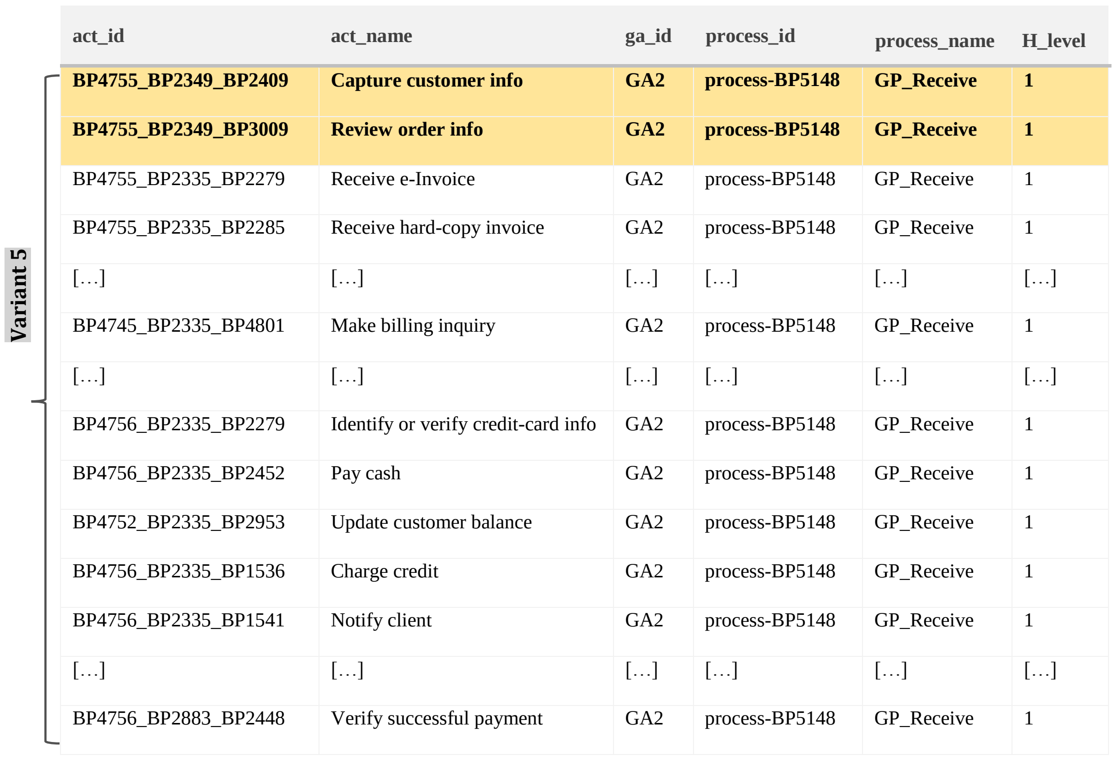
This algorithm updates step activities consolidated to different GAs corresponding to respective genericity levels of the hierarchy. As we can see from the figure different step activities are abstracted to respective GAs representing different levels of the process variants hierarchy. For example, activity *Identify or verify credit-card info* is positioned to different hierarchy levels, i. e., *GA1: Pre-request payment* after substituting the specialized activity *E1: Request payment by credit-card*, *GA3: Invoice type* after substituting the specialized activity *G1: Receive e-invoice* and *GA4: Payment method* after substituting the specialized activity *R1: Identify or verify credit-card info*. The same is applied to all other step activities of each process variant. We give a demonstration of these specialized direct and non-direct activities as instances step-by-step in the following Figure 3.28. The former activities are highlighted with a light orange color and the latter ones without a color. As shown in the first figure (a) instances about respective specialized activities of *GA1: Pre-request payment* are given. We see that instances from *Variant 4* are missing as they don't consider a pre-request payment activity. And in the second figure (b) we show instances about respective specialized activities of *GA2: Customer info* where results from only *Variant 5* are given. And the next figure (c) shows instances of specialized activities of *GA3: Invoice type* whereas the last one (d) shows instances of specialized activities of *GA4: Payment method*. Each of these sub figures display activities from concrete processes expanded at different levels of the hierarchy.


Variant 1
Variant 2
Variant 3

act_id	act_name	ga_id	process_id	process_name	H_level
BP4745_BP2349_BP2409	**Request payment by credit-card**	**GA1**	**process-BP5148**	**GP_Receive**	**1**
BP4745_BP2335_BP2279	Receive e-Invoice	GA1	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4745_BP2335_BP4801	Make billing inquiry	GA1	process-BP5148	GP_Receive	1
BP4746_BP2335_BP2279	Identify or verify credit-card info	GA1	process-BP5148	GP_Receive	1
BP4746_BP2335_BP1536	Charge credit	GA1	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4746_BP2883_BP2448	Verify successful payment	GA1	process-BP5148	GP_Receive	1
BP4747_BP2349_BP3009	**Request payment by bank transfer**	**GA1**	**process-BP5148**	**GP_Receive**	**1**
BP4747_BP2335_BP2285	Receive hard-copy invoice	GA1	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4747_BP2335_BP2417	Make billing inquiry	GA1	process-BP5148	GP_Receive	1
BP4748_BP2335_BP2452	Fill in the settlement info	GA1	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4748_BP2883_BP2448	Verify successful payment	GA1	process-BP5148	GP_Receive	1
BP4751_BP2349_BP2409	**Request payment by credit-card**	**GA1**	**process-BP5148**	**GP_Receive**	**1**
BP4751_BP2349_BP3009	**Request payment by bank transfer**	**GA1**	**process-BP5148**	**GP_Receive**	**1**
BP4751_BP2335_BP2279	Receive e-Invoice	GA1	process-BP5148	GP_Receive	1
BP4751_BP2335_BP2285	Receive hard-copy invoice	GA1	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4751_BP2335_BP2421	Make billing inquiry	GA1	process-BP5148	GP_Receive	1
BP4752_BP2335_BP2279	Identify or verify credit-card info	GA1	process-BP5148	GP_Receive	1
BP4752_BP2335_BP2953	Update customer balance	GA1	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4752_BP2883_BP2448	Verify successful payment	GA1	process-BP5148	GP_Receive	1

(a) Activities expanded at the 1st level of genericity

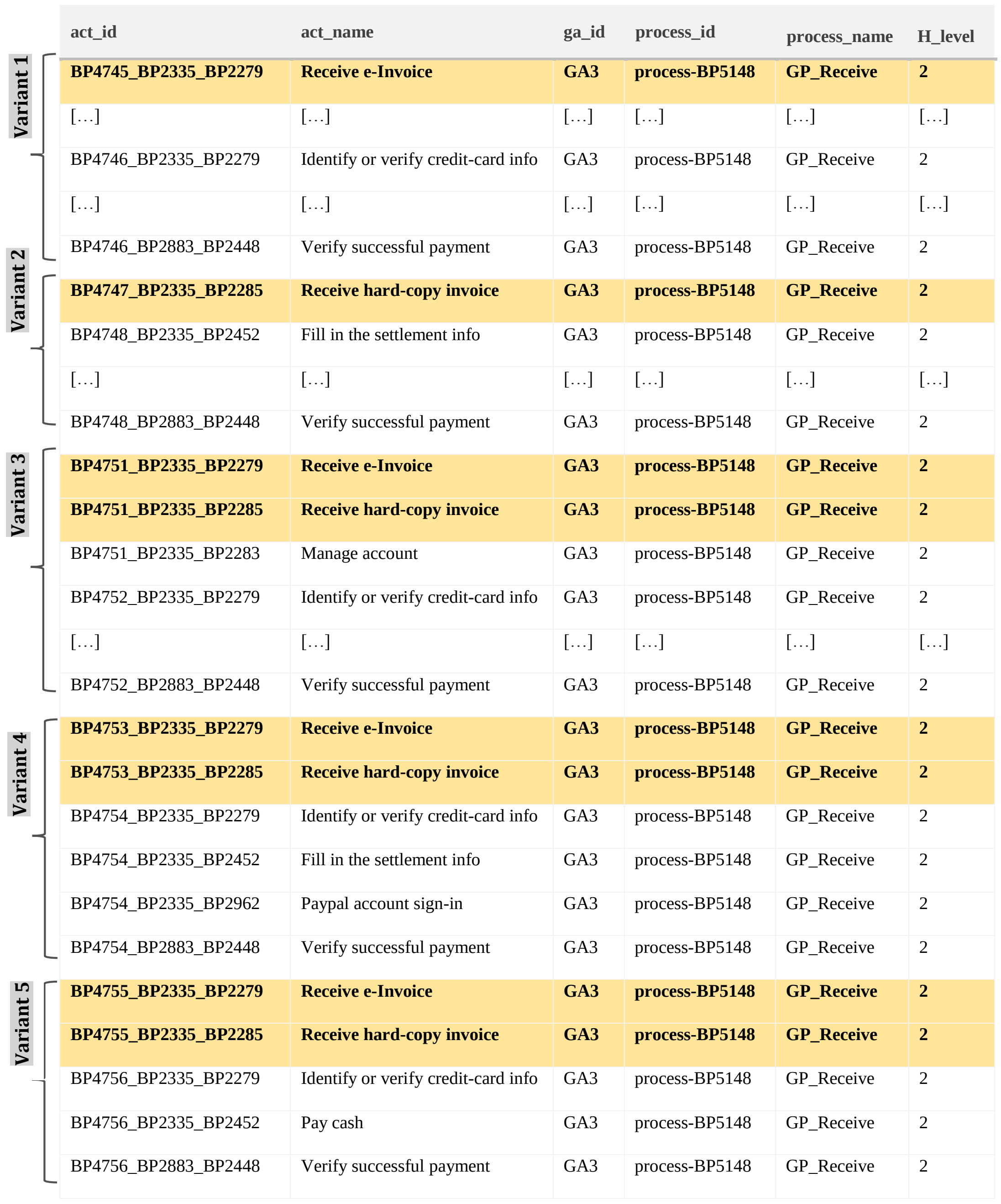
Figure 3.28: Process variants hierarchy expanded to different genericity levels



act_id	act_name	ga_id	process_id	process_name	H_level
BP4755_BP2349_BP2409	**Capture customer info**	**GA2**	**process-BP5148**	**GP_Receive**	**1**
BP4755_BP2349_BP3009	**Review order info**	**GA2**	**process-BP5148**	**GP_Receive**	**1**
BP4755_BP2335_BP2279	Receive e-Invoice	GA2	process-BP5148	GP_Receive	1
BP4755_BP2335_BP2285	Receive hard-copy invoice	GA2	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4745_BP2335_BP4801	Make billing inquiry	GA2	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4756_BP2335_BP2279	Identify or verify credit-card info	GA2	process-BP5148	GP_Receive	1
BP4756_BP2335_BP2452	Pay cash	GA2	process-BP5148	GP_Receive	1
BP4752_BP2335_BP2953	Update customer balance	GA2	process-BP5148	GP_Receive	1
BP4756_BP2335_BP1536	Charge credit	GA2	process-BP5148	GP_Receive	1
BP4756_BP2335_BP1541	Notify client	GA2	process-BP5148	GP_Receive	1
[…]	[…]	[…]	[…]	[…]	[…]
BP4756_BP2883_BP2448	Verify successful payment	GA2	process-BP5148	GP_Receive	1

(b) Activities expanded at the 1st level of genericity

Figure 3.28: Process variants hierarchy expanded to different genericity levels (cont.)



	act_id	act_name	ga_id	process_id	process_name	H_level
Variant 1	**BP4745_BP2335_BP2279**	**Receive e-Invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4746_BP2335_BP2279	Identify or verify credit-card info	GA3	process-BP5148	GP_Receive	2
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4746_BP2883_BP2448	Verify successful payment	GA3	process-BP5148	GP_Receive	2
Variant 2	**BP4747_BP2335_BP2285**	**Receive hard-copy invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4748_BP2335_BP2452	Fill in the settlement info	GA3	process-BP5148	GP_Receive	2
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4748_BP2883_BP2448	Verify successful payment	GA3	process-BP5148	GP_Receive	2
Variant 3	**BP4751_BP2335_BP2279**	**Receive e-Invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4751_BP2335_BP2285	**Receive hard-copy invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4751_BP2335_BP2283	Manage account	GA3	process-BP5148	GP_Receive	2
	BP4752_BP2335_BP2279	Identify or verify credit-card info	GA3	process-BP5148	GP_Receive	2
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4752_BP2883_BP2448	Verify successful payment	GA3	process-BP5148	GP_Receive	2
Variant 4	**BP4753_BP2335_BP2279**	**Receive e-Invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4753_BP2335_BP2285	**Receive hard-copy invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4754_BP2335_BP2279	Identify or verify credit-card info	GA3	process-BP5148	GP_Receive	2
	BP4754_BP2335_BP2452	Fill in the settlement info	GA3	process-BP5148	GP_Receive	2
	BP4754_BP2335_BP2962	Paypal account sign-in	GA3	process-BP5148	GP_Receive	2
	BP4754_BP2883_BP2448	Verify successful payment	GA3	process-BP5148	GP_Receive	2
Variant 5	**BP4755_BP2335_BP2279**	**Receive e-Invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4755_BP2335_BP2285	**Receive hard-copy invoice**	**GA3**	**process-BP5148**	**GP_Receive**	**2**
	BP4756_BP2335_BP2279	Identify or verify credit-card info	GA3	process-BP5148	GP_Receive	2
	BP4756_BP2335_BP2452	Pay cash	GA3	process-BP5148	GP_Receive	2
	BP4756_BP2883_BP2448	Verify successful payment	GA3	process-BP5148	GP_Receive	2

(c) Activities expanded at the 2^{nd} level of genericity

Figure 3.28: Process variants hierarchy expanded to different genericity levels (cont.)

	act_id	act_name	ga_id	process_id	process_name	H_level
Variant 1	**BP4746_BP2335_BP2279**	**Identify or verify credit-card info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4746_BP2335_BP1536	Charge credit	GA4	process-BP5149	GP_Pay	3
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4746_BP2883_BP2448	Verify successful payment	GA4	process-BP5149	GP_Pay	3
Variant 2	**BP4748_BP2335_BP2452**	**Fill in the settlement info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	BP4748_BP2883_BP2448	Verify successful payment	GA4	process-BP5149	GP_Pay	3
Variant 3	**BP4752_BP2335_BP2279**	**Identify or verify credit-card info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	BP4752_BP2335_BP2973	**Fill in the settlement info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	BP4752_BP2335_BP2953	Update customer balance	GA4	process-BP5149	GP_Pay	3
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4752_BP2883_BP2448	Verify successful payment	GA4	process-BP5149	GP_Pay	3
Variant 4	**BP4754_BP2335_BP2279**	**Identify or verify credit-card info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	BP4754_BP2335_BP2452	**Fill in the settlement info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	BP4754_BP2335_BP2962	**Paypal account sign-in**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4754_BP2883_BP2448	Verify successful payment	GA4	process-BP5149	GP_Pay	3
Variant 5	**BP4756_BP2335_BP2279**	**Identify or verify credit-card info**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	BP4756_BP2335_BP2452	**Pay cash**	**GA4**	**process-BP5149**	**GP_Pay**	**3**
	[…]	[…]	[…]	[…]	[…]	[…]
	BP4756_BP2883_BP2448	Verify successful payment	GA4	process-BP5149	GP_Pay	3

(d) Activities expanded at the 3^{rd} level of genericity

Figure 3.28: Process variants hierarchy expanded to different genericity levels (cont.)

Accordingly, we demonstrate graphically the full expanded hierarchy of these process variants at different levels of the genericity in Figure 3.29 . We highlight with yellow and green color direct and non-direct specialized activities respectively that are expanded in each of the level of hierarchy. Obviously, the highest position of the hierarchy will give the full view of all activities among process variants. Instead, moving down into the hierarchy the view on these activities will get reduced and will comprise only a fragment of the process behaviour among variants.

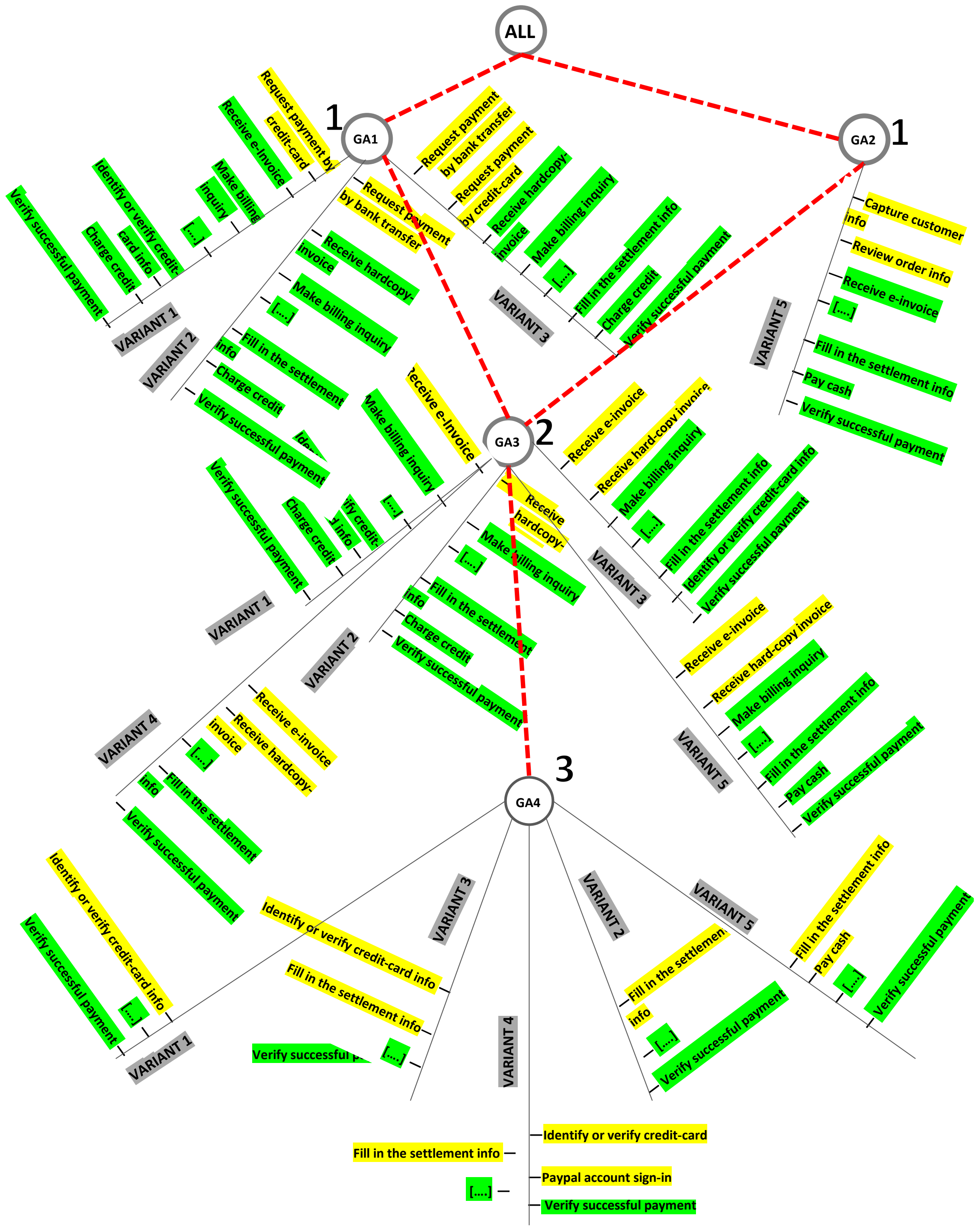


Figure 3.29: A full view of the expanded process variants hierarchy

In the following we compare all current meta-modelling approaches proposed by the research community to manage process variants.

3.4 Comparative analysis of current approaches

Recently, some comparative studies have been reported in business process variability domain. (Rosa et al., 2017) conducted a systematic inventory of approaches to customizable process modelling. The authors identify and classify major approaches and provide a comparative evaluation with the objective to answer three research questions (such as represent common and distinct features of customizable process modelling approaches and research gaps exists in current LR). (Ayora et al., 2015) conducted a systematic literature review to evaluate existing variability support across all phases of the business process life cycle. The authors considered and categorized primary studies based on eight research questions (such as underlying business process modelling language used, tools available for enabling process variability, and validation of methods proposed). They developed a framework, called VIVACE to enable process engineers to evaluate existing process variability approaches. Then, they evaluate their framework against three main approaches from LR: C-EPCs, Provop, and PESOA. Our survey differs from theirs, as we restrict our search to only select those approaches (five out of twenty-eight papers from digital libraries) that introduce a meta-model for capturing variants of a business process. Some other work from (Valença et al., 2013) focus on identifying not only characteristics of business process variability but also challenges in this field through a literature mapping study. But they didn't compare or analyse the surveyed approaches from LR. Whereas, (Tőhring et al., 2014) compare two approaches (C-YAWL and vBPMN)on the basis of a reference process model but using different types of configuration and adaptation mechanisms. In contrast, the present survey as a first step describes each main approach in detail (see Section 3.2) secondly, applies an example to it, and last draws a comparative analysis based on evaluation of each criterion derived from our LR.

Table 3.9 summarizes the evaluation results for process variants meta-models approaches. Each column indicates to what extent the approach in question covers each evaluation criterion defined as follows. We used a "+" sign to indicate a criterion that is fulfilled, a "–" sign to indicate a criterion that is not fulfilled, and a "+/–" sign to indicate partial fulfillment. The first column lists the sixth main approaches including our approach. The next sixth columns indicate the coverage of each criterion. The last column indicate the modelling language(s) underlined by each approach.

RQ1: Which process types and process perspectives are covered by process variability meta-models? From results of literature review (LR) in respect to this research question we can conclude that two type of processes exist: design-time(i. e., variations is considered only during process modelling phase) and runtime (i. e., variations is considered only during process enactment for example to handle exceptions). Whereas, process perspectives may categorized the surveyed approaches to mainly functional (what activities are captured) and behaviour perspectives (the control-flow sequence), eventhough some approaches deals somehow also with some aspects of organizational (resources to be consumed) and informational perspective (consumption of data). Thereof, the criteria derived from **process types** results are:

- **Conceptual**: If an approach is designed to support conceptual modelling only than this means that variability is captured during process definition and these variant models will not be executed on top of a BPMS. Thereof, we say that this approach meets this criterion.

- **Executable**: An approach meets this criterion if variability is considered for process models that are meant to be executed by a typical BPMS. Moreover, during their enactment there are no inconsistencies reported in associating between different elements of a process model (e. g., activities and their data input or output).

Moreover, for **process modelling perspectives** results the criteria derived might be:

- **Control-flow**: An approach meets this criterion if variability is captured along with activities and decision gateways that might become variation points (e. g., capture a skipped activity in one of the variants).
- **Resources**: If the variability is captured in the participated resources (human or system) that are planned to perform different tasks. In so doing, resources can become variation points (e. g., a typical resource is not performing in some of the process variants). If the approach does not represent them graphically but it is only mentioned then we say that the approach partially fulfills this criterion (Rosa et al., 2017).
- **Data Objects**: An approach meets this criterion if data objects (i. e., produced-input data objects and consumed-output data objects) might become variation points. For example, a pay invoice confirmation is not captured in one of the variants of a order-to-pay process. If the approach does not represent them graphically but it is only mentioned then we say that the approach partially fulfills this criterion.

RQ2: Which supporting technique is used to introduce or capture variability between process models? In respect to this research question, the criteria derived from **supporting technique** results are:

- **behavioural**: The approach takes as input a collection of process variants and derive a process variant by hiding and blocking process elements. Any behavioural anomalies such as deadlocks should be avoided.
- **Structural**: The approach takes as input a base process model and after applying a set of change operations to it a process variant is derived. Any structural anomalies such as disconnected activities should be avoided.

According to the technique supported by specific approach some transformations should be done to process model in order to derive a variant. These transformations (by restriction/extension) might be categorized as criteria for:

- **Restriction**: An approach matches this criterion if a process model is configured by restricting its behaviour.

Table 3.9: Comparative analysis of approaches for business process variability management

Main Approaches	Process Type		Process Perspective			Supporting Techniques		Variability Type		Process Specialization	Process modelling Language
	Conceptual	Executable	Control-flow	Resources	Objects	behavioural	Structural	Restriction	Extension		
PESOA (Bayer et al., 2005)	+	−	+/−	−	+	−	−	+	−	−	BPMN, UML ADs
BPFM (Moon et al., 2008)	+	−	+/−	−	−	−	−	+	−	−	UML ADs
vBPMN (Dőhring and Zimmermann, 2011)	+	+/−	+	−	−	+	+	−	+	−	Block-structured BPMN
PF (Kulkarni and Barat, 2011)	+	−	+/−	−	−	−	−	+	−	−	BPMN
C-iEPC (Rosa et al., 2011)	+	−	+	+	+	+	+	+	−	−	C-iEPC
Our method PV Hierarchy	+	+	+	+	−	+	−	+	−	+	BPMN

- **Extension**: An approach matches this criterion if a process model is configured by extending its behaviour

RQ3: Which process is a specialization(generalization) of another process? Currently, to the best of our knowledge there is no approach that gives an insight which of the process is a specialization of another one. Then, a valid criteria derived might be:

- **Process Specialization**: Specialization relationship between processes. An approach matches this criterion if a specialization/generalization relationship exists among processes.

From the information of the above table we can conclude that all approaches covers the the conceptual level of process variants and the control-flow process perspective. Whereas, (Rosa et al., 2011) captures variability in the participated resources (human or system)and data objects too. The supporting technique used to introduce or capture variability between process models is proposed by us, (Rosa et al., 2011) and (Dőhring and Zimmermann, 2011) and the latter proposed both *restriction* and *extension* to capture variability. And our method meets the criterion about *process specialization* which is not mentioned in any of the methods discussed.

3.5 Summary and discussion

In this section 3.5 we summarize the most important aspects of Chapter 3 as one of the key chapters of this research work. We started represented all current meta-model approaches of process variants and compare our approach against each of them based on different criteria. After comparing we can categorize all of them into groups based on variation of business processes: *Activity Specialization* and *Node Configuration* as shown in Figure 3.30

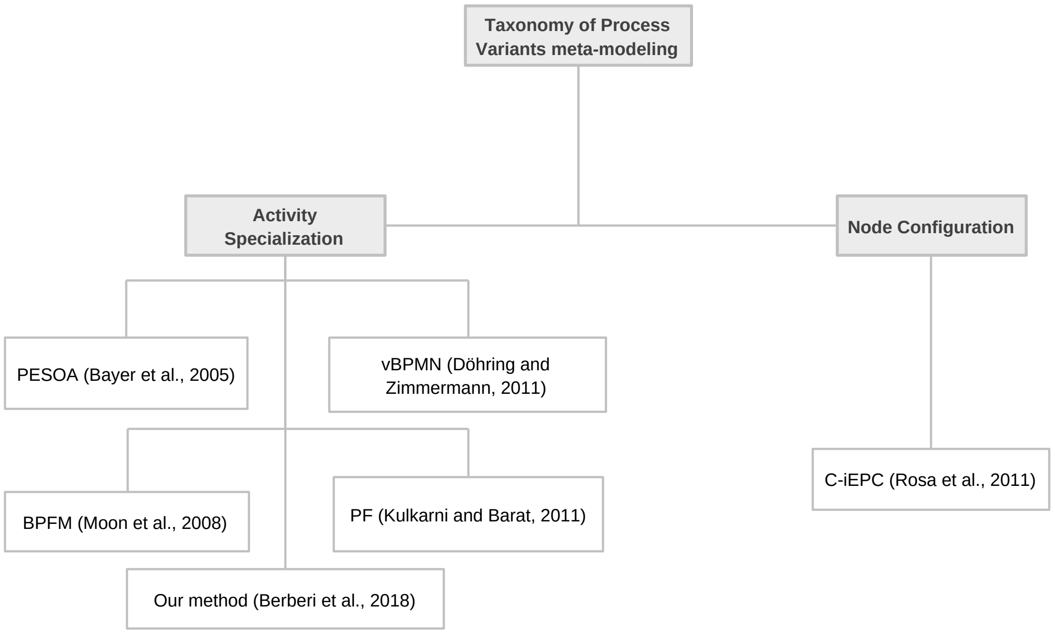

Taxonomy of Process Variants meta-modeling
Activity Specialization
Node Configuration
PESOA (Bayer et al., 2005)
vBPMN (Döhring and Zimmermann, 2011)
BPFM (Moon et al., 2008)
PF (Kulkarni and Barat, 2011)
Our method (Berberi et al., 2018)
C-iEPC (Rosa et al., 2011)

Figure 3.30: Taxonomy of process variants meta-modelling approaches

We presented a full meta-model capturing both business processes design-time and run-time aspects. Our novel meta-model introduces two new concepts: *Generic Activity* and *Generic Process* to capture variability in activities of a collection of process variants. We described in detail each concept used and how to define specializations between activities and processes. Next chapter discusses current analysis of business processes (not process variants) with a process warehouse approach.

CHAPTER 4

Process Warehouse for Business Processes Analysis

Chapter 4 gives a background of the two main research topics, business process management and process warehouse. Section 4.2.2 provides an overview of process warehouse modelling approaches whereas section 4.2.3 presents multidimensional process mining approaches. Section 4.4 concludes with a summary of the proposed approaches to analyse and improve business processes continuously through the most important analysis parameters which are elicited from models of business processes.

4.1 Introduction

In this section we introduce the relevant background for this research topic, w.r.t., business process management and process warehouse as follows.
Business Process Management
Business process management is based in the explicit representation of business processes with their activities and the execution constraints between them. Business processes can be subject to analysis, improvement, and enactment once they are defined (Weske, 2012). Performance evaluation phase also known as performance analysis or post-execution analysis completes the BPM life cycle. It uses the information generated during process execution to assess process performance that serve as a basis to improve processes and their execution performance. Process performance analysis concerns two mainstreams approaches i. e., *process model-based analysis* (calculate performance indicators based on the alignment of executed event data to an apriori process model) and *process execution-based analysis* (calculate performance indicators based on the alignment of executed event data to a mined process model). Recently, some process analysis tools offer a combination use of models and timed or resourced event data using different performance measures. By aligning an event log with timestamps on a process model, one can measure delays, for example, the time in-between two subsequent activities. The result can be used to highlight bottlenecks and gather information for simulation or prediction techniques (Van Der Aalst, 2013).

Process Warehouse

Here, we introduce some different interpretation of the PW concept according to different authors sorted by least recent first. Scheer who was considered as one of the pioneers of BPM, in the early of '90ies gave a comprehensive definition of PW: "The result of systematically capturing, storing and maintaining business process know-how in a repository is called a process warehouse "(Scheer and Nüttgens, 2000).
Process Warehouse has been introduced as an alternative to the data warehouse (DW): it is defined as a "warehouse that stores data about executed business processes, such as actors, activities performed by actors, the execution time of activities, the frequency of activities and the availability of resources and thereby can be used as an adequate basis for the analysis and optimization of those processes" according to (List et al., 2000). After two years, (List et al., 2002b) modified the definition of process warehouse as "a separate read-only analytic database, which provides the foundation of a process oriented decision support system with the aim to analyze and improve business processes continuously".

Based on these characterizations of process warehouse, we adopt the following definition.

Definition 4.1.1. *A* process warehouse *is a data warehouse that stores data about process model variants and their instances as an adequate basis for their analysis and improvement.*

Many researchers focus on the execution analysis of processes using a data warehouse technology because analysis is usually on aggregated performance data and not on individual cases (Grigori et al., 2004). Data warehouse technology (together with decision support tools like online analytical process systems and data mining techniques (Chaudhuri and Dayal, 1997) has been implemented in several domains, such as healthcare and banking, to enhance the analytical capabilities of enterprises.
However, the technology does not provide adequate bases for performance analyses of business processes' execution. It is because data stored in a DW is extracted from different heterogeneous transaction processing systems and thereby, lack in providing the facts concerning business process executions (Marjanovic, 2007) i. e., the information about start time of a process, end time of process, the activities executed for a process instance, the actors responsible for the execution of a process and resources consumed in a process are not captured in the data warehouse. Thereof, valuable decisions on potential process improvements cannot be taken by using conventional data warehouses (Fabio et al., 2002; Grigori et al., 2001; Sayal et al., 2002). To overcome this problem, a new concept of process-oriented data warehouse (Casati et al., 2007), or shortly named process warehouse (PW) (List et al., 2000) has been introduced. Eventhough, very few papers (Benker, 2016; Koncilia et al., 2015; Pau et al., 2007; Eder et al., 2002) introduced process warehouse as an appropriate means for analysing the performance of business process execution using well established data warehouse technology and on-line analytical processing (OLAP) tools. Whereas, approaches like (Shahzad and Zdravkovic, 2012; Niedrite et al., 2007) presented a goal-driven (i. e., a goal is a state of a process in terms of the quality of the service property that is intended to be achieved) requirement analysis (Giorgini et al., 2005) together with the method to obtain the conceptual model of a data warehouse.

Some other researchers from process mining discipline related to the well-known OLAP cube like (Vogelgesang and Appelrath, 2015) proposed relation data warehouse schema (*PMCube*), *event cube* (Ribeiro and Weijters, 2011) and *process-cube* (van der Aalst, 2013; Mamaliga, 2013; Bolt and van der Aalst, 2015) for an efficient multi-dimensional event pattern analysis at different abstraction levels.
In this research, we focus on analysing different event data generated from different variants using a process warehouse technology. More information about design challenges of the proposed process warehouse-based approaches, multidimensional process mining approaches and their usage we discuss in the following sections.

4.2 State-of-the-art of process warehouse for business process analysis

This section provides an overview of current process-oriented data warehouses (shortly named process warehouse) to analyse and improve business processes continuously through the most important analysis parameters which are elicited from the generic meta-model of business processes. After giving a brief overview in the discipline of process warehouse we discuss design challenges and issues in subsection 4.2.1. In the following subsections 4.2.2 and 4.2.3 we classify proposed *process warehouse-based* approaches suitable for analysing business processes. Moreover, an evaluation of approaches is drawn in section 4.3.

4.2.1 Process warehouse design challenges

Different approaches have been proposed to Process Warehouse (PW) development process. In this subsection, we survey the literature related to the key design issues and challenges of process warehouse approaches.
We take into account only the issues and challenges of building a data warehouse for processes and not of a data warehouse in general.
modelling challenges addressed by a process warehouse originates from constraints related to multidimensional model constraints itself, which are fundamentals and thus may not be violated or trivially overcome. A list of these challenges and issues collected from sources (Berberi et al., 2018; Mansmann et al., 2007b; Bonifati et al., 2001; List and Korherr, 2006; Jarke et al., 2000) and recently reported only superficially in (Shahzad and Johannesson, 2009) is presented as follows:

- *Heterogeneity of fact entries*: Processes consist of heterogeneous components, such as activities and events. If we model *Component* as one fact type this will lead to loss of subclass properties, as all fact entries must fully adhere to the fact scheme, i. e., have the same dimensional characteristics and uniform granularity in each dimension. While mapping subclasses to separate fact types disables treating all components as the same class in part of their common properties (Mansmann et al., 2007b).

- *Conceptually complex aggregations*: Defining a summary table is a crucial issue. Aggregate functions and dimension hierarchy values are required in summary tables. All entries of the same fact type should roll up along the same set of aggregation paths. Generating aggregate data, e. g., rate workflow resources, has been one of the most valuable subjects of developing a packaged solution for collecting and analysing workflow execution data for HP Process Manager (Bonifati et al., 2001).

- *Many-to-many relationship between fact and dimension (M:M)*: Sometimes a bridge table is used to allow many to many relationship, e. g., multiple work items may be allocated to multiple participants. A bridge solution is appropriate if the many-to-many relationship is simply providing greater context and does not affect the measures. However, (Kimball and Ross, 2013) stresses that every table that expresses a many-to-many relationship must be a fact table. This "law" of Kimball prohibits non-strict hierarchies and many-to-many relationships between facts and dimensions (Mansmann et al., 2007b)

- *Generalization/specialization relationship*: Features such as specialization or generalization relationship are not yet introduced in consolidating steps of different models (variants) of the same business process in one dimension (Berberi et al., 2018). For example, from process models sequence patterns such as A->B->E and A->B->E->F can be revealed where A, B, E, F are activity steps. Thus, we can infer that A->B->E is more general than A->B->E->F if a generalization relationship exists among them. More details about this approach the reader can refer to Chapter 5.

- *Integration of business process context perspective*: Allow business process to be presented from a wide angle. It provides an overview perspective of the process and describes major business process characteristics, such as goals and their measures, the deliverables, the process owner, the process type and the customer at a glance (List and Korherr, 2006)

- *Interchangeability of fact/measure and dimension roles*: The case where a fact scheme act as a dimension of another fact scheme. For example, in a surgical workflow scenario: Surgery has dimensional characteristics of its own (Location, Patient, etc.) and therefore, it can be treated as a fact type. However, with respect to single work steps, *Surgery* clearly plays the role of a dimension e. g., *Event* rolls-up to *Surgery* (Mansmann et al., 2007b).

- *Diversity and evolution management*: When new features are added to workflow models, supported by different commercial products, then consequently many efforts may be required during the redesign of data warehouse. In addition, a data warehouse for workflow logs should be able to host data that comes from different heterogeneous sources e. g.,histories of engineering processes and products (Jarke et al., 2000), i. e., different WfMSs, and not only warehousing HP audit trail logs as it is presented in HP Process Manager tool (Bonifati et al., 2001; Fabio et al., 2002)

4.2.2 Process warehouse modelling approaches

Specific studies on PWs have been mainly focused on proposing the PW design schema (multidimensional data model), generic solution (Benker, 2016; Koncilia et al., 2015; Casati et al., 2007), or specific to a certain domain, like chemical engineering (Brandt et al., 2006) and healthcare(Mansmann et al., 2007a). Although some of these studies explicitly state that an excerpt of the data model is presented, analyses of PW design proposals conclude that the presented data models are incomplete and therefore cannot be used for analysis of business processes (Shahzad and Johannesson, 2009). (Brandt et al., 2006) presented an extended design of PW that is capable of capturing two types of traces to facilitate the support of creative design processes. The two types are, *product* and *process traces*. The former describe the properties and relationships of concrete design objects and later describe the history of actions that lead to the creation of the product objects (Shahzad, 2012). We conduct a comprehensive search through main digital libraries like (IEEE, ACM, Springerlink, Elsevier) using keywords such as, data warehouse for business process, process analysis, multidimensional process, process-oriented data warehouse. During our survey, we have found 15 process warehousing approaches that are adequate to be used for business process analysis after classifying only those that covers process warehouse design aspects. And other 5 papers introduced relational/multidimensional data warehouse design for process mining analysis. We extend the work reported from (Shahzad and Johannesson, 2009) introducing four new process warehouse-based approaches (Shahzad and Zdravkovic, 2012; Koncilia et al., 2015; Benker, 2016; Berberi et al., 2018) proposed recently. Furthermore, we discuss if respective design challenges have been considered for each approach.
A short overview of these approaches is as follows:

Our method (Berberi et al., 2018): *A Process Warehouse Model Capturing Process Variants*- We present the core of a process warehouse model with a generalization hierarchy of processes which captures process variants. This generalization hierarchy can be generated from a meta-model of business process models which introduces the notion of generic activities which generalize a set of activities (e. g., pay by credit card, by check, or by third-party (PayPal) could all be generalized to an activity payment). Based on given hierarchies of activities we define generalizations of processes for the "process" dimension of a process warehouse. This hierarchy can then be used to roll-up or drill down when analysing the logs of the executions of the various process variants and it makes it much easier to compare KPIs between different variants at different levels of genericity.

(Benker, 2016): *A Generic Process Data Warehouse Schema for BPMN Workflows*- The approach proposes to derive a generic data warehouse structures from the meta model of the BPMN, the actual de-facto standard of workflow languages. Generic offers portability between application domains and stability in case of changing workflows. For representing the multidimensional schema, the semantic data warehouse model is used. A dimension is shown with all of its hierarchy levels and the aggregation relationships between these levels.

(Koncilia et al., 2015): *A Generic Data Warehouse Architecture for analysing Workflow Logs*- This approach proposes a Sequence Warehouse (SeWA) architecture and OLAP tools to analyse data stemming from workflow logs, including process variants. A data structure called *Sequence Cube* is created with a dimension representing patterns to overcome the obstacles caused by branches and loops in a process. A conceptual model for DW is missing.

(Shahzad and Zdravkovic, 2012): *Process warehouses in practice: a goal-driven method for business process analysis-* This approach is based on goal-oriented methodology for requirement analysis in order to design a data warehouse. The goal-oriented methodology is used within a hybrid approach mixing demand-driven and supply driven design framework to produce data warehouse design. Requirement analysis is based in two different perspectives: organizational modelling, centered on stakeholders, and decision modelling, focused on decision makers. From the two perspectives, goal analysis, fact analysis and attributes are identified. Requirements are then mapped onto source schema and hierarchies are constructed and refined. Therefore, for the purpose of identifying and retrieving automatically the necessary and sufficient information from the PW for process analysis, they linked a goal to PW content, thereby creating traceability between the PW and the goals. They developed a prototype that consists of three modules (goal, link and analysis module) to demonstrate the applicability of the proposed method. In this approach, 'dimensional fact model' is used for conceptual modelling of process warehouse.

(Niedrite et al., 2007): *Goal-Driven Design of a Data Warehouse-Based Business Process Analysis System-* This approach is based on defining goals, identifying quantifiable questions and the related indicators that will help in achieving the measurement goals. Their method consists of deriving a data warehouse model from indicator definitions with Object Constraint Language (OCL) to accomplish the process measurement. Then to identify potential facts and dimensions of the DW, the structure of the OCL expressions is thoroughly analysed. They implemented their method using the university data warehouse as a project case study. In this approach, common logical modelling technique (UML notation) is used for conceptual modelling of process warehouse.

(Neumuth et al., 2008): *Data warehouse technology for surgical workflow analysis-* According to this approach, a conceptual model of a surgical workflow is obtained in accordance with the multidimensional data model, which can be derived from the Extended Entity Relationship (EER) by examining relationship cardinalities and functional dependencies between attributes. They adopted a data warehousing technology for the requirements of surgical data and analysis. To enable additional aggregation levels, member values within a dimension are further organized into classification hierarchies (e. g., date can be aggregated by month or by week). In this approach, 'dimensional fact model' is used for conceptual modelling of process warehouse.

(Casati et al., 2007): *A generic data warehousing business process data-* The approach presents a generic solution for warehousing business process data. Two levels of representing facts are concerned: step-level and process instance level. At the step level, all execution facts are modeled in a single table, with one row per step execution (relevant step states are create, activate and complete). To correlate step-level information with process instance or business data, which sometimes can occur, they introduced links (referencing columns) between the fact tables at the step and process granularities (from a step fact to its process instance fact, and from the process instance fact to its business data fact). In this approach, common logical modelling technique (ER notation) is used for conceptual modelling of process warehouse.

(Pau et al., 2007): *Data warehouse model for audit trial analysis in workflows-* This approach proposes a conceptualization of key elements in workflow audit trail data that are relevant to the evaluation of business process performance. An Object Role modelling (ORM) was presented as a conceptual model for audit trails, which comprises a set of entity types and a set of relationships. They construct logical models for workflow evaluation using ADAPT (Application Design for Analytical Processing Technologies) notation. The derivation process that takes data out from other information sources is expressed in ER (Entity Relationship) or in dimensional modelling. They apply dimensional modelling techniques to define schemas for storing workflow audit trail data in data warehouses. In order to allow many-to-many relationship between the dimension tables and fact tables they used a bridge table solution. They outline a sample performance analysis of an order fulfillment process workflow in a typical e-businesses scenario. Key performance indicators such as Customer Satisfaction is presented in the underlined business process, which is defined in terms of service level (i. e., rate of the customer orders which can be delivered on time) and cycle time (i. e., total elapsed time from the receipt of a customer order till the delivery of requested items). Common logical modelling technique is used for modelling process warehouse.

(Mansmann et al., 2007a): *Multidimensional data modelling for business process analysis-* The approach is based on re-engineering the business process modelling in conformity with the multidimensional data model. As the source models (business process) and target models (OLAP) have conflicts and incompatible objectives, e. g., business process modelling is concerned with efficiency in operational level and workflow behavior whereas OLAP is concerned with aggregating over accumulated numerical data modeled as a set of uniformly structures fact entries. They illustrate a recording scheme of a surgical process model as a case study. They focus on obtaining the structure of data cubes by applying vertical and horizontal process decomposition. By applying vertical decomposition they identify core elements of a process by determining two granularity levels of the facts,e. g., *Surgery* and *Activity*, *State*, *Event*, whereas, by applying horizontal decomposition they identify the dimensions of a data cube which is drawn by recognizing different complementary perspectives in a workflow model, following the factual perspective such as function, organization, and operation. 'Dimensional fact model' is used for conceptual modelling of process warehouse.

(Giorgini et al., 2005): *Goal-oriented requirement analysis for data warehouse design-* The approach focus on goal oriented methodology for requirement analysis in order to design a data warehouse. A mixed combination between demand-driven and supply-driven is used to produce data warehouse design. For requirement analysis two different perspectives (organizational modelling and decisional modelling) are integrated. Requirements are then mapped onto source schema and hierarchies are constructed and refined. In this approach, 'dimensional fact model' is used for conceptual modelling of process warehouse.

(Schiefer et al., 2003): *Process data store: A real-time data store for monitoring business processes-* The approach focus on providing nearly real-time access to critical performance indicators of business processes to improve the speed and effectiveness of workflows. The process data store has two types of data, a) very detailed event data, b) detailed up-to-date process data at various granularity levels. Common logical modelling technique is used for conceptual modelling of process warehouse. In this approach, 'ADAPT notations' are used for modelling of process warehouse.

(Eder et al., 2002): *A data warehouse for workflow logs*- The approach aims in exploiting the valuable information stored in a workflow log using a data warehouse technology. This approach can be categorized as a hybrid approach: information supply (data-driven) is provided by the structure data of the workflow logs and information demand (user-driven) is provided by a set of interesting queries. The DW schema is designed taking into account the possibility to answer all the formulated queries and some of the dimensions such as workflows and participants are resulted directly from the generic meta-model. This DW is used for analysing and monitoring processes, discovering process instances and activities, which regularly lead to deadline misses. In this approach, 'ADAPT notations' are used for modelling of process warehouse.

(Kueng et al., 2001): *A holistic process performance analysis through a performance data warehouse*- According to this approach a DW can be used to facilitate business process improvement based on holistic performance measurement. In this approach two databases are designed: a relational DW that stores the measures related with the dimensions; and a database that stores auxiliary data describing the following entities: Organization (employees, departments, business processes, business units and their manager), Goal tree (primary goals of the organization and the derived goals for the corresponding organizational entities and their specification by measures) and Access rights for the performance data and the auxiliary data. In this approach, 'ADAPT notations' are used for modelling of process warehouse.

(Böhnlein and Ulbrich-vom Ende, 2000): *Business Process Oriented development of data warehouse structures*- Business process oriented development of data warehouse is focused on deriving data warehouse structures from business process models. DW design is based on the informational requirements of the users, and then these requirements have to be compared with the actual information offered by the operational data sources. Finally, the initial data warehouse structures is derived from a conceptual object schema where main processes are marked off. Some of the identified concepts are, e. g., metrics (e. g., nr of enrolled students), dimensions (e. g., enrollment dim with attributes: date, semester and year of enrollment), dimensions hierarchies (semester->year) and constraints (aggregation functions along dimension hierarchies) A 'dimensional fact model' is used for conceptual modelling process warehouse.

(Jarke et al., 2000): *The Challenge of Process Data Warehousing*- This approach discuss an enterprise at the conceptual level of DW is split into a set of loosely connected partial models, which look at different facets of a chemical engineering process. To achieve the coherence between these partial models a process flowsheet is presented which represents an abstraction of the plant structure; and at the logical and physical level, the process engineering tools are presented that are more heterogeneous than traditional OLTP data sources. According to this approach, a knowledge-based metadata repository is proposed which is supported by selected materialized instance data for recording this heterogeneous engineering process. However, a conceptual model of PW is not presented.

4.2.3 Multidimensional process mining approaches

During our survey, we have found a number of relational and multidimensional data warehouse design used for process mining analysis. A brief overview of the state-of-art of the presented approaches is discussed as follows:

(Vogelgesang and Appelrath, 2015): *A Relational Data Warehouse for Multidimensional Process Mining*- In this approach, the concepts of PMCube, a data warehouse-based approach for multidimensional process mining is proposed. Here, attributes of an event log (e. g.,case, activity or timestamp) are used to derive dimensions (normalized ones) of the cube constrained to a specific domain. Each combination of dimension values forms a cell of the cube that contains a subset of the event log (sublog) related to these dimension values. Furthermore, they partition event logs into groups of cases called sublogs with homogeneous features in a dynamic and flexible way. For each sublog, a separated process model is discovered (using one of the miner algorithms, e. g., heuristic or genetic miner) and compared to other models to identify group-specific differences for the process. They introduce generic query patterns which map OLAP queries to SQL to push the operations (i. e., aggregation and filtering) to the database management system (DBMS) providing acceptable loading times for multidimensional event data. Entity-relationship data model is used as a conceptual data model for representing multidimensional event logs.

(Bolt and van der Aalst, 2015): *Multidimensional Process Mining Using Process Cubes*- The approach formalizes the process cubes notion where the event data organized using different dimensions. Each cell in the cube corresponds to a set of events which can be used as an input by any process mining technique (refer to section 6.2.2). In this approach OLAP data cube is adapted to accommodate event data through multidimensional process mining. This adaptation is far from trivial given the nature of event data which cannot be easily summarized or aggregated, conflicting with classical OLAP assumptions. For example, multidimensional process mining can be used to analyse the different versions of a sales processes, where each version can be defined according to different dimensions such as location or time, and then the different results can be compared (visualized as a 2-D grid). This analysis results may provide valuable insights for process optimization. Dimension, dimension attributes and value sets, hierarchy defined for each dimension and cells that contains event values are defined, eventhough no conceptual data model is presented.

(Mamaliga, 2013): *Realizing a Process Cube Allowing for the Comparison of Event Data*- In this approach, a framework is developed for the aim to support the construction of the process cube and allows multidimensional filtering on it, in order to separate subcubes for further processing. They design an hybrid database structure that combines a high-speed in-memory multidimensional database with a sparsity-immune relational database to overcome the sparsity problem (i. e., the missing data values at the intersection of dimensions) of the resulted process cube that stores event logs. A hierarchy level is defined only in the time dimension.
(Ribeiro and Weijters, 2011): *Event Cube: Another Perspective on Business Processes*- An event cube is presented in this approach, for process discovery and analysis. The data cube is formed by a a complete set of cuboids through a lattice of cuboids. A cuboid is

a set of cells that share the same dimension. Measures are represented through a set of aggregated values according to a given aggregation function. Additionally, it is said that a cuboid is materialized when all of its cells are materialized. The same principle applies to the data cube and its cuboids. By materializing a selection of dimensions of interest, an Event Cube can be built to accommodate all the necessary measurements to perform process mining (discovery of process models) and for the purpose of reporting after performing typical OLAP operations on the cube. Neither formalized concepts are given, nor a conceptual data model is presented.

(van der Aalst, 2013): *Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining*- According to this approach process cubes notion is presented to organize events and mined process models using different dimensions. Each cell in the process cube corresponds to a set of events and can be used to discover a process model, to check conformance with respect to some process model, or to discover bottlenecks. The idea is related to the well-known OLAP (Online Analytical Processing) data cubes and associated operations such as slice, dice, roll-up, and drill-down. However, there are also significant differences because of the process-related nature of event data. A case study of four variants of the same municipal complaints handling process is demonstrated. A process cube is built with three dimensions: *case type* (properties of the case the event belongs to, e. g.,"gold customer"), *event class* (properties of the individual event e. g., the event's activity name, its associated resource, or the geographic location associated with it) and *time window* (timestamps found in the event log, e. g., a particular day, month, or year).

4.3 Evaluation of process warehouse approaches

There are some studies such as (List et al., 2002a) with the scope of comparing different development methodologies for a process warehouse case study. This study is limited in classifying few approaches based on the methodology (i. e., data-driven, user-driven or goal-driven) used to develop a process warehouse and not about the proposed data model.

We use the work from (Shahzad and Johannesson, 2009) and (Vogelgesang et al., 2016) as complementary to refine the process analysis framework by adding new analysis parameters (e. g., variant analysis, process-pattern analysis) and new design issues (e. g., generalization/specialization relationship).

In tables 4.1 and 4.2 we give an evaluation of process warehouse approaches based on different analysis parameters categorized from various perspectives:

- *Activity analysis*: queries like executed activities, such as failed, successful, not executed, suspended activities etc.are answered. This parameter checks whether activities analysis can be done by a process warehouse or not.

- *Informational analysis*: queries like what information flows between elements of a process and which elements are involved in the flows are answered.

- *Subprocess analysis*: queries like process decomposition and subprocess collaboration is done in order to achieve a single goal are answered.

- *Control-flow analysis*: queries like which elements can be executed in series or in parallel etc. are answered.

- *Cycle-time analysis*: queries like the amount of time consumed by each process, start time and end time of a process are answered.

- *Process path analysis*: queries like which path is followed in parallel flows against an event are answered.

- *Deadline analysis*: queries like the number of times the deadlocks occurrences during process execution are answered.

- *Process-pattern analysis*: queries like process-patterns frequency, duration time or cost among different process variants are answered.

- *Resource analysis*: queries like which resources are considered for the enactment of a process are answered.

- *Organizational unit analysis*: queries like which processes are associated with which actor or role in an organizational unit are answered.

- *Participant analysis*: queries like number of participants associated with a process and the number of processes associated with a specific process are answered.

- *Software or service analysis*: queries like which software is associated with which process are answered.

- *Data input analysis*: queries like the amount of input required to trigger a process etc. are answered.

- *Consumption analysis*: queries like consumed time and the workload for each process resource are answered.

- *Data output analysis*: queries like the number of times a process was successfully executed etc. are answered.

- *Process variant analysis*: queries like aggregated measures over a group of similar activities among different variants are answered.

The summarized results from Table 4.1 and Table 4.2 are denoted by: ✓ if the analysis parameter is considered or supported (it either can be the focus of the approach or covered by the approach), **X** if the analysis parameter is not supported and **NoInfo** if the analysis parameter cannot be defined if it's present or not in the approach.
Whereas, in Table 4.3 and Table 4.4 we give an evaluation of process warehouse approaches based on different modelling concepts and some design issues considered. Approaches denoted by a *DFM* presented a Dimensional Fact Model as a data warehouse model notation or *ER* as a common modelling notation used or *ADAPT* (Application Design for Analytical Processing Technologies). Parameters classified in *Data modelling Concepts* such as *Dimension*, *Measure* etc.are denoted by ✓ if the concept parameter is considered or supported, **X** if the concept parameter is not supported and **NoInfo** if the concept parameter cannot be defined if it's present or not in the approach. The same for other parameters classified by *Design Challenges* group.

Table 4.1: Evaluation of PW approaches through different analysis parameters (a)

		(Niedrite et al., 2007)	(Mansmann et al., 2007a)	(Casati et al., 2007)	(Pau et al., 2007)	(Giorgini et al., 2005)	(Schiefer et al., 2003)	(Eder et al., 2002)	(Kueng et al., 2001)	(Jarke et al., 2000)	(Böhnlein and Ulbrich-vom Ende, 2000)
Functional Perspective	Activity analysis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Informational flow	✓	✓	✓	NoInfo	✓	✓	X	X	NoInfo	X
	Subprocess analysis	X	✓	X	X	✓	✓	✓	✓	✓	X
Behavior Perspective	Control-flow analysis	X	✓	✓	✓	✓	✓	✓	NoInfo	NoInfo	✓
	Cycle-time analysis	✓	✓	✓	✓	✓	✓	✓	✓	NoInfo	✓
	Path analysis	X	X	✓	X	NoInfo	✓	✓	X	NoInfo	X
	Deadline analysis	X	X	✓	✓	NoInfo	X	✓	✓	NoInfo	X
	Process-pattern analysis	X	X	✓	X	X	X	X	X	X	X
Organizational Perspective	Resource analysis	✓	✓	✓	X	✓	✓	X	X	✓	X
	Participant analysis	✓	✓	✓	✓	NoInfo	✓	✓	✓	✓	✓
	Organizational unit analysis	X	✓	NoInfo	NoInfo	✓	✓	✓	✓	NoInfo	X
	Software or service analysis	X	✓	NoInfo	X	NoInfo	X	✓	NoInfo	NoInfo	X
	Process variant analysis	X	X	X	X	X	X	X	X	X	X
Informational Perspective	Data Input analysis	✓	✓	✓	X	NoInfo	X	X	NoInfo	NoInfo	X
	Consumption analysis	X	✓	✓	✓	✓	X	X	X	NoInfo	X
	Data Output analysis	✓	✓	✓	✓	NoInfo	✓	✓	X	NoInfo	✓

From Table 4.1 and Table 4.2 results we can conclude that all of the approaches support the most of functional perspective of business processes. From behavioral perspective almost all approaches support the most of analysis parameter except for process pattern analysis which is considered only in (Berberi et al., 2018; Casati et al., 2007; Koncilia et al., 2015). Whereas *Process variant analysis* classified in organization perspective is supported only in our approach as this is the focus of our research. Other parameters of organizational perspective are considered mostly in our approach as well as (Benker, 2016; Shahzad and Zdravkovic, 2012; Vogelgesang and Appelrath, 2015; Van Der Aalst, 2013) and not considered at all in approaches such as (Kueng et al., 2001; Böhnlein and Ulbrich-vom Ende, 2000; Koncilia et al., 2015). Informational perspective mostly is considered in (Casati et al., 2007; Mansmann et al., 2007b; Pau et al., 2007) whereas data objects as stated by (Benker, 2016) are left out from recent approaches because are not precise enough as in BPMN there is no possibility to specify these latter structures like UML Class Diagrams or Entity Relationship Models.

Table 4.2: Evaluation of PW approaches through different analysis parameters (b)

		(Berberi et al., 2018)	(Benker, 2016)	(Koncilia et al., 2015)	(Shahzad and Zdravkovic, 2012)	(Vogelgesang and Appelrath, 2015)	(Bolt and van der Aalst, 2015)	(van der Aalst, 2013)	(Mamaliga, 2013)	(Ribeiro and Weijters, 2011)	(Neumuth et al., 2008)
Functional Perspective	Activity analysis	✓	✓	NoInfo	✓	X	✓	✓	✓	X	✓
	Informational flow	✓	✓	NoInfo	✓	X	✓	✓	✓	X	✓
	Subprocess analysis	✓	✓	X	✓	X	X	X	X	X	NoInfo
Behavior Perspective	Control-flow analysis	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Cycle-time analysis	✓	X	✓	✓	✓	✓	✓	✓	X	✓
	Path analysis	✓	X	✓	✓	✓	✓	✓	✓	✓	X
	Deadline analysis	X	X	X	X	X	X	X	X	X	X
	Process-pattern analysis	✓	X	✓	X	X	X	X	X	X	X
Organizational Perspective	Resource analysis	✓	✓	X	✓	✓	X	✓	✓	✓	✓
	Participant analysis	✓	✓	X	✓	✓	X	✓	✓	X	✓
	Organizational unit analysis	✓	✓	X	✓	X	✓	✓	✓	X	X
	Software or service analysis	X	X	X	✓	X	X	X	X	X	X
	Process variant analysis	✓	X	X	X	X	X	X	X	X	X
Informational Perspective	Data Input analysis	X	X	X	X	X	X	X	X	X	X
	Consumption analysis	X	X	X	X	X	X	X	X	X	X
	Data Output analysis	X	X	X	X	X	X	X	X	X	X

From Table 4.3 and Table 4.4 results we can conclude that most of the approaches used the common modelling notation- *ER* to design PW schema. Only some approaches such as (Mansmann et al., 2007b; Neumuth et al., 2008; Giorgini et al., 2005) proposed a dimensional fact model for their PW schema, whereas (Eder et al., 2002; Kueng et al., 2001) used "ADAPT" as a modelling notation. All the approaches discussed about basic concepts in data modelling. Only a few approaches from Table 4.3 such as (Eder et al., 2002; Mansmann et al., 2007b; Pau et al., 2007) considered only one or two design challenges on building their PW. Whereas, many approaches from Table 4.4 considered most of the design challenges during the design of PW, except for Generalization-specialization relationship that is supported only in our method.

Table 4.3: Evaluation of PW approaches based on different concepts modelling and design issues (a)

	(Niedrite et al., 2007)	(Mansmann et al., 2007a)	(Casati et al., 2007)	(Pau et al., 2007)	(Giorgini et al., 2005)	(Schiefer et al., 2003)	(Eder et al., 2002)	(Kueng et al., 2001)	(Jarke et al., 2000)	(Böhnlein and Ulbrich-vom Ende, 2000)
Data modelling Notations	ER	DFM	ER	ER	DFM	ER	ADAPT	ADAPT	ER	ER
							Data modelling Concepts			
Dimension	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dimension Attribute	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fact	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Aggregation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Measure	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
							Design Challenges			
Heterogeneity of fact entries	X	✓	✓	✓	X	NoInfo	NoInfo	NoInfo	X	X
Conceptually complex aggregations	X	✓	NoInfo	✓	X	NoInfo	NoInfo	NoInfo	X	X
Many-to-many relationship between fact and dimension (M:M)	X	✓	NoInfo	NoInfo	X	NoInfo	NoInfo	NoInfo	X	X
Generalization-specialization relationship	X	X	X	X	X	X	X	X	X	X
Integration of business process context perspective	X	✓	NoInfo	NoInfo	X	✓	NoInfo	NoInfo	X	X
Interchangeability of fact/measure and dimension roles	X	NoInfo	NoInfo	✓	X	NoInfo	✓	NoInfo	X	X
Diversity and evolution management	X	NoInfo	NoInfo	NoInfo	X	NoInfo	NoInfo	NoInfo	X	X

Table 4.4: Evaluation of PW approaches based on different concepts modelling and design issues (b)

	(Berberi et al., 2018)	(Benker, 2016)	(Koncilia et al., 2015)	(Shahzad and Zdravkovic, 2012)	(Vogelgesang and Appelrath, 2015)	(Bolt and van der Aalst, 2015)	(van der Aalst, 2013)	(Mamaliga, 2013)	(Ribeiro and Weijters, 2011)	(Neumuth et al., 2008)
Data modelling Notations	ER	ER	NoInfo	DFM	ER	NoInfo	NoInfo	NoInfo	NoInfo	DFM
Data modelling Concepts										
Dimension	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Dimension Attribute	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fact	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Aggregation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Measure	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Design Challenges										
Heterogeneity of fact entries	✓	✓	✓	✓	✓	✓	✓	✓	X	X
Conceptually complex aggregations	✓	✓	✓	✓	✓	✓	✓	✓	X	X
Many-to-many relationship between fact and dimension (M:M)	✓	NoInfo	NoInfo	NoInfo	NoInfo	NoInfo	NoInfo	NoInfo	X	X
Generalization-specialization relationship	✓	X	NoInfo	X	X	X	X	X	X	X
Integration of business process context perspective	✓	✓	NoInfo	✓	✓	✓	✓	✓	✓	X
Interchangeability of fact/measure and dimension roles	✓	✓	X	✓	NoInfo	NoInfo	NoInfo	NoInfo	X	X
Diversity and evolution management	✓	✓	X	✓	✓	✓	✓	✓	X	X

Based on the the analysis framework of business processes using a PW approach, our study reflects the following:

(a) it is possible to evaluate process warehouse's capability,

(b) during the design of a process warehouse many of the challenges are not addressed,

(c) any of the studied approaches does not consider process variant analysis and

(d) still there is a need of a process warehouse that entirely supports the analysis of processes from all the perspectives in a business process domain

4.4 Summary and discussions

Finally, section 4.4 summarizes and discusses. In Figure 4.1 we show a categorization of current process modelling approaches as discussed in details in Section 4.2.2. We categorize them in two groups: first group: a process warehouse is designed based on a given process model and second group: a process warehouse is designed based on a mined process model (i. e., discovered from event log data). In the first group are partitioned fifteen approaches whereas in the other only five as displayed by each text box. In next chapter we present our process warehouse model to analyse process variants.

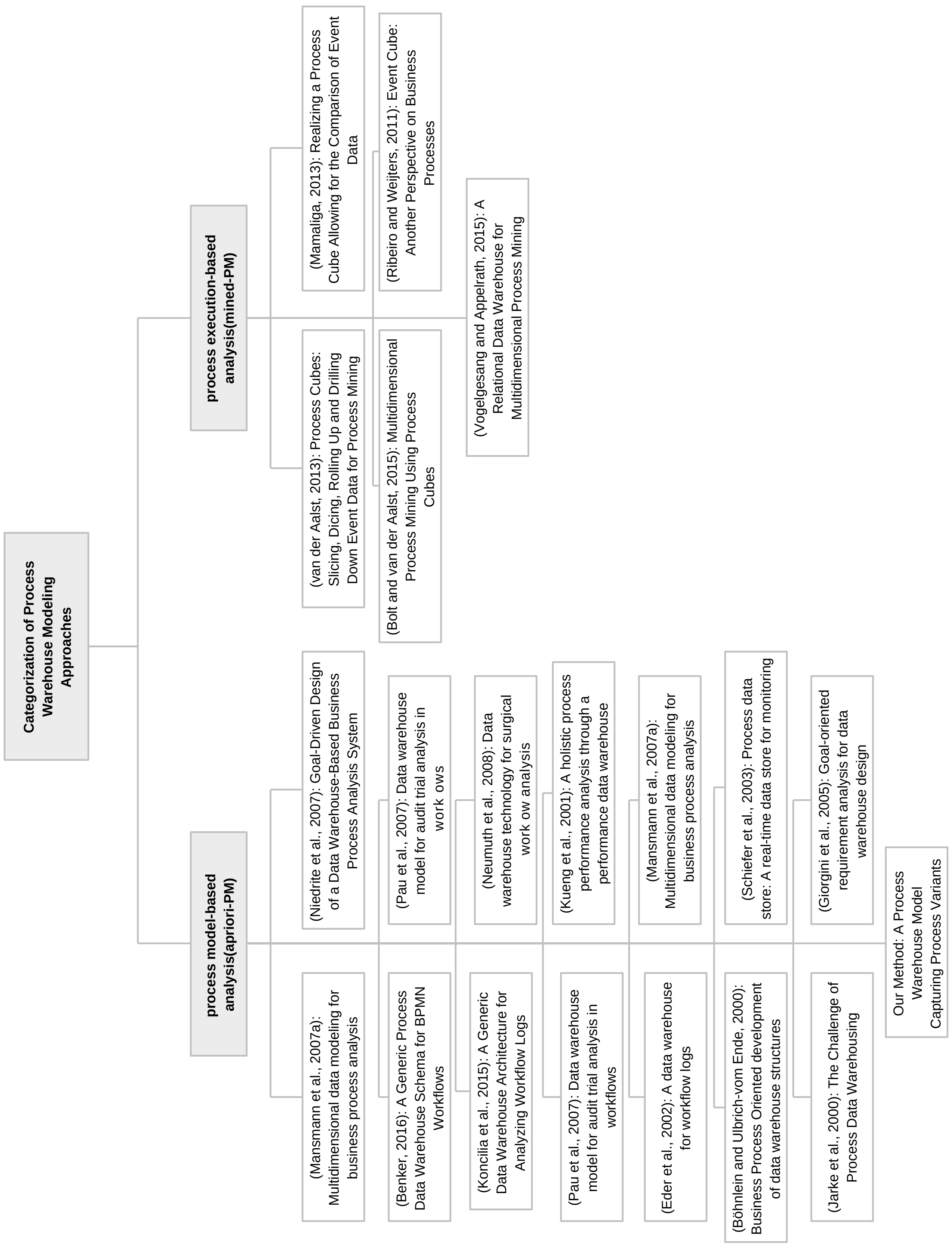

Categorization of Process Warehouse Modeling Approaches
process model-based analysis(apriori-PM)
process execution-based analysis(mined-PM)
(Mansmann et al., 2007a): Multidimensional data modeling for business process analysis
(Niedrite et al., 2007): Goal-Driven Design of a Data Warehouse-Based Business Process Analysis System
(Benker, 2016): A Generic Process Data Warehouse Schema for BPMN Workflows
(Pau et al., 2007): Data warehouse model for audit trial analysis in work ows
(Koncilia et al., 2015): A Generic Data Warehouse Architecture for Analyzing Workflow Logs
(Neumuth et al., 2008): Data warehouse technology for surgical work ow analysis
(Pau et al., 2007): Data warehouse model for audit trial analysis in workflows
(Kueng et al., 2001): A holistic process performance analysis through a performance data warehouse
(Eder et al., 2002): A data warehouse for workflow logs
(Mansmann et al., 2007a): Multidimensional data modeling for business process analysis
(Böhnlein and Ulbrich-vom Ende, 2000): Business Process Oriented development of data warehouse structures
(Schiefer et al., 2003): Process data store: A real-time data store for monitoring
(Jarke et al., 2000): The Challenge of Process Data Warehousing
(Giorgini et al., 2005): Goal-oriented requirement analysis for data warehouse design
Our Method: A Process Warehouse Model Capturing Process Variants
(van der Aalst, 2013): Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining
(Mamaliga, 2013): Realizing a Process Cube Allowing for the Comparison of Event Data
(Bolt and van der Aalst, 2015): Multidimensional Process Mining Using Process Cubes
(Ribeiro and Weijters, 2011): Event Cube: Another Perspective on Business Processes
(Vogelgesang and Appelrath, 2015): A Relational Data Warehouse for Multidimensional Process Mining

Figure 4.1: Categorization of Process Warehouse Modelling Approaches

CHAPTER 5

Process Warehouse for Process Variants Analysis

Chapter 5 presents a new generic process warehouse model for capturing adequately process variants. We introduce firstly the steps on how to design a process warehouse model based on Kimball recommendations. In sections 5.2.1 and 5.2.2 we describe in detail how we structure the hierarchies between processes and activities of different process variants and how we map respective logs to process dimension of the PW schema. Furthermore, we design a cube and deploy it to provide business users access.

5.1 Introduction

The design, construction and implementation of a data warehouse in general is a challenging task that should not be underestimated. In this section we introduce key concepts related to data warehousing for processes and discuss basic functionality associated with a multidimensional view of data. During design phase of a data warehouse ad-hoc queries should be specifically supported as conceptual requirements to anticipate the use of the DWH. They are called the bedrock of a successful data warehouse/business intelligence (DW/BI) system (Mundy et al., 2011). Usually, the process of gathering business requirements and converting them into a DW/BI system strategy deals with interviewing business analysts and IT representatives.

Clearly, a model behind the user query and the reporting experience must be well designed to support the full range of analyses. Accordingly, the multidimensional model is used as a paradigm for the representation of data in a data warehouse as it is fundamentally connected to its ease of use and intuitiveness for IT newbies (Golfarelli and Rizzi, 2009). Most of the productivity tools such as spreadsheets have adopted the multidimensional model as a visualization paradigm thus increasing its success rate. A dimensional model is made up of a central fact table (or tables) and its associated dimensions. The dimensional model is also called a star schema because it looks like a star with the fact table in the middle and the dimensions serving as the points on the star. Whereas, another common dimensional model is a snowflake as a variation of star schema where dimensions are normalized into multiple related tables.

The dimensional model consists of a normalized fact table with denormalized dimension tables from the relational data modeling perspective. Here, we define dimensional model three basic components as follows:

- *Facts*: Each fact table stores the measurements associated with a specific business scenario, like taking an order, paying an invoice or handling a customer support request. A record in a fact table is a measurement event. These events usually have numeric values that quantify the magnitude of the event, such as quantity ordered, sale amount, or call duration. These numbers are called facts or *measures*. An event in a sale business example stands for a specific item sold on a specific date at a specific store and can be described by a *quantity* sold measure. Whereas, an event in our application domain refers to an activity in a specific process instance.

  The primary key to the fact table is usually a multi-part key made up of a subset of the foreign keys from each dimension table involved in the scenario.

  Most facts are numeric and additive (such as orders amount or unit orders), meaning they can be summed up across all dimensions.
  Other facts are semi-additive (such as market share or account balance), and still others are non-additive (such as unit price). There are exceptions where some facts are non numeric like package weight of a product. In this case they contain discrete descriptive information. Such descriptive information is more naturally used for constraining a query, rather than being summed in a computation (Mundy et al., 2011). This distinction is helpful when deciding whether a data element is part of a dimension or fact. And finally, there are some facts derived or computed from other facts, e. g., a net sale number might be calculated from gross sales minus sales tax.

- *Grain* is defined as the level of detail contained in a fact table. It's recommended to build fact tables with the lowest level of detail that is possible from the original source —generally known as the *atomic* level. In so doing, the data may be rolled up to any level of summary needed across an dimension due to the flexibility assured of atomic fact tables. Each fact table is kept at a single grain. For example, it would be confusing and dangerous to have individual sales order line items in the same fact table as the monthly forecast.

- *Dimensions*: Dimensions are the nouns of the dimensional model, describing the objects that participate in the business, such as employee, customer, physician etc. Each dimension table joins to all the business scenarios in which it participates. For example, the product dimension participates in supplier orders, inventory, shipments, and returns business processes (Mundy et al., 2011).

Typical functionality of a data warehouse:

Data warehouses exist to facilitate complex, data-intensive, and frequent ad-hoc queries. Accordingly, data warehouses must provide far greater and more efficient query support than is demanded of transactional databases. The data warehouse access component supports enhanced spreadsheet functionality, efficient query processing, structured queries, ad-hoc queries, data mining, and materialized views (Mundy et al., 2011). In particular, enhanced spreadsheet functionality includes support for state-of-the art spreadsheet applications (for example, MS Excel) as well as for OLAP applications programs.

These offer preprogrammed functionalities such as the following:

- *Roll-up*: Data is summarized with increasing generalization (e. g., weekly to quarterly to annually).
- *Drill-down*: Increasing levels of detail are revealed (the complement of rollup).
- *Pivot*: Cross tabulation (also referred to as rotation) is performed.
- *Slice and dice*: Projection operations are performed on the dimensions.
- *Sorting*: Data is sorted by ordinal value.

5.2 A generic process warehouse model for process variants

We based our process warehouse model on (Kimball and Ross, 2013) recommendations to build a multi-dimensional model to database design which is composed of one table with a multipart key, called the fact table, and a set of smaller tables called dimension tables (star-schema and snowflake schema). Despite these two key decisions such as Identify dimensions and Identify facts other two preceding key decisions we should consider when designing a dimensional model are: "Select the business process and declare the grain".

As information requirements of PW we specify some queries we should answer that might be of interest based on previous user demands related to customer invoice payments process. A list of ad-hoc queries are as follows:

- Display the minimum execution time of patterns "$P_X, *$ " from all variants, where $P_X = C$ *or* D *or* E *or* F and $*-$ *all other activities in different process paths*, i. e., all processed payments after an order is processed either by an online shop or call-center?
- Display the cycle time of patterns "$P_Y, *$ " from all variants, where $P_Y = G$ *or* H i. e., all processed payments after customer receives his/her invoice?
- Display the average process duration of payments with patterns "$P_Z, *$ " performed by an organization unit, where $P_Z = O, P, Q, R$, i. e., of different payment options?
- Display the cycle time of patterns "$P_X, *$ " for each process variant, where $P_X = C$ *or* D *or* E *or* F ?

Whereas, for the information supply of PW we build on the process meta-model presented in Section 3.3. We introduce now the design of PW schema as shown in Figure 5.1). Furthermore, we build a cube that will allow us to analyse process variants and answer all the queries formulated above. It consists of the following six dimensions:

- Process
- Activity
- Process_Variant
- GeoLocation
- Participant
- Time
- Date

Process warehouses typically feature the dimensions process/activity, organization (department hierarchy), actor, geolocation and time. We use a graphical representation based on the ER model notation to design our PW schema as depicted in Figure 5.1. The process dimension usually only represents the part-of decomposition of processes. With the introduction of generic activities as discussed in the preceding section we are able to also provide consolidation hierarchies for process variants. The PW schema is derived based on the process meta model.
A *Process* dimension will be derived with all concrete processes of all variants as members and a corresponding generic process. Dimension *Activity* with activities uniquely identified to which concrete process they belong. To distinguish between multiple appearances of the same activity (activity label matches) in different variants a combination with the id of concrete process is used. We separate these two dimensions for the purpose of consolidating activity instances (occurrences) to elements which represents a combination of activities and concrete processes. In so doing, we can apply different measures in the level of instances of processes or instances of activities.
Then to consolidate these elements either to process or to activities we relate corresponding attribute to another dimension(as a bridge dimension) named *Process_ Variant.* This dimension is derived from the meta-model after performing steps of the algorithm Algorithm 4.

It stores activities that are consolidated to a generic activity as abstracted activities between process variants. Here, the same activity among different variants is combined to different generic activities and one generic activity can have many activities. So, this many-many relationship between members of this dimension can be resolved by using two separate dimension in a 1-* association. This approach allows to drill-down to the instances starting either from generic activity to activity or from generic activity to concrete process. Additionally, *Process_ Variant* dimension stores information about which are the activities(concrete and generic) of different variants as specializations of generic process(es).

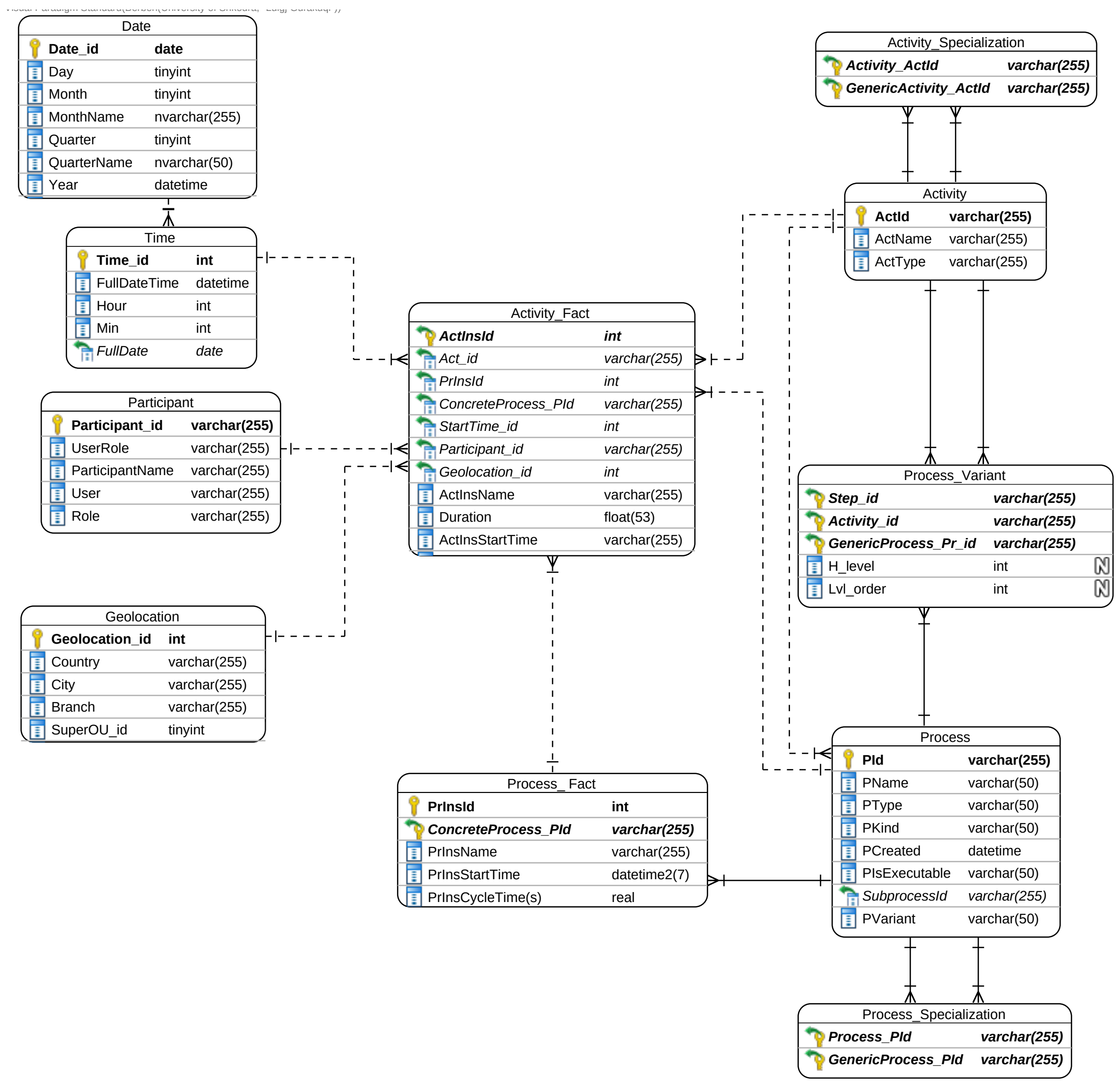


Figure 5.1: The process warehouse fact constellation schema

Some other typical dimensions from resource or organizational perspectives might be *Participant* (alternatively named agent) and *Geolocation*. We decided to separate these two dimensions to express the fact that different users can belong to different departments in different processes.

Participant stores information about users (human or system) with specific role, e. g., a call center agent, that is responsible for the execution of activities. *Geolocation* stores information about the geographic locations of a branch structured in different organization units (departments) where a user belongs.

Time as a business intelligent dimension is another important dimension for our process warehouse that stores information about time needed to execute an activity or process etc. Here we decide to model Time and Date calendar in two separate dimensions named *Time* and *Date*. For activity instances we have the information given in seconds (time units) when they are executed, whereas if we refer to process instances user is usually interested to give this info on hours or days. We design the process warehouse schema as a fact constellation schema with two fact tables, i. e. *Activity_ Fact* and *Process_ Fact* sharing respective dimension tables. The main idea is the possibility for business users to perform analysis based either on different process instances or on different activity instances.

5.2.1 Process warehouse dimension hierarchies

Hierarchical structures (tree structures) are used to create and manage a large number of alternate categories within a dimension in order to create drill down paths. Hierarchy consolidation paths from the highest level to the lowest one are defined for every dimension table, to show the relationship between their relative attributes. At the lowest level of *Process_ Variant* dimension is *step_ Id* which refers to attribute named *Act_ Id* within the *Activity* dimension table. To consolidate concrete activities to generic activities of a specific hierarchy level we define a hierarchy as shown in Figure 5.2. Hierarchy named *Activity2GA* express the fact that an activity is rolled-up to a generic activity that belongs to a generic process at a specific hierarchy level.

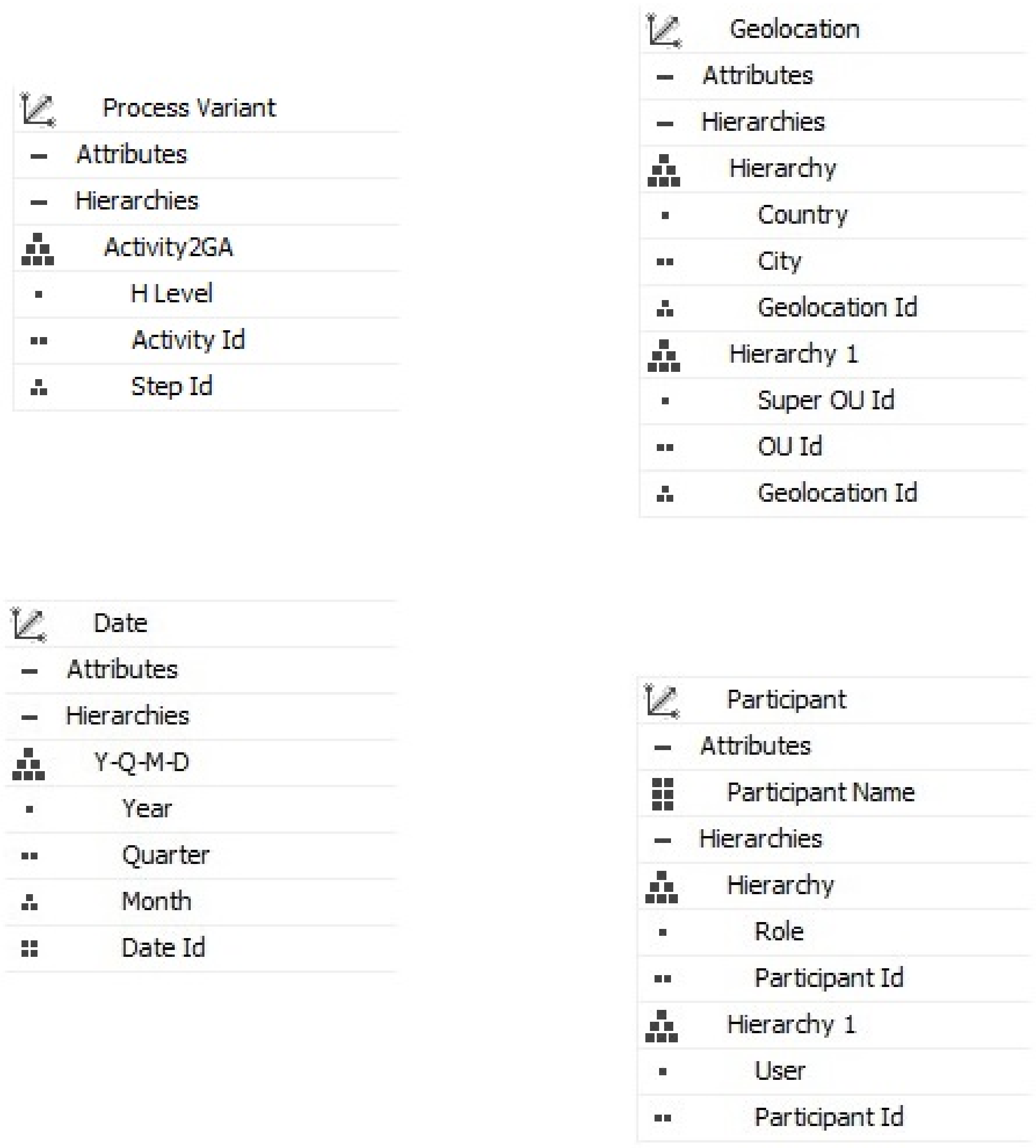
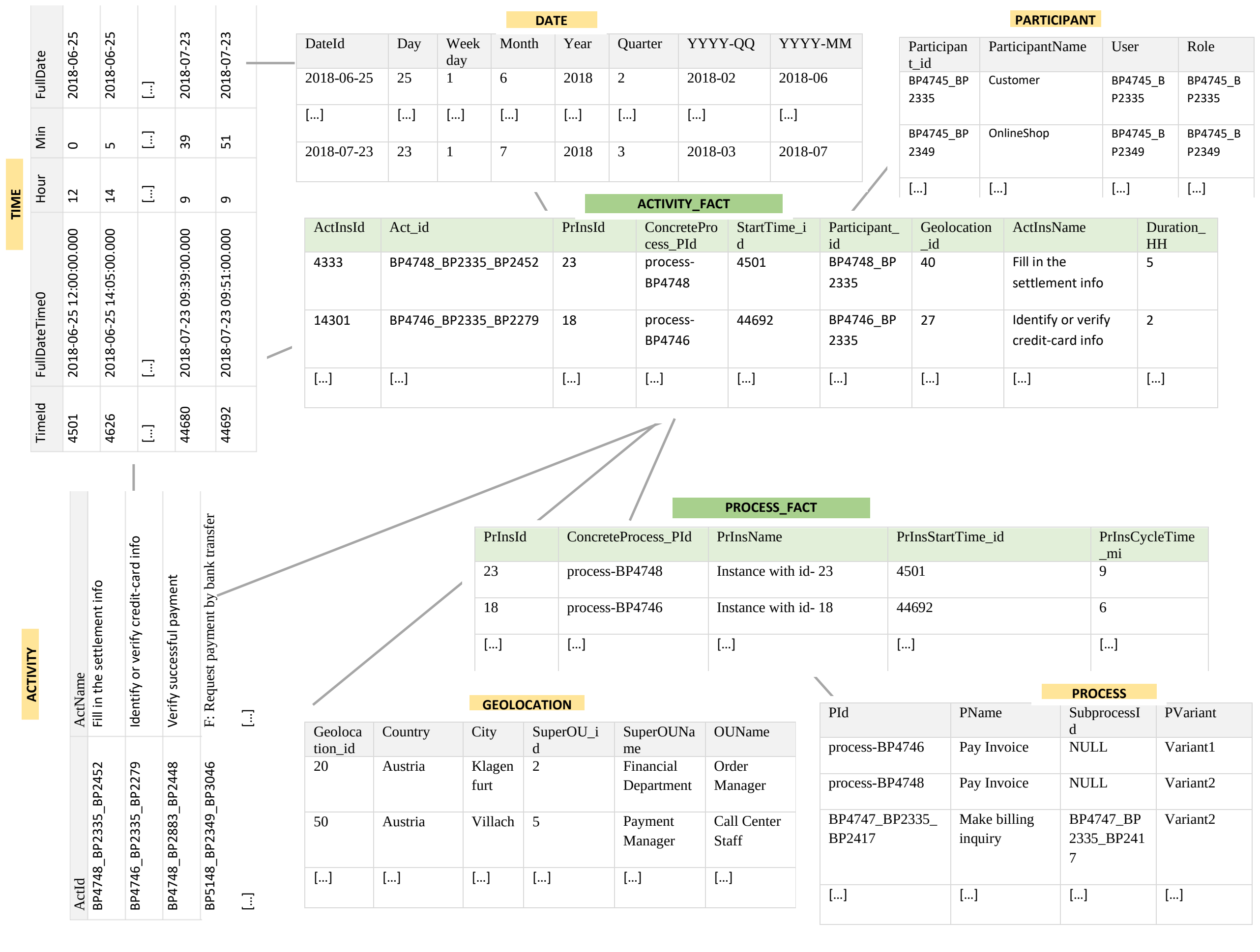


Figure 5.2: Hierarchy defined for some dimension of PW cube



TIME

TimeId	FullDateTime0	Hour	Min	FullDate
4501	2018-06-25 12:00:00.000	12	0	2018-06-25
4626	2018-06-25 14:05:00.000	14	5	2018-06-25
[...]	[...]	[...]	[...]	[...]
44680	2018-07-23 09:39:00.000	9	39	2018-07-23
44692	2018-07-23 09:51:00.000	9	51	2018-07-23

DATE

DateId	Day	Week day	Month	Year	Quarter	YYYY-QQ	YYYY-MM
2018-06-25	25	1	6	2018	2	2018-02	2018-06
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]
2018-07-23	23	1	7	2018	3	2018-03	2018-07

PARTICIPANT

Participant_id	ParticipantName	User	Role
BP4745_BP2335	Customer	BP4745_BP2335	BP4745_BP2335
BP4745_BP2349	OnlineShop	BP4745_BP2349	BP4745_BP2349
[...]	[...]	[...]	[...]

ACTIVITY_FACT

ActInsId	Act_id	PrInsId	ConcreteProcess_PId	StartTime_id	Participant_id	Geolocation_id	ActInsName	Duration_HH
4333	BP4748_BP2335_BP2452	23	process-BP4748	4501	BP4748_BP2335	40	Fill in the settlement info	5
14301	BP4746_BP2335_BP2279	18	process-BP4746	44692	BP4746_BP2335	27	Identify or verify credit-card info	2
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]

ACTIVITY

ActId	ActName
BP4748_BP2335_BP2452	Fill in the settlement info
BP4746_BP2335_BP2279	Identify or verify credit-card info
BP4748_BP2883_BP2448	Verify successful payment
BP5148_BP2349_BP3046	F: Request payment by bank transfer
[...]	[...]

PROCESS_FACT

PrInsId	ConcreteProcess_PId	PrInsName	PrInsStartTime_id	PrInsCycleTime_mi
23	process-BP4748	Instance with id- 23	4501	9
18	process-BP4746	Instance with id- 18	44692	6
[...]	[...]	[...]	[...]	[...]

GEOLOCATION

Geolocation_id	Country	City	SuperOU_id	SuperOUName	OUName
20	Austria	Klagenfurt	2	Financial Department	Order Manager
50	Austria	Villach	5	Payment Manager	Call Center Staff
[...]	[...]	[...]	[...]	[...]	[...]

PROCESS

PId	PName	SubprocessId	PVariant
process-BP4746	Pay Invoice	NULL	Variant1
process-BP4748	Pay Invoice	NULL	Variant2
BP4747_BP2335_BP2417	Make billing inquiry	BP4747_BP2335_BP2417	Variant2
[...]	[...]	[...]	[...]

Figure 5.3: Instances of the process warehouse for the payment orders application

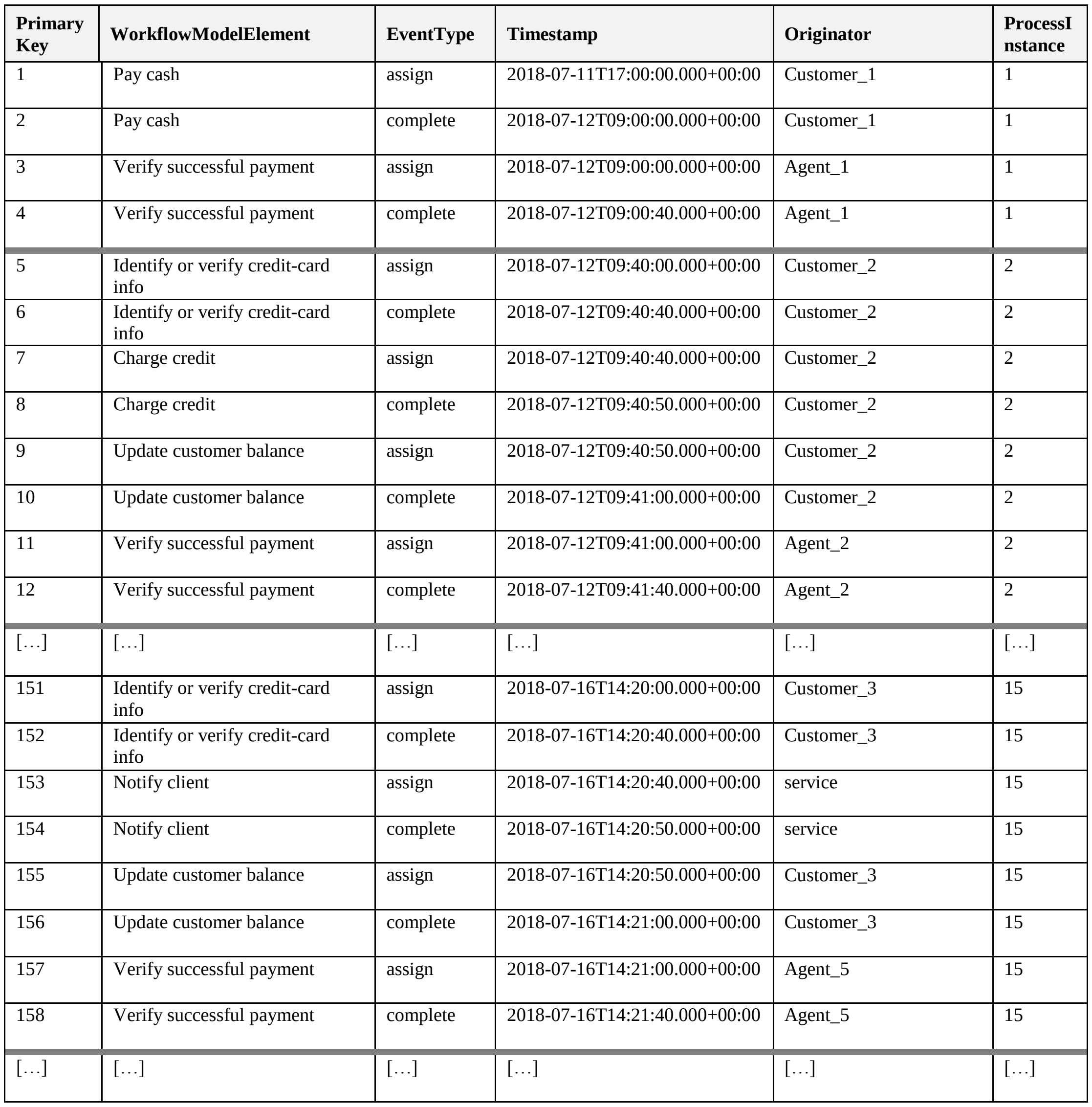
Date dimension has one hierarchy named *Y-Q-M-D* hierarchy with a specific consolidation path e. g., a day is rolled-up to a month, a month to a quarter and a quarter to a year, as shown in Figure 5.2. The *Participant* dimension with the lowest level of the participant him/herself is consolidated to a combination of users and roles, expressing the fact that a user can have a specific role in an organization unit. The *Geolocation* dimension with two defined hierarchies with the lowest level of *Geolocation_ Id*. This attribute can be consolidated to an organization uni t*(OU_ Id)* and an OU to a super organization unit *(superOU_ Id)* , e. g., Financial Department can have a higher hierarchy Management Department. In the other hierarchy we express the fact that a *Geolocation_ Id* can be consolidated to a *City* and a *City* to a *Country*, for instance *Geolocation_ Id=4* is consolidated to *City='Klagenfurt'* located in *Country='Austria'*.

5.2.2 Mapping process variants logs to process dimension

The variant LOGS record information about events ordered sequentially (a timestamp is given). Each event refers to an activity (WorkflowModelElement attribute) and a process instance. To match these events to an activity instance we combine each event by each process instance. And having the information that a process instance refers to which concrete process we can match all these activity instance to concrete processes as well(read the script below). Moreover, we combine each activity instance to which activity of concrete process it refers by matching activity labels. We already discussed and showed an excerpt of these step_activity instances in Table 3.2. Additional information may be recorded to events logs, e. g., the user (individual) which executes an activity and cost. In Figure 5.4 we give an excerpt of an event log of process *PayInvoice* of variant 5.

For more details we give the following *SQL* script as follows to express the mapping of the event logs (AuditTrailEntry temporary table) to our database table named *ProcessInstance*:

```sql
INSERT INTO [dB_Variants_Logs].dbo.ProcessInstance (PrInsId,PrInsName,
                                                     PrInsStartTime,
                                                     PrInsEndTime,
                                                     PrInsCycleTime,
                                        ConcreteProcess_Process_PId)

SELECT FK_ProcessInstance,'Instance with id- ' +
       CAST(FK_ProcessInstance as varchar),
       CAST(MIN([Timestamp]) as datetime2(7)),
       CAST(MAX([Timestamp]) as datetime2(7)),
       DATEDIFF(SS,CAST(MIN([Timestamp]) as datetime2(7)),
                   CAST(MAX([Timestamp]) as datetime2(7))),
       @pid
FROM #AuditTrailEntry
GROUP BY FK_ProcessInstance
ORDER BY FK_ProcessInstance
```



Primary Key	WorkflowModelElement	EventType	Timestamp	Originator	ProcessInstance
1	Pay cash	assign	2018-07-11T17:00:00.000+00:00	Customer_1	1
2	Pay cash	complete	2018-07-12T09:00:00.000+00:00	Customer_1	1
3	Verify successful payment	assign	2018-07-12T09:00:00.000+00:00	Agent_1	1
4	Verify successful payment	complete	2018-07-12T09:00:40.000+00:00	Agent_1	1
5	Identify or verify credit-card info	assign	2018-07-12T09:40:00.000+00:00	Customer_2	2
6	Identify or verify credit-card info	complete	2018-07-12T09:40:40.000+00:00	Customer_2	2
7	Charge credit	assign	2018-07-12T09:40:40.000+00:00	Customer_2	2
8	Charge credit	complete	2018-07-12T09:40:50.000+00:00	Customer_2	2
9	Update customer balance	assign	2018-07-12T09:40:50.000+00:00	Customer_2	2
10	Update customer balance	complete	2018-07-12T09:41:00.000+00:00	Customer_2	2
11	Verify successful payment	assign	2018-07-12T09:41:00.000+00:00	Agent_2	2
12	Verify successful payment	complete	2018-07-12T09:41:40.000+00:00	Agent_2	2
[…]	[…]	[…]	[…]	[…]	[…]
151	Identify or verify credit-card info	assign	2018-07-16T14:20:00.000+00:00	Customer_3	15
152	Identify or verify credit-card info	complete	2018-07-16T14:20:40.000+00:00	Customer_3	15
153	Notify client	assign	2018-07-16T14:20:40.000+00:00	service	15
154	Notify client	complete	2018-07-16T14:20:50.000+00:00	service	15
155	Update customer balance	assign	2018-07-16T14:20:50.000+00:00	Customer_3	15
156	Update customer balance	complete	2018-07-16T14:21:00.000+00:00	Customer_3	15
157	Verify successful payment	assign	2018-07-16T14:21:00.000+00:00	Agent_5	15
158	Verify successful payment	complete	2018-07-16T14:21:40.000+00:00	Agent_5	15
[…]	[…]	[…]	[…]	[…]	[…]

Figure 5.4: An example of an event log of a concrete process

The result of the execution after *insert into* we show in the following table:

Table 5.1: Process instance records in dB

PrInsId	PrInsName	PrInsStartTime	PrInsEndTime	PrInsCycleTime (s)	ProcessId
[..]	[..]	[..]	[..]	[..]	[..]
1	Instance with id- 1	2018-07-05 09:00:00.000	2018-07-05 09:05:30.000	330	process-BP4755
2	Instance with id- 2	2018-07-05 09:25:00.000	2018-07-05 09:30:30.000	330	process-BP4755
3	Instance with id- 3	2018-07-05 09:50:00.000	2018-07-05 09:53:16.000	196	process-BP4755
4	Instance with id- 4	2018-07-05 10:15:00.000	2018-07-05 10:20:30.000	330	process-BP4755
5	Instance with id- 5	2018-07-05 10:40:00.000	2018-07-05 10:43:01.000	181	process-BP4755
[..]	[..]	[..]	[..]	[..]	[..]
1	Instance with id- 1	2018-07-02 09:00:00.000	2018-07-02 09:22:41.000	1361	process-BP4753
2	Instance with id- 2	2018-07-02 09:30:00.000	2018-07-02 09:52:41.000	1361	process-BP4753
3	Instance with id- 3	2018-07-02 10:00:00.000	2018-07-02 10:23:30.000	1410	process-BP4753
4	Instance with id- 4	2018-07-02 10:30:00.000	2018-07-02 10:52:41.000	1361	process-BP4753
5	Instance with id- 5	2018-07-02 11:00:00.000	2018-07-02 11:22:41.000	1361	process-BP4753
[..]	[..]	[..]	[..]	[..]	[..]

5.3 Summary and discussion

In this section we summarize the most important aspects of Chapter 5 as one of the key chapters of this research work. We represented our generic process warehouse model to analyse process variants. To build the PW we specified some queries (based on the processing invoice payment running example) that should be answered as information requirements whereas process meta-model with generic activities defined is used as information supply. We demonstrated how we derived process dimension as one of the most important dimension in analysing process with variants. We showed how hierarchy consolidation paths from the highest level to the lowest one were defined for every dimension table, to show the relationship between their relative attributes. And last we presented how we map each variant log to this process dimension.
In the next chapter we report on techniques proposed to manage such large process model collections in the process mining domain.

CHAPTER 6

Process Mining for Process Variants

Chapter 6 gives a background on the discipline of process mining. Section 6.2 provides an overview of current process mining techniques to discover, monitor and analyse business process variants. In the following sections 6.2.2 we classify proposed *process mining techniques* suitable for discovering, monitoring and improving processes. Moreover, from subsections sections 6.2.3.1 to 6.2.3.3 we discuss techniques proposed for managing large process model collections.

6.1 Introduction

For more than a decade, process mining emerged as a new scientific discipline on the interface between process models and event data (van der Aalst, 2011). On the one hand, conventional Workflow Management (WfM)(van der Aalst and van Hee Kees, 2002) and Business Process Management (BPM)(Van Der Aalst, 2013) systems are mostly model-driven with little consideration for event data. On the other hand, Business Intelligence (BI), Data Mining (DM) and Machine Learning (ML) focus on data without considering end-to-end process models.
Process mining aims to bridge the gap between BPM and WfM on the one hand and DM, BI, and ML on the other hand. Process mining input is *event data* (e. g., an event log) (van der Aalst et al., 2012).
An event log is a collection of related events where each event refers to an activity (i. e., a well-defined step in a process) and is related to a particular case (i. e., a process instance). An ordered list of events belonging to a case is referred to as a "process run".
A variety of IT systems commonly available within organizations, such as Enterprise Resource Planning (ERP) systems, Database Management Systems (DBMSs) etc., are used to record event logs.

Information about organizational behaviour (resource initiating/executing the activity) available in event logs of today's organizations can be extracted to discover new process models using petri nets van Dongen et al. (2009) or enhance existing ones (Fahland and van der Aalst, 2012).

Additional information may be present in event logs e. g., the timestamp of the event, or data elements (input or output data) recorded with the event.

One of the key contributions of process mining is its ability to relate observed and modelled behaviour at the event level, i. e., , traces observed in reality (process instances in event log) are aligned with traces allowed by the model (complete runs of the model)(van der Aalst et al., 2014).

Even though, existing traditional process mining techniques assume processes to be in steady state, the process itself may be changing during its enactment (Bose et al., 2011). Today's organizations have to be flexible and adapt to changing circumstances. New law regulations are also forcing them to change their processes. The success of an organization is highly dependent on its ability to react to changes in its operating environment. Therefore, flexibility and change have been studied in-depth in the context of BPM where a set of change logs (i. e.,a set of process changes sequences performed on some initial process model) is produced from a sequence of change operations (i. e., alter a set of activities or change their order relations) imposed on a process model(Gunther et al., 2008).

Process Management Systems (PMS) frameworks like ADEPT2 (Reichert et al., 2005) support both ad-hoc changes of single process instances and the propagation of process type changes to running instances. Examples of ad-hoc changes are the insertion, deletion, movement, or replacement of activities. To exploit knowledge about process changes from change logs (Gunther et al., 2008) integrate process mining with adaptive PMS. The set of process changes results in multiple process variants due to the evolution of predefined process models.

6.2 Related work

6.2.1 Challenges of mining process variants

Mining process variants refers to discovering a (new) reference process model covering the given variant collection best (Li et al., 2011). By adopting the discovered model in the PAIS, future process adaptations and costs for change will decrease. But, finding such an improved reference model is far from trivial when considering control flow patterns like sequence, branching, conditional branching, parallel and loops. In the following we outline some issues, challenges and related scenarios (depicted in figure 6.1 of mining these process variants as discussed in (Li et al., 2011):

- *First scenario:* derive process variants by configuring a known reference process model. Here, issues to deal with concern the significant structural differences may result between old and new reference model if the new reference process model is discovered without considering the old one. As well as, the new reference process model may be "close" (fit) to the original reference model but not to the given variant collection or vice-versa.
  Therefore, process engineers should have the flexibility to control to what degree they want to maximally modify the original reference model to better fit to the given variant collection.

- *Second scenario:* discover a (new) reference process model by "merging" a collection of related process without any a-priori knowledge of the original reference process model these variants were derived from.

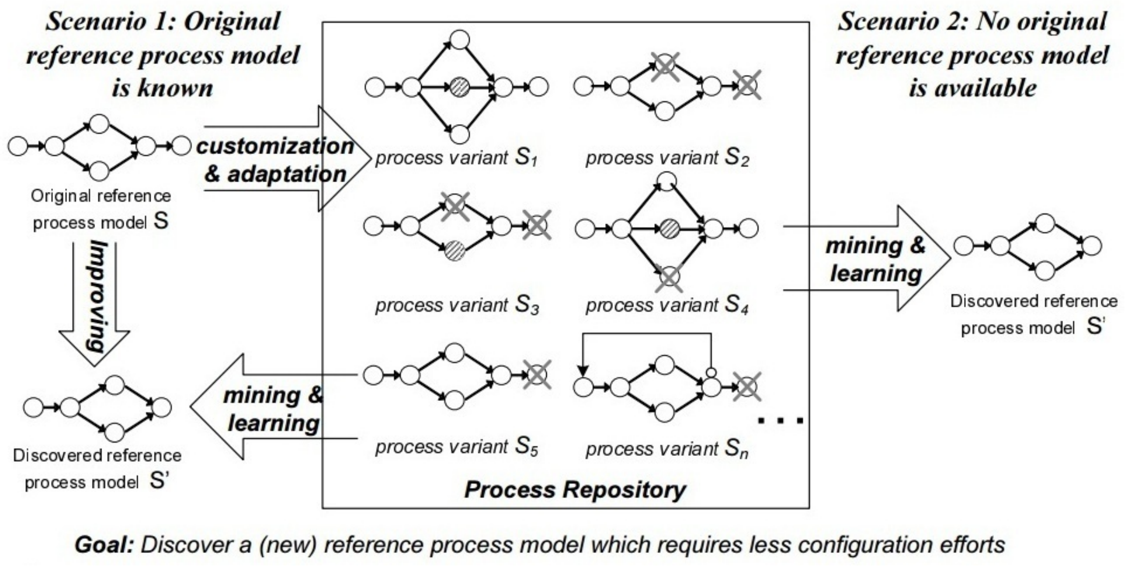

Scenario 1: Original reference process model is known
Original reference process model S
customization & adaptation
Improving
Discovered reference process model S'
mining & learning
process variant S1
process variant S2
process variant S3
process variant S4
process variant S5
process variant Sn
Process Repository
Scenario 2: No original reference process model is available
mining & learning
Discovered reference process model S'
Goal: Discover a (new) reference process model which requires less configuration efforts

Figure 6.1: Different scenarios for discovering reference process models (Li et al., 2011)

The major goal of mining process variants is to derive a generic process model out of a given collection of process variants. In so doing, different process variants can be efficiently

configured out of the generic model.
Authors in (Li et al., 2008) measure the efforts for respective process configurations by the number of change operations needed to transform the generic model into the respective model variant. The challenge is to find a generic model such that the average number of change operations needed (i. e., the average distance) becomes minimal. Different algorithm (clustering, heuristic etc.) have been proposed to discover (or evolving) reference process models as discussed in the next subsections.

6.2.2 Process mining techniques

Event logs can be used as input to conduct various types of process mining (van der Aalst et al., 2014), as is illustrated in Figure 6.2. We list three relevant techniques of process mining:

(i) The first type of process mining is *discovery*.
A discovery technique takes an event log and produces a model without using any a-priori information. Process discovery is the most prominent process mining technique. For many organizations it is surprising to see that existing techniques are indeed able to discover real processes merely based on example executions in event logs.

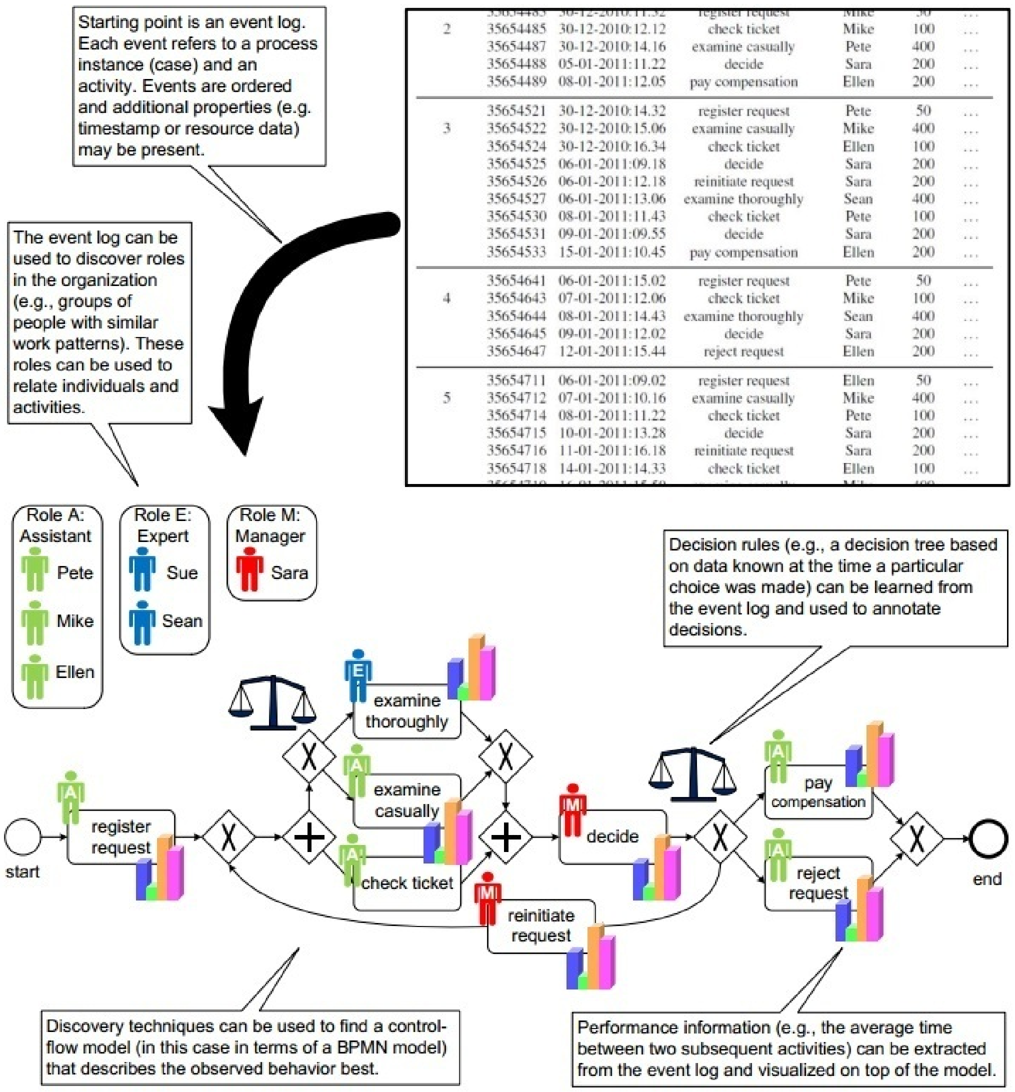
(ii) The second type of process mining is *conformance*.
Here, an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa.

(iii) The third type of process mining is *enhancement*.
Here, the idea is to extend or improve an existing process model using information about the actual process recorded in some event log. Whereas conformance checking measures the alignment between model and reality, this third type of process mining aims at changing or extending the a-priori model. For instance, by using timestamps in the event log one can extend the model to show bottlenecks, service levels, throughput times, and frequencies.

The first technique, *process discovery*, construct a process model from an event log, thus capturing the behaviour seen in the log. The naïve α-algorithm is introduced to understand the notion of process discovery (van der Aalst, 2011).
A process discovery algorithm is a function γ that maps an event log L specified in XES[1]$\in B(A^*)$(van der Aalst et al., 2012) onto a process model (a marked Petri net) $\gamma(L) = (N, M)$ such that the model is "representative" for the behaviour seen in the event log. Ideally, N is a sound WF-net and all traces in L correspond to possible firing sequences of (N, M).

The α -algorithm scans the event log for particular patterns. For example, if activity a is followed by b but b is never followed by a, then it is assumed that there is a causal dependency between a and b. To reflect this dependency, the corresponding Petri net

[1]See www.xes-standard.org for detailed information about the standard.format


Starting point is an event log. Each event refers to a process instance (case) and an activity. Events are ordered and additional properties (e.g. timestamp or resource data) may be present.
The event log can be used to discover roles in the organization (e.g., groups of people with similar work patterns). These roles can be used to relate individuals and activities.
Role A: Assistant
Pete
Mike
Ellen
Role E: Expert
Sue
Sean
Role M: Manager
Sara
Decision rules (e.g., a decision tree based on data known at the time a particular choice was made) can be learned from the event log and used to annotate decisions.
start
register request
examine thoroughly
examine casually
check ticket
decide
reinitiate request
pay compensation
reject request
end
A
E
M
Discovery techniques can be used to find a control-flow model (in this case in terms of a BPMN model) that describes the observed behavior best.
Performance information (e.g., the average time between two subsequent activities) can be extracted from the event log and visualized on top of the model.

2	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...
3	35654521	30-12-2010:14.32	register request	Pete	50	...
	35654522	30-12-2010:15.06	examine casually	Mike	400	...
	35654524	30-12-2010:16.34	check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	decide	Sara	200	...
	35654526	06-01-2011:12.18	reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	check ticket	Pete	100	...
	35654531	09-01-2011:09.55	decide	Sara	200	...
	35654533	15-01-2011:10.45	pay compensation	Ellen	200	...
4	35654641	06-01-2011:15.02	register request	Pete	50	...
	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
5	35654711	06-01-2011:09.02	register request	Ellen	50	...
	35654712	07-01-2011:10.16	examine casually	Mike	400	...
	35654714	08-01-2011:11.22	check ticket	Pete	100	...
	35654715	10-01-2011:13.28	decide	Sara	200	...
	35654716	11-01-2011:16.18	reinitiate request	Sara	200	...
	35654718	14-01-2011:14.33	check ticket	Ellen	100	...

Figure 6.2: Process mining techniques (van der Aalst et al., 2014)

should have a place connecting a to b. Four log-based ordering relations that aim to capture relevant patterns in the log are identified such as, $x \rightarrow_L y, y \rightarrow_L x, x\#Ly$, or $x\|_L y$, where $x, y \in \mathbb{A}$(set of activities) of an event log L. Function γ defines a so-called "Play-in" technique i. e., example behaviour is taken as input and the goal is to construct a model.

Let's consider a simple event log L_1:

$$L_1 = [\langle\ a,b,e,f\ \rangle^2, \langle\ a,b,e,c,d,b,f\ \rangle^3, \langle\ a,b,c,e,d,b,f\ \rangle^2, \langle\ a,b,c,d,e,b,f\ \rangle^4, \langle\ a,e,b,c,d,b,f\ \rangle^3]$$

An example of the computed process model from an event log L_1 is shown in figure 6.3. The 8 steps of the α -algorithm for $L = L_1$ are as follows:

(1) $T_L = \{a, b, c, d, e, f\}$ -check which activities appear in the log

(2) $T_I = \{a\}$ -set of start activities

(3) $T_O = \{f\}$ -set of end activities

(4) $X_L = \{(a,b),(a,e),(b,c),(b,f),(d,b),(e,f),(a,d,b),(b,c,f)\}$ -find places and their connections; the aim is to construct places named $p(A,B)$ such that A is the set of input transitions ($p_{(A,B)} = A$) and B is the set of output transitions ($p_{(A,B)} = B$) of $p_{(A,B)}$.

(5) all nonmaximal pairs are removed, thus yielding:

$$Y_L = \{(\{a\},\{e\}),(\{c\},\{d\}),(\{e\},\{f\}),(\{a,d\},\{b\}),(\{b\},\{c,f\})\}$$

(6) Every element of $(A,B) \in Y_L$ corresponds to a place $p(A,B)$ connecting transitions A to transitions B. In addition, P_L also contains a unique source place i_L and a unique sink place o_L.

$P_L = \{p_{(\{a\},\{e\})}, p_{(\{c\},\{d\})}, p_{(\{e\},\{f\})}, p_{(\{a,d\},\{b\})}, p_{(\{b\},\{c,f\})}, i_L, o_L\}$

(7) the arcs of the WF-net are generated. All start transitions in T_I have i_L as an input place and all end transitions T_O have o_L as output place. All places $p_{(A,B)}$ have A as input nodes and B as output nodes.

$$F_L = \{(a, p_{(\{a\},\{e\})}), (p_{(\{a\},\{e\})}, e), (c, p_{(\{c\},\{d\})}), (p_{(\{c\},\{d\})}, d),$$
$$(e, p_{(\{e\},\{f\})}), (p_{(\{e\},\{f\})}, f), (a, p_{(\{a,d\},\{b\})}), (d, p_{(\{a,d\},\{b\})}),$$
$$(p_{(\{a,d\},\{b\})}, b), (b, p_{(\{b\},\{c,f\})}), (p_{(\{b\},\{c,f\})}, c), (p_{(\{b\},\{c,f\})}, f),$$
$$(i_L, a), (f, o_L)\}$$

(8) The result is a Petri net $\alpha(L) = (P_L, T_L, F_L)$ that describes the behaviour seen in event log L

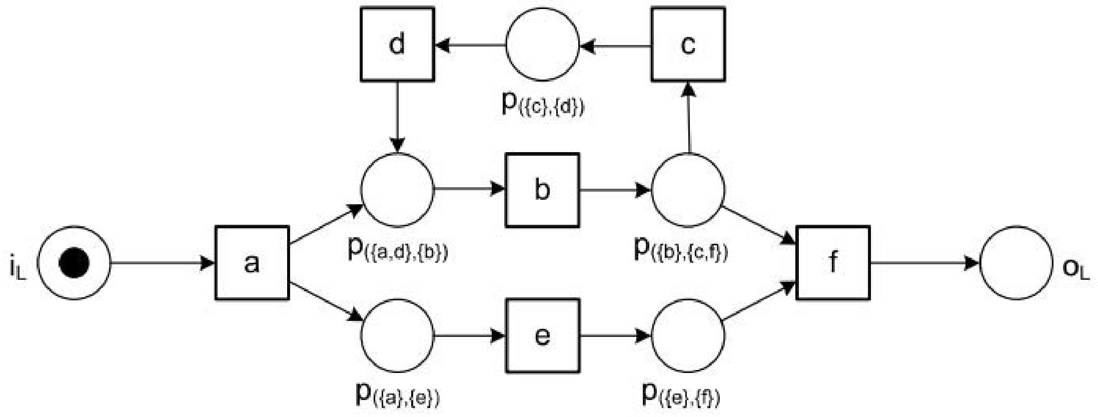


Figure 6.3: WF-net derived from L adapted from (van der Aalst et al., 2012)

Some limitations of this algorithm are:

- two process models structurally different may have the same possible behaviour (trace equivalent)
- dealing with loops, "short-loop" of length two. To deal with this there is a way to improve the basic α -algorithm, α^+ -algorithm as described in (de Medeiros et al., 2003).
- activities frequencies are not taken into account. Therefore, the algorithm is very sensitive to noise and incompleteness.

It is important to observe process discovery that is, by definition, restricted by the expressive power of the target language, i. e., the *representational bias*. To discuss and identify the representational bias of a language interested reader may refer to *workflow patterns*(van der Aalst et al., 2003).

The second technique, *conformance checking* measures the quality of a process model with respect to an event log. Conformance checking relates events in the event log to activities in the process model and compares both. Commonalities and discrepancies between the modelled behaviour and the observed behaviour are given. Conformance checking is relevant for business alignment and auditing. For example, the event log can be replayed on top of the process model to find undesirable deviations suggesting fraud or inefficiencies. Moreover, conformance checking techniques can also be used for measuring the performance of process discovery algorithms and to repair models that are not aligned well with reality (van der Aalst et al., 2012). The main idea of conformance checking is to compare the behaviour of a process model and the behaviour recorded in an event log for finding commonalities and discrepancies. Such analysis results in *global conformance measures*(e. g., 80 % of the cases in the event log can be replayed i. e., process model and log are used as an input, by the model) and *local diagnostics* (e. g., activity x was executed 10 times although this was not allowed according to the model). The interpretation of non-conformance depends on the purpose of the model. If the model is intended to be *descriptive*, then discrepancies between model and log indicate that the model needs to be improved to capture reality better. If the model is *normative*, then such discrepancies may be interpreted in two ways. Some of the discrepancies found may expose *undesirable deviations*, i. e., conformance checking signals the need for a better control of the process. Other discrepancies may reveal *desirable deviations*. A naïve approach toward conformance checking would be to simply count the fraction of cases that can be "parsed completely" (i. e., the proportion of cases corresponding to firing sequences leading from [start] to [end]). Thus, this proportion of event log behaviour according to the process model is measured using a quantified quality criteria, called *fitness*. The other three quality criteria i. e.,*generalisation* (the discovered model should generalize the example behaviour seen in the event log), *precision* (the discovered model should not allow for behaviour completely unrelated to what was seen in the event log), *simplicity* (the discovered model should be as simple as possible) are less relevant for conformance checking (van der Aalst et al., 2012).
There are various ways of defining fitness. It can be defined at the case level, e. g., the fraction of traces in the log that can be fully replayed. It can also be defined at the event level, e. g., the fraction of events in the log that are indeed possible according to the model.

The fitness of a case with trace σ on WF-net N is defined as follows:

$$\mathit{fitness}\,(\sigma, N) = \frac{1}{2}(1 - \frac{m}{c}) + \frac{1}{2}(1 - \frac{r}{p})$$

where p=produced tokens, c=consumed tokens, m=missing tokens), and r=remaining tokens. For an illustrative example reader may have an insight to Chapter 7 in (van der Aalst et al., 2012).
Clearly, the α-algorithm is unable to extract a target model that balances the four quality criteria. Therefore, more advanced approaches are presented such as *heuristic mining* (takes into account frequency of events and sequences to construct model) (Weijters and van der Aalst, 2003), *genetic minig* (an iterative procedure to select the best individual i. e., a process model out of many models created randomly, after applying fitness, crossover and mutation as design choices)(de Medeiros et al., 2007), *fuzzy mining* (construct hierarchical models based on frequency of activities and paths) (Günther and van der Aalst, 2007), and *region-based mining* (construct a system model from a description of its behaviour using transition systems) (See Chapter 6 in (van der Aalst et al., 2012)). Furthermore, some combinations of aforementioned approaches e. g., genetic mining with heuristic mining etc., are proposed to be more suitable for practical applications.
All these algorithms have been implemented in the forms of plugins in a generic open-source framework called ProM[2]. Process mining may be used to improve the alignment of information systems, business processes, and the organization. By analysing the real processes and diagnosing discrepancies, new insights can be gathered showing how to improve the support by information systems.

The third technique, *enhancement* extends or improves an existing process model using information about the actual process recorded in some event log. There exist two types of enhancements: *repair* (i. e., align process model or the process itself with reality) and *extension* (add a new perspective to the process model by cross-correlating it with the log). Besides the control-flow perspective other perspectives may be added to the model as follows:

- *organizational perspective*: analyse the social network and subsequently identify organizational entities that connect activities to groups of resources.

- *case perspective*: use case and event attributes in the log for decision mining. This shows which data is relevant and should be included in the model.

- *time perspective*: use timestamps and frequencies to learn probability distributions that adequately describe waiting and service times and routing probabilities.

Interested reader may refer to Chapter 8 in (van der Aalst et al., 2012).
Currently, ProM allows for the discovery of different process perspectives including the control-flow perspective ("How?"), the case perspective ("What?"), the organizational perspective ("Who?"), and time perspective (When?). Nevertheless, ProM does not support the management of collections of models and logs.

[2]www.processmining.org

6.2.3 Process model collections management techniques

As organizations start to develop and maintain large collections of process models, there is an increasing need for continuous and efficient management of these process repositories. In order to reduce redundancy and improve maintainability of process model collections, efficient techniques to manage multiple process models in relation to each other as well as management of different versions of a single model are required.
An overview of state-of-the-art management techniques for process model collections is discussed in (Dijkman et al., 2012). Figure 6.4 represents the management techniques which are discussed in sections sections 6.2.3.1 to 6.2.3.3

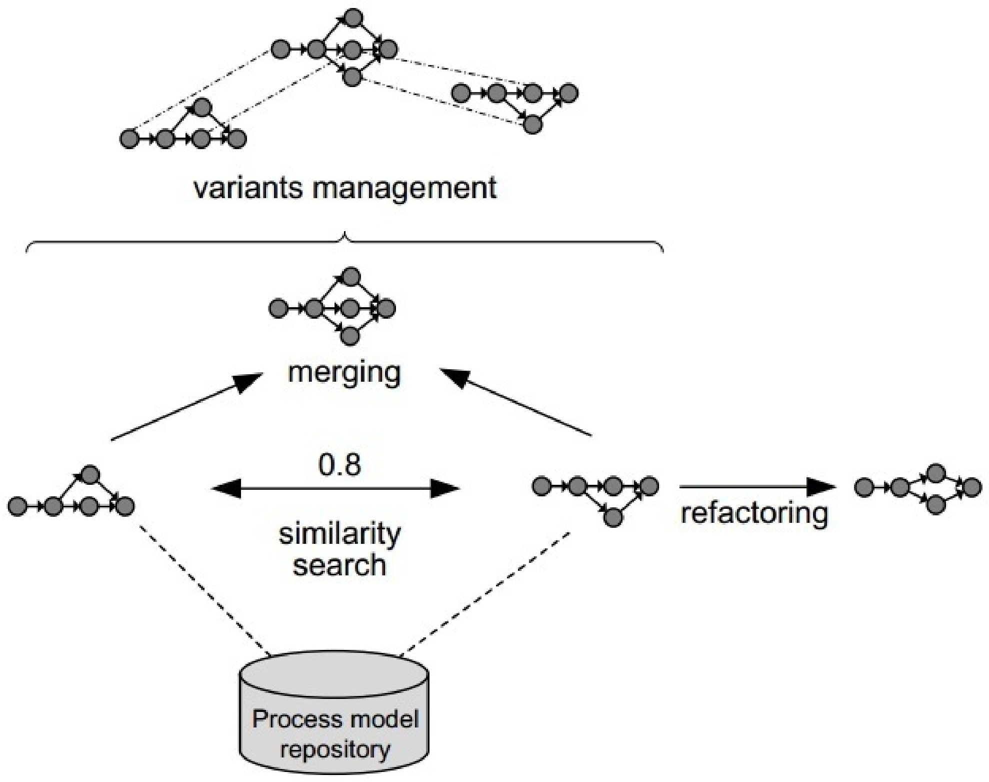


Figure 6.4: Overview of techniques for the management of process model collections adapted from (Dijkman et al., 2012)

If a collection of process models is given, variants management mechanisms (Pascalau et al., 2011; Ekanayake et al., 2011; Weidlich et al., 2011) are required to keep track of the organization of the collection such that users can browse it and view its evolution as changes are made.
Various types of relations exists between the process models of the repository based on these mechanisms. For example, (Ekanayake et al., 2011) exploit information on shared clones across process models and versions thereof, in order to provide change propagation and access control features. Other common relations that can be used to manage variants are aggregation and generalization relations (Kurniawan et al., 2012). An aggregation re-

lation exists between a business process model and its parts (usually sub-processes) while a generalization relation exists between a more general process model and a more specific one.
To develop a hierarchical classification of process models aggregation and generalization are typically used, thus enabling users to navigate a collection of process models by traversing the hierarchy. While there are too many techniques for managing large process model collections, a collection remains "static" in most cases until a process improvement initiative is carried out.
To move towards "selfadapting" process model collections (van der Aalst et al., 2014) recently propose to integrate process mining and the management of process model collections. In so doing, *Apromore* is proposed as a solution for consolidation(La Rosa et al., 2013) and analysis(Conforti et al., 2015) of liquid process model collections.

6.2.3.1 Process Similarity search

The notion of similarity search (Becker and Laue, 2012) is defined in the set of management techniques of model collections. Given a collection of process models and a search process model, similarity search techniques identify and return those models from the collection that are deemed similar (e. g., potentially inexact matches) to the search model. A potential use case of similarity search is for an organization to identify which of its own processes are similar to a standardized (reference) process model. To handle this challenge researchers proposed:

(i) the definition and implementation of similarity measures that return a similarity rating (e. g., between 0 and 1) for two process models (Dijkman et al., 2011a); and

(ii) the implementation of indexing techniques for improving the retrieval of models containing the query model as a sub-graph (e. g.,(Jin et al., 2013)) or those of similar process models (Yan et al., 2010)

More in general, for querying efficiently the repositories of process models such indexing techniques can be used. Queries can be expressed in textual form (e. g., (Jin et al., 2011)) or can be expressed as a model (e. g., (Jin et al., 2013; Awad and Sakr, 2012)).

6.2.3.2 Process Merging

Another set of management techniques relates to the notion of process model merging. Process model merging aim to merge a collection of process variants into one consolidated process model which can be very useful in the context of organizational mergers, restructurings and rationalizations. This can lead to a collection of reduced size that has been standardized and optimized for the current business context, which in turn can significantly improve the maintainability of the collection as a whole.
Some of the merging techniques take into account notions of label similarities when merging models. Then, it is possible to merge activities with identical labels only (Reijers et al., 2009; Sun et al., 2006; Mendling and Simon, 2006) or merge activities from different models that have similar but not identical labels (Gottschalk et al., 2008a; La Rosa et al., 2010).

Some other merging techniques enforce behaviour-preservation such that the merged model maintains the behaviour of all individual models allowing one to replay the behaviour of each input variant on the merged model (La Rosa et al., 2013).

6.2.3.3 Process Refactoring

As a collection of process models evolves over time, it may start to display unnecessary internal complexity. A common example is redundancy in the form of exact or approximate clones. Such clones are typically the result of copy/paste activities and they adversely affect the maintainability of process model collections, besides leading to unwanted inconsistencies in the repository, if they are modified independently of each other. Clones can manifest themselves both at the level of entire process models as well as fragments thereof (van der Aalst et al., 2014).
Researchers have proposed various techniques for detecting such clones within process model repositories (see e. g., (Guo and Zou, 2008; Ekanayake et al., 2012; Dumas et al., 2013)). Refactoring techniques, inspired from software engineering, have been explored to improve the maintainability and readability of process model collections. Examples of such refactoring techniques are extracting the identified clones and storing them as reusable subprocesses (Dumas et al., 2013),standardizing approximate clones (Ekanayake et al., 2012), and modularizing process models into different levels of abstraction (Weber et al., 2011; Dijkman et al., 2011b, 2009).

6.3 Summary and discussions

Finally, section 6.3 summarizes the main aspect of the process mining discipline to bridge the gap between BPM and WfM on the one hand and DM, BI, and ML on the other hand. Three relevant techniques of process mining were discussed: *discovery*(takes an event log and produces a model without using any a-priori information), *conformance*(an existing process model is compared with an event log of the same process) and *enhancement*(check if reality, as recorded in the log, conforms to the model and vice versa). To reduce redundancy and to improve maintainability of process model collections, efficient techniques to manage multiple process models in relation to each other as well as management of different versions of a single model have been proposed and discussed superficially. Next chapter describes how we design and implemented the prototype as a proof-of-concept.

CHAPTER 7

Proof-of-Concept Prototype

Chapter 7 gives information about the realization of our proposed method developing a prototype solution as a proof-of concept to demonstrate the functionality and to verify that our introduced new concepts can be achieved in a practical implementation. It starts by the relational database (see Section 7.1) generated automatically from the meta-model and after populating it with data about process models and logs we generate all steps (activities) consolidated into a hierarchy which is derived into a customized process dimension of our process warehouse (see Section 7.2) .

7.1 Relational database

To demonstrate the functionality of our meta-model approach, we have implemented a solution, a generated relational database (dB) from our meta-model. After parsing all process models and logs to populate the dB we apply different algorithms to generate the hierarchy between these processes. In the following Figure 7.1 is a graphical representation of how we achieved the importing, parsing and loading to our dB all process variants and respective logs, together with the reference process model.

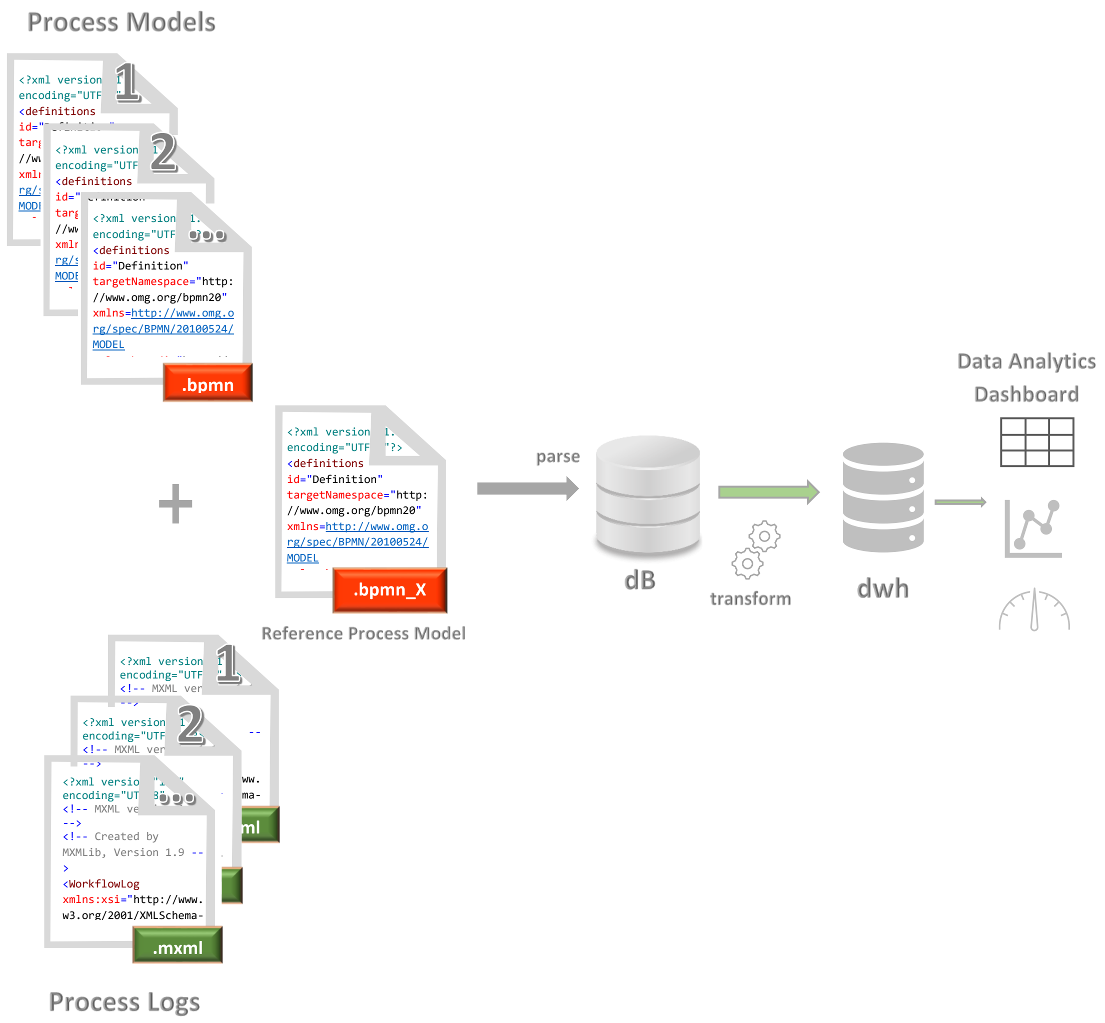
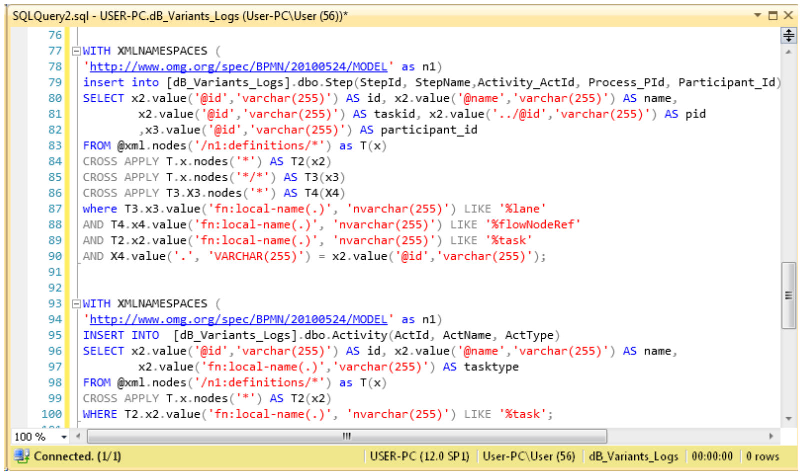


Figure 7.1: Process models and logs relational database

Data about processes are stored in *BPMN* format whereas data about simulated logs are stored in *MXML* format. Obviously, there are a lot of free and open-source tools that perform different type of conversions of different formats. For instance, *ProM* and *ProM Import*[1] convert your logs to *MXML*[2] through different serialization methods based on the original file format. These xml-based files should be validated against their xml schema definition before importing. Specifically, we created a job script in SQL which is executed in one of the installed instance of MSSQL Server to automatically perform importing and exporting all relevant data about processes to our dB. This job contains some defined steps as **T-SQL** code which are executed consecutively. Moreover, we give an excerpt of the script dealing with parsing the process data from variants and logs (i. e. xml-based files) as given in the screenshot of Figure 7.2 :

[1]Process mining import framework: http://www.promtools.org/promimport/
[2]*XML* schema definition of *MXML*files, http://www.processmining.org/WorkflowLog.xsd

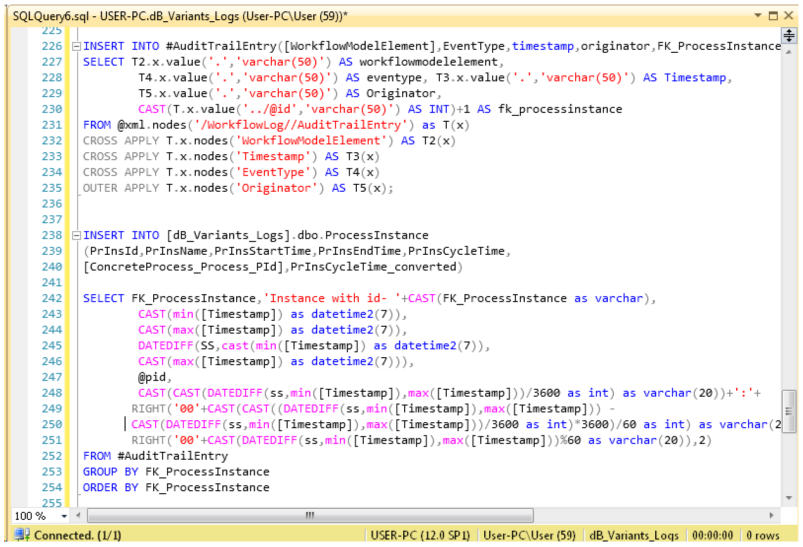

SQLQuery2.sql - USER-PC.dB_Variants_Logs (User-PC\User (56))*

```sql

WITH XMLNAMESPACES (
'http://www.omg.org/spec/BPMN/20100524/MODEL' as n1)
insert into [dB_Variants_Logs].dbo.Step(StepId, StepName,Activity_ActId, Process_PId, Participant_Id)
SELECT x2.value('@id','varchar(255)') AS id, x2.value('@name','varchar(255)') AS name,
        x2.value('@id','varchar(255)') AS taskid, x2.value('../@id','varchar(255)') AS pid
        ,x3.value('@id','varchar(255)') AS participant_id
FROM @xml.nodes('/n1:definitions/*') as T(x)
CROSS APPLY T.x.nodes('*') AS T2(x2)
CROSS APPLY T.x.nodes('*/*') AS T3(x3)
CROSS APPLY T3.X3.nodes('*') AS T4(X4)
where T3.x3.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%lane'
AND T4.x4.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%flowNodeRef'
AND T2.x2.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%task'
AND X4.value('.', 'VARCHAR(255)') = x2.value('@id','varchar(255)');


WITH XMLNAMESPACES (
'http://www.omg.org/spec/BPMN/20100524/MODEL' as n1)
INSERT INTO  [dB_Variants_Logs].dbo.Activity(ActId, ActName, ActType)
SELECT x2.value('@id','varchar(255)') AS id, x2.value('@name','varchar(255)') AS name,
        x2.value('fn:local-name(.)','varchar(255)') AS tasktype
FROM @xml.nodes('/n1:definitions/*') as T(x)
CROSS APPLY T.x.nodes('*') AS T2(x2)
WHERE T2.x2.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%task';
```

100 %

Connected. (1/1) | USER-PC (12.0 SP1) | User-PC\User (56) | dB_Variants_Logs | 00:00:00 | 0 rows

(a) T-SQL script of parsing process variants

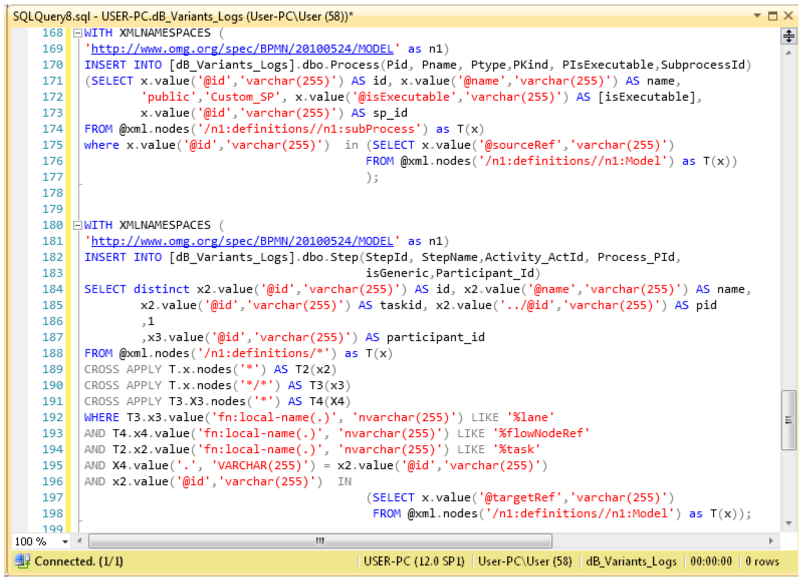

SQLQuery6.sql - USER-PC.dB_Variants_Logs (User-PC\User (59))*

```sql

INSERT INTO #AuditTrailEntry([WorkflowModelElement],EventType,timestamp,originator,FK_ProcessInstance
SELECT T2.x.value('.','varchar(50)') AS workflowmodelelement,
        T4.x.value('.','varchar(50)') AS eventype, T3.x.value('.','varchar(50)') AS Timestamp,
        T5.x.value('.','varchar(50)') AS Originator,
        CAST(T.x.value('../@id','varchar(50)') AS INT)+1 AS fk_processinstance
FROM @xml.nodes('/WorkflowLog//AuditTrailEntry') as T(x)
CROSS APPLY T.x.nodes('WorkflowModelElement') AS T2(x)
CROSS APPLY T.x.nodes('Timestamp') AS T3(x)
CROSS APPLY T.x.nodes('EventType') AS T4(x)
OUTER APPLY T.x.nodes('Originator') AS T5(x);


INSERT INTO [dB_Variants_Logs].dbo.ProcessInstance
(PrInsId,PrInsName,PrInsStartTime,PrInsEndTime,PrInsCycleTime,
[ConcreteProcess_Process_PId],PrInsCycleTime_converted)

SELECT FK_ProcessInstance,'Instance with id- '+CAST(FK_ProcessInstance as varchar),
        CAST(min([Timestamp]) as datetime2(7)),
        CAST(max([Timestamp]) as datetime2(7)),
        DATEDIFF(SS,cast(min([Timestamp]) as datetime2(7)),
        CAST(max([Timestamp]) as datetime2(7))),
        @pid,
        CAST(CAST(DATEDIFF(ss,min([Timestamp]),max([Timestamp]))/3600 as int) as varchar(20))+':'+
       RIGHT('00'+CAST(CAST((DATEDIFF(ss,min([Timestamp]),max([Timestamp])) -
      |CAST(DATEDIFF(ss,min([Timestamp]),max([Timestamp]))/3600 as int)*3600)/60 as int) as varchar(2
       RIGHT('00'+CAST(DATEDIFF(ss,min([Timestamp]),max([Timestamp]))%60 as varchar(20)),2)
FROM #AuditTrailEntry
GROUP BY FK_ProcessInstance
ORDER BY FK_ProcessInstance

```

100 %

Connected. (1/1) | USER-PC (12.0 SP1) | User-PC\User (59) | dB_Variants_Logs | 00:00:00 | 0 rows

(b) T-SQL script of parsing process logs


SQLQuery8.sql - USER-PC.dB_Variants_Logs (User-PC\User (58))*

```sql
WITH XMLNAMESPACES (
'http://www.omg.org/spec/BPMN/20100524/MODEL' as n1)
INSERT INTO [dB_Variants_Logs].dbo.Process(Pid, Pname, Ptype,PKind, PIsExecutable,SubprocessId)
(SELECT x.value('@id','varchar(255)') AS id, x.value('@name','varchar(255)') AS name,
        'public','Custom_SP', x.value('@isExecutable','varchar(255)') AS [isExecutable],
        x.value('@id','varchar(255)') AS sp_id
FROM @xml.nodes('/n1:definitions//n1:subProcess') as T(x)
where x.value('@id','varchar(255)')  in (SELECT x.value('@sourceRef','varchar(255)')
                                        FROM @xml.nodes('/n1:definitions//n1:Model') as T(x))
                                        );


WITH XMLNAMESPACES (
'http://www.omg.org/spec/BPMN/20100524/MODEL' as n1)
INSERT INTO [dB_Variants_Logs].dbo.Step(StepId, StepName,Activity_ActId, Process_PId,
                                        isGeneric,Participant_Id)
SELECT distinct x2.value('@id','varchar(255)') AS id, x2.value('@name','varchar(255)') AS name,
        x2.value('@id','varchar(255)') AS taskid, x2.value('../@id','varchar(255)') AS pid
        ,1
        ,x3.value('@id','varchar(255)') AS participant_id
FROM @xml.nodes('/n1:definitions/*') as T(x)
CROSS APPLY T.x.nodes('*') AS T2(x2)
CROSS APPLY T.x.nodes('*/*') AS T3(x3)
CROSS APPLY T3.X3.nodes('*') AS T4(X4)
WHERE T3.x3.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%lane'
AND T4.x4.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%flowNodeRef'
AND T2.x2.value('fn:local-name(.)', 'nvarchar(255)') LIKE '%task'
AND X4.value('.', 'VARCHAR(255)') = x2.value('@id','varchar(255)')
AND x2.value('@id','varchar(255)')  IN
                                        (SELECT x.value('@targetRef','varchar(255)')
                                         FROM @xml.nodes('/n1:definitions//n1:Model') as T(x));
```

100 %

Connected. (1/1) | USER-PC (12.0 SP1) | User-PC\User (58) | dB_Variants_Logs | 00:00:00 | 0 rows

(c) T-SQL script of parsing reference process model

Figure 7.2: A screenshot of T-SQL script to parse and load process model variants and logs

We mainly used bulk load operations to import xml data as temporary objects and then wrote some complex queries using *Xquery* methods to combine and parse the process data to our relevant tables. To identify and parse the reference process model as a generic(global) process we exploited the annotated stereotypes which are captured in the bpmn file with the special attribute value *Stereotype* under the <*Model*> and <*ModelType*> tag elements. After the exporting task is finished we ensure that the referential integrity constraints in our dB are successfully validated and held to confirm the consistency of the database.

7.2 Process warehouse implementation

To establish the practical feasibility of our process warehouse approach, we have implemented a solution on analysis services of SQL Server 2018. After specifying the data source and data source view of our PW we manage to deploy the cube for a comprehensive analysis of process variants, namely Cube_Variants. Below we outline the cube and measures defined.

7.2.1 Cube design and deployment

In figure Figure 7.3 we show a design of the cube after specifying the view of our data source(PW schema):

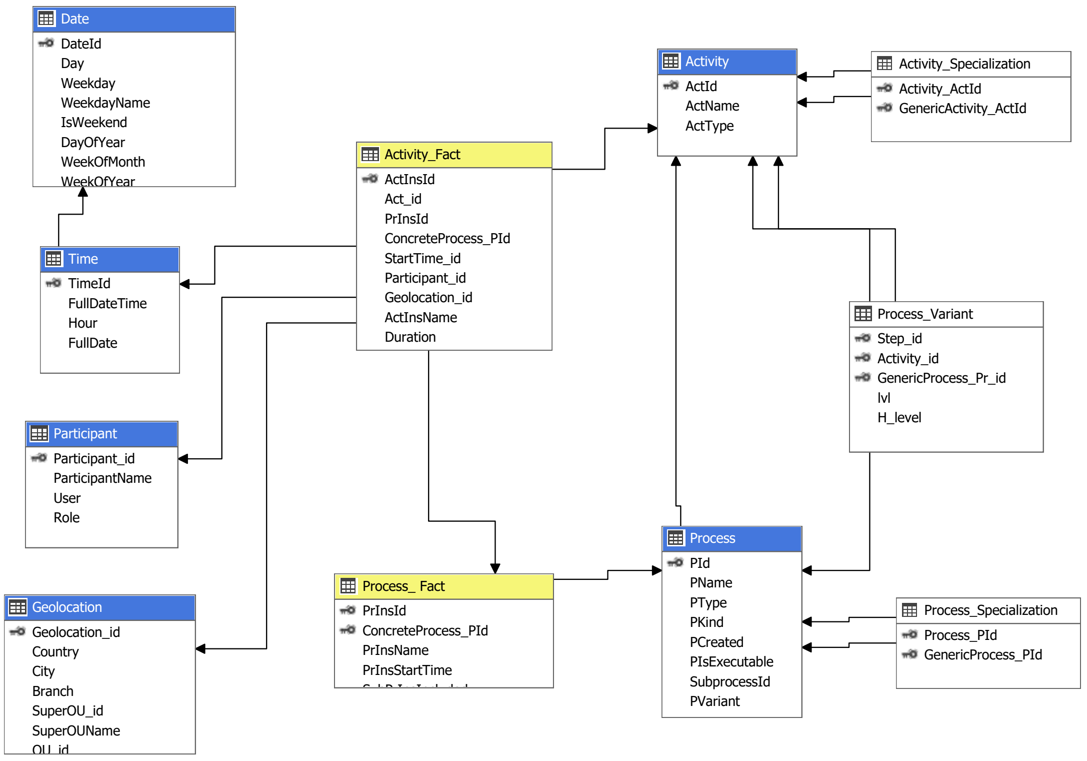
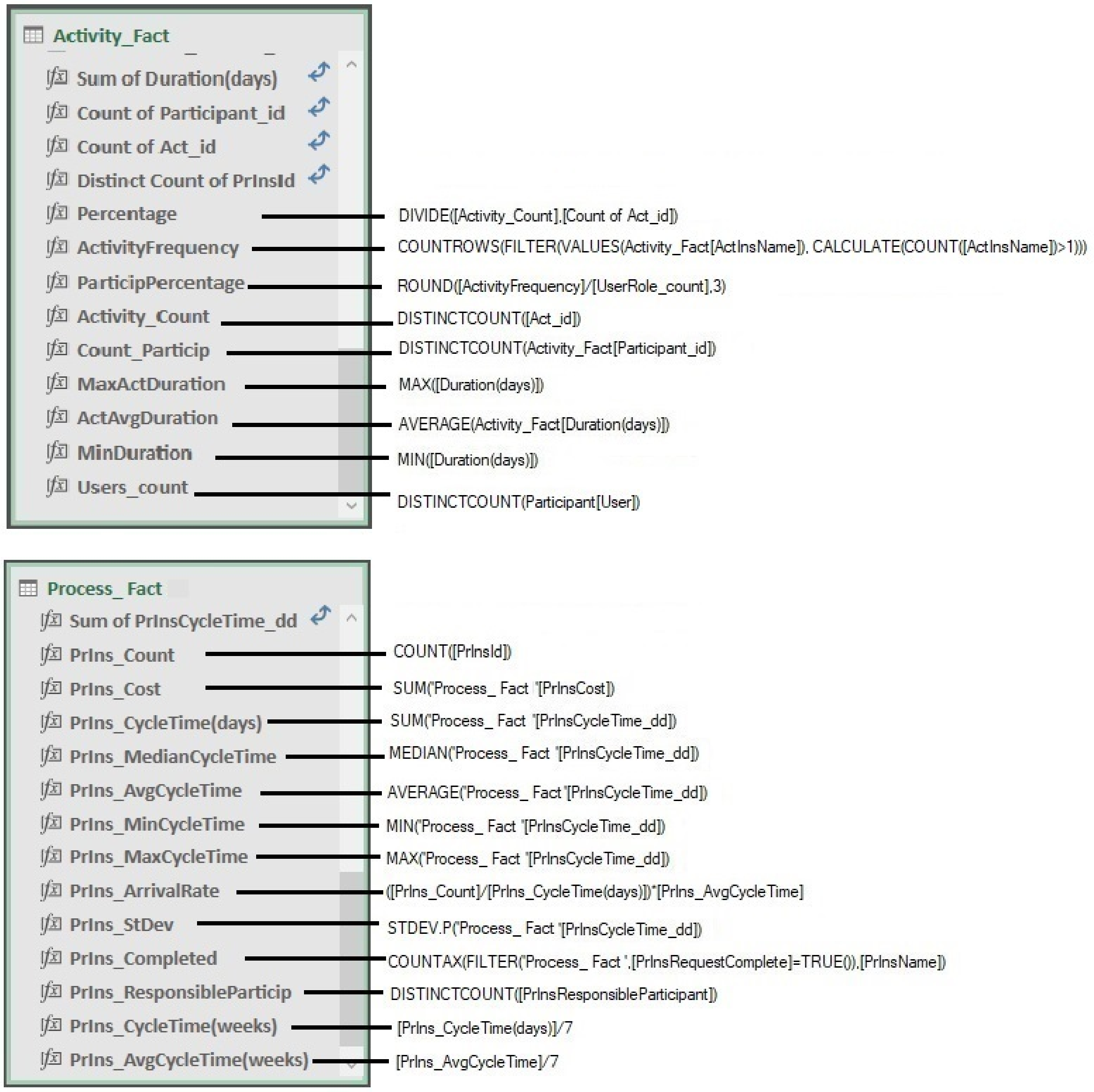


Figure 7.3: Cube design of process warehouse data source view

As shown in the above figure in the center we align two fact tables: *Activity_Fact* and *Process_Fact* indicating what activity instances and what process instances are recorded respectively. Whereas the other surrounded tables are dimension tables that stores how these activities and processes should be reported. After the successful deployment of the cube we need to use a business intelligence tool to perform analysis based on new relevant measures to answer different queries.

7.2.2 Measures

Here, we explain measures we created by using power pivot bi add-in in excel as a business intelligence tool. Power Pivot performs powerful data analysis by using formula language in Power Pivot called Data Analysis Expression (DAX). Obviously, the usage of DAX is clear: to define custom calculations for Calculated Columns and for Measures (also known as calculated fields) and KPIs. After we make the connection with our previously described data source the data model is generated on the fly. Then, we start defining the relevant measures to answer different queries combined in different power pivot tables.
We show all measures defined through different tables and their respective formulas as shown in Figure 7.4 and Figure 7.5. All measures defined in Figure 7.4 are self explanatory such as *ActAvgDuration* which calculates the average duration of an executed activity by using the aggregated function *Average* or *ActivityFrequency* that finds the number of times an activity occurs by using *countrows* and *filter* functions etc.



Activity_Fact	
Sum of Duration(days)	
Count of Participant_id	
Count of Act_id	
Distinct Count of PrInsId	
Percentage	DIVIDE([Activity_Count],[Count of Act_id])
ActivityFrequency	COUNTROWS(FILTER(VALUES(Activity_Fact[ActInsName]), CALCULATE(COUNT([ActInsName])>1)))
ParticipPercentage	ROUND([ActivityFrequency]/[UserRole_count],3)
Activity_Count	DISTINCTCOUNT([Act_id])
Count_Particip	DISTINCTCOUNT(Activity_Fact[Participant_id])
MaxActDuration	MAX([Duration(days)])
ActAvgDuration	AVERAGE(Activity_Fact[Duration(days)])
MinDuration	MIN([Duration(days)])
Users_count	DISTINCTCOUNT(Participant[User])

Process_ Fact	
Sum of PrInsCycleTime_dd	
PrIns_Count	COUNT([PrInsId])
PrIns_Cost	SUM('Process_ Fact '[PrInsCost])
PrIns_CycleTime(days)	SUM('Process_ Fact '[PrInsCycleTime_dd])
PrIns_MedianCycleTime	MEDIAN('Process_ Fact '[PrInsCycleTime_dd])
PrIns_AvgCycleTime	AVERAGE('Process_ Fact '[PrInsCycleTime_dd])
PrIns_MinCycleTime	MIN('Process_ Fact '[PrInsCycleTime_dd])
PrIns_MaxCycleTime	MAX('Process_ Fact '[PrInsCycleTime_dd])
PrIns_ArrivalRate	([PrIns_Count]/[PrIns_CycleTime(days)])*[PrIns_AvgCycleTime]
PrIns_StDev	STDEV.P('Process_ Fact '[PrInsCycleTime_dd])
PrIns_Completed	COUNTAX(FILTER('Process_ Fact ',[PrInsRequestComplete]=TRUE()),[PrInsName])
PrIns_ResponsibleParticip	DISTINCTCOUNT([PrInsResponsibleParticipant])
PrIns_CycleTime(weeks)	[PrIns_CycleTime(days)]/7
PrIns_AvgCycleTime(weeks)	[PrIns_AvgCycleTime]/7

Figure 7.4: Measures listed with formulas (1)

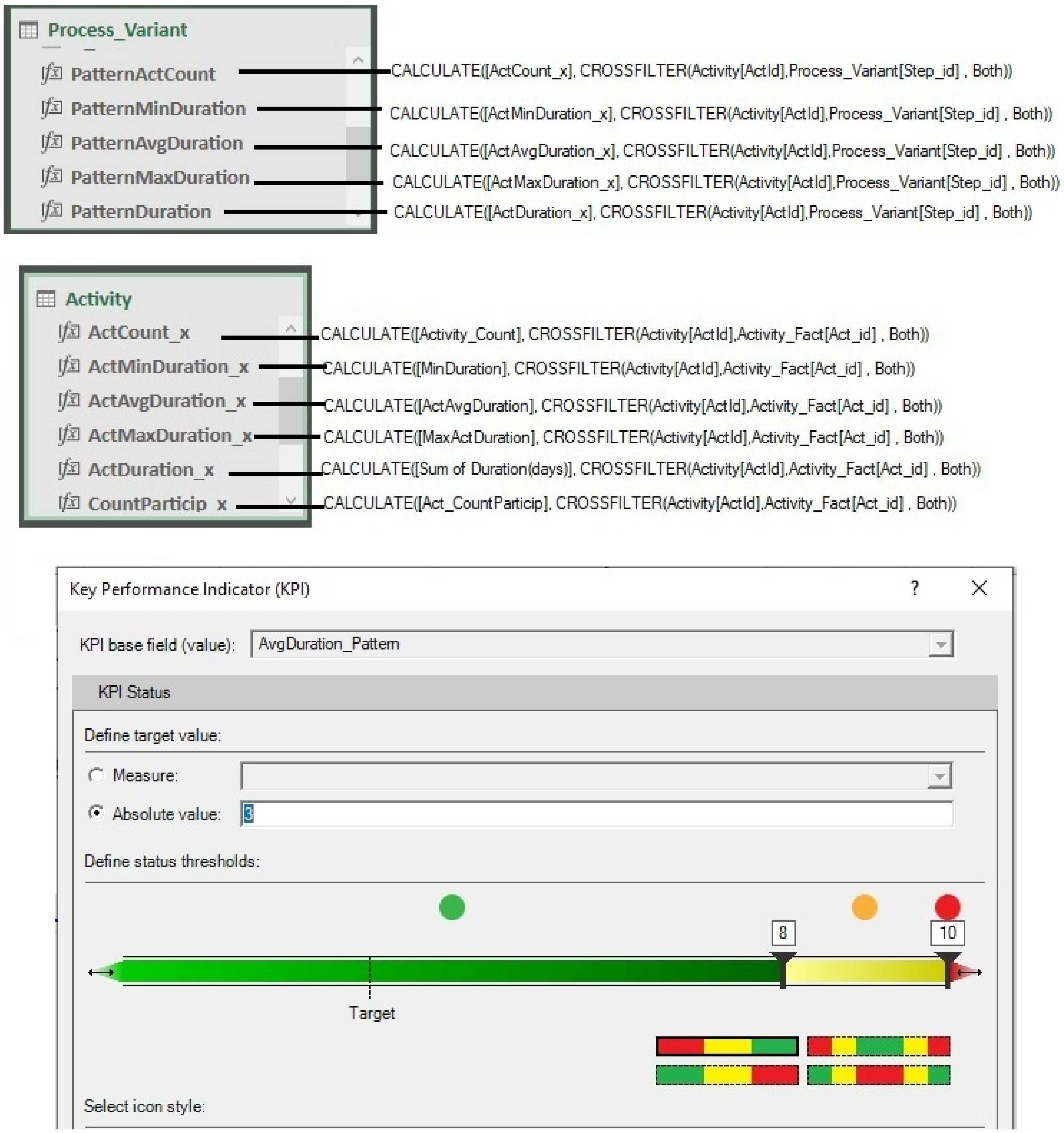
Whereas the second list of measures as shown in Figure 7.5 is more complex as calculations are based and performed on relationships that exists between columns on different tables. We use bidirectional *cross-filtering* in PowerBi as a new added feature in Excel 2016. In so doing, we as data modellers have more control over how we can apply filters when working with related tables.
As displayed in the following figure some of the measures where *cross-filtering* syntax is used are:

- $ActAvgDuration\_x = CALCULATE([ActAvgDuration],$

$CROSSFILTER(Activity[ActId], Activity\_Fact[Act\_id], Both))$

- $PatternAvgDuration = CALCULATE([ActAvgDuration\_x],$
  $CROSSFILTER(Activity[ActId], Process\_Variant[Step\_id], Both))$

We apply these filters cause we want to do calculations over different patterns of different process variants but all patterns are recorded in *ProcessVariant* table. As patterns consist of different activities and some activity instances related measures already defined in the first list (as shown in Figure 7.4) belongs to fact table *Activity_Fact*. So, first we need to apply filters by propagating the filter context (i. e., *Activity_Fact[Act_id]*)to the second related table (*Activity[ActId]*)on the other side of the table relationship. Afterwards, we have to apply bidirectional cross-filtering calculations over two related tables on both side (i. e., *Activity[ActId]*, *ProcessVariant[Step_id]*) of this relationship.



Process_Variant

PatternActCount	CALCULATE([ActCount_x], CROSSFILTER(Activity[ActId],Process_Variant[Step_id] , Both))
PatternMinDuration	CALCULATE([ActMinDuration_x], CROSSFILTER(Activity[ActId],Process_Variant[Step_id] , Both))
PatternAvgDuration	CALCULATE([ActAvgDuration_x], CROSSFILTER(Activity[ActId],Process_Variant[Step_id] , Both))
PatternMaxDuration	CALCULATE([ActMaxDuration_x], CROSSFILTER(Activity[ActId],Process_Variant[Step_id] , Both))
PatternDuration	CALCULATE([ActDuration_x], CROSSFILTER(Activity[ActId],Process_Variant[Step_id] , Both))

Activity

ActCount_x	CALCULATE([Activity_Count], CROSSFILTER(Activity[ActId],Activity_Fact[Act_id] , Both))
ActMinDuration_x	CALCULATE([MinDuration], CROSSFILTER(Activity[ActId],Activity_Fact[Act_id] , Both))
ActAvgDuration_x	CALCULATE([ActAvgDuration], CROSSFILTER(Activity[ActId],Activity_Fact[Act_id] , Both))
ActMaxDuration_x	CALCULATE([MaxActDuration], CROSSFILTER(Activity[ActId],Activity_Fact[Act_id] , Both))
ActDuration_x	CALCULATE([Sum of Duration(days)], CROSSFILTER(Activity[ActId],Activity_Fact[Act_id] , Both))
CountParticip_x	CALCULATE([Act_CountParticip], CROSSFILTER(Activity[ActId],Activity_Fact[Act_id] , Both))

Key Performance Indicator (KPI)

KPI base field (value): AvgDuration_Pattern

KPI Status

Define target value:

Measure:

[✓ Absolute value:] 3

Define status thresholds:

Select icon style:

Figure 7.5: Measures listed with formulas (2) and an example of KPI

In the bottom of this figure we give a screenshot of how we create a KPI, e. g., *AvgDurationPattern_KPI* by defining a *base field(value)* which in our case is *AvgDurationPattern* to evaluate the current value and the status against a defined target. The target here can be either an absolute value (i. e., as shown for instance number 3, so the average duration target of a pattern is 3 days) or another existing measure. The status thresholds is between value:8 and 10, which means that if average duration for pattern is ≤ 8 then it's status is good(green status), whereas ≥ 10 is bad (red status) and between 8 and 10 is moderated (yellow status). These KPIs can be customized using different icon styles. An example of it we show on next chapter.

CHAPTER 8

Evaluation Results

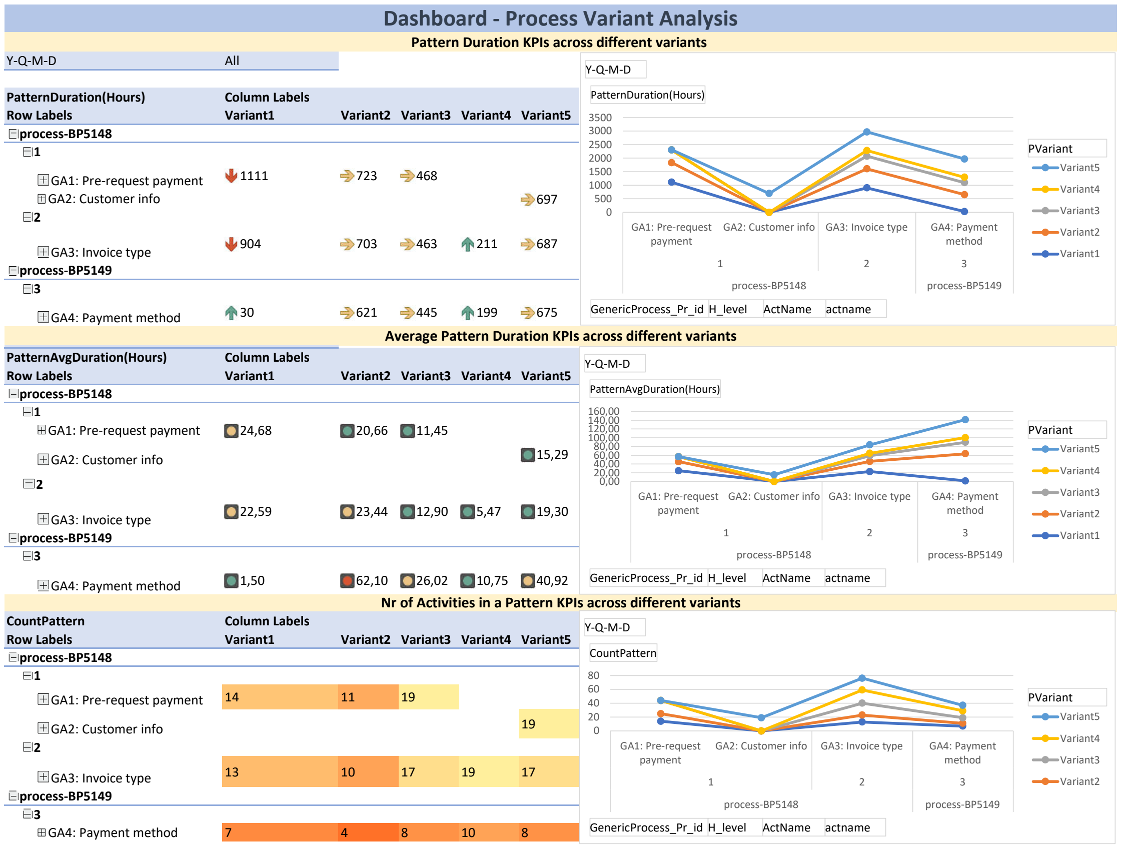
Chapter 8 gives results from cube deployment of the process warehouse based on two case studies, e. g., synthetic event logs of processing customer invoice payments and real-life event logs for a building permit application of five Dutch municipalities.

8.1 Case study 1: Synthetic event logs

In this first case study we consider synthetic event logs generated from simulation using BIMP tool[1]. They represent artificial event logs from processing customer invoice payments process variants. We considered 100 process instances (i. e., cases) for each process in a variant (a total of 1000 process instances), 5536 activity instances and 13114 events (start and complete) generated for all process instances. We have the information of which event occurs in which process instance that belongs to which concrete process. We focus on analysing time-perspective measures as we don't have information about cost of each process instance.

We use *PowerPivot* in Excel 2018 to perform powerful data analysis where our data is process-related. Based on different calculation columns and measures we created we build different Pivot tables to answer different queries.
In the dashboard are summarized different aggregated measures with corresponding lines charts.

[1]http://bimp.cs.ut.ee/simulator



Dashboard - Process Variant Analysis

Pattern Duration KPIs across different variants

Y-Q-M-D All

PatternDuration(Hours) Row Labels	Column Labels Variant1	Variant2	Variant3	Variant4	Variant5
⊟process-BP5148					
⊟1					
⊞GA1: Pre-request payment	↓1111	→723	→468		
⊞GA2: Customer info					→697
⊟2					
⊞GA3: Invoice type	↓904	→703	→463	↑211	→687
⊟process-BP5149					
⊟3					
⊞GA4: Payment method	↑30	→621	→445	↑199	→675

Average Pattern Duration KPIs across different variants

PatternAvgDuration(Hours) Row Labels	Column Labels Variant1	Variant2	Variant3	Variant4	Variant5
⊟process-BP5148					
⊟1					
⊞GA1: Pre-request payment	●24,68	●20,66	●11,45		
⊞GA2: Customer info					●15,29
⊟2					
⊞GA3: Invoice type	●22,59	●23,44	●12,90	●5,47	●19,30
⊟process-BP5149					
⊟3					
⊞GA4: Payment method	●1,50	●62,10	●26,02	●10,75	●40,92

Nr of Activities in a Pattern KPIs across different variants

CountPattern Row Labels	Column Labels Variant1	Variant2	Variant3	Variant4	Variant5
⊟process-BP5148					
⊟1					
⊞GA1: Pre-request payment	14	11	19		
⊞GA2: Customer info					19
⊟2					
⊞GA3: Invoice type	13	10	17	19	17
⊟process-BP5149					
⊟3					
⊞GA4: Payment method	7	4	8	10	8

Figure 8.1: Dashboard of process variants analysis

In Figure 8.1 we show results about duration(days) of four process patterns across the five process variants (request invoice and pay) for a short period. Exactly there are only three months during year 2018 and two other months in 2019; this coincides with the date-time we run the simulations. Obviously, there are not so much data generated in order to have considerable information to share but enough to evaluate as a proof of concept. More analysis we present in the second case scenario with real-life data. On top of the above dashboard we report on duration of process patterns across five variants. To help users to quickly evaluate the current values a corresponding KPI is created where a target value is given and then a status threshold between a low and high threshold is specified. An alternative way to show visual measures of performance is trough a *conditional-if* rules. The rule in this case is: "if value $<$ 33% then *PatternDuration*(in hours) metric is good, else if $>=$67% is bad". So, each value is associated with an arrow icon to indicate either positive performance (up green arrow), poor performance (down red arrow) and in-between (right orange arrow). Furthermore, users can filter these results to only a specific year/quarter/-month or day and consecutively the charts are automatically generated.
Variant 4 (Receive Invoice) has the best performance in pattern *GA2*, which means that the part of the process after the type of invoice is performed last shorter than any other variant. This variant reports good performance as well for the last pattern *GA4*, but the shortest time in processing payment context part goes to Variant 1. In contrast this variant

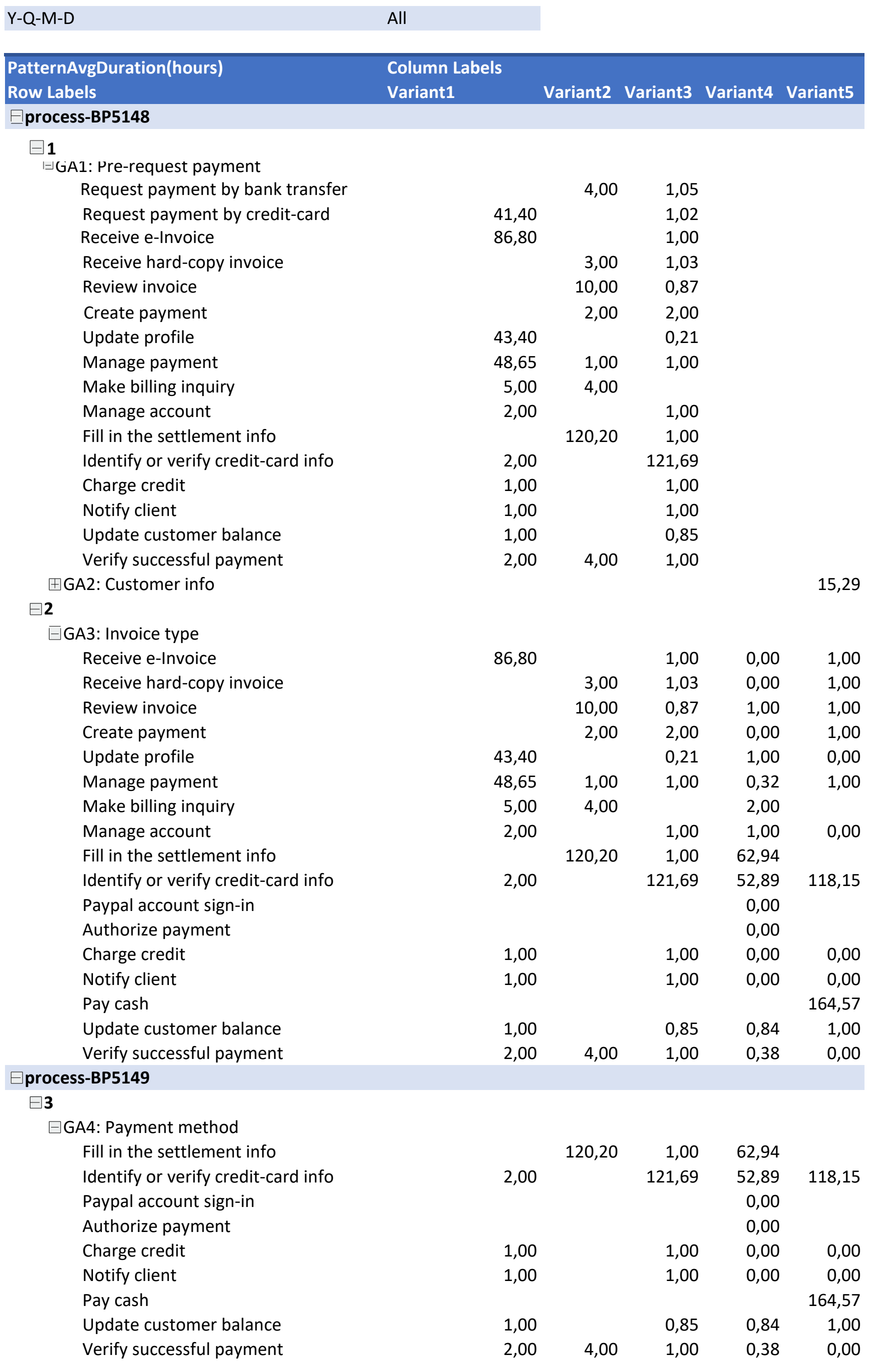
presents the greatest values in processing patterns: *GA1* and *GA2*. Whereas, variant 3, 2 and 5 report some values in-between. There are some empty values among the cells in the dashboard which means that a corresponding value of the pattern for a specific variant is missing, so does not exist.
In the middle part of the dashboard we demonstrate the average duration of process patterns across the five variants. From the results reported variant 4 shows the best performance followed by variant 3 and then variant 5. Variant 1 again reports a lower processing time in *GA4* but not so good values on the other two patterns. Whereas, variant 2 shows the greatest value in handling payment processing.
These results show case that the third research question is answered where KPIs are computed in different parts of different variants.

The bottom results in the dashboard reports on how many activities are performed by each process pattern. We highlight these values with a two-color scale from light yellow(for greater values) to dark orange (for smaller values). Variant 1 reports the lowest number of activities involved in all the three patterns instead variant 5 the greatest number. Some considerable number of activities report as well the other two variants 3 and 4, whereas variant 1 and 2 have lower values. So, as an overall variant 3, 4 and 5 report good performance whereas variant 1 and 2 a bad performance.
Instead if we consider specific patterns the results differs as stated above. So, we manage to offer a comparison between different patterns at different levels of genericity across the five variants and as well answers the queries from Chapter 5 as follows:

- Display the cycle time of patterns "$P_Y, *$" from all variants, where $P_Y = G \; or \; H$ i. e., all processed payments after customer receives his/her invoice?
- Display the average process duration of payments with patterns "$P_Z, *$", where $P_Z = O, P, Q, R$, i. e., of different payment options?
- Display the cycle time of patterns "$P_X, *$" for each process variant, where $P_X = C \; or \; D \; or \; E \; or \; F$?

In addition, other queries can be answered based on other dimensions in our PWH schema, such as the organization unit each participant belongs, e. g., *"What is the avg process duration of payments with pattern* $P_Z = O, P, Q, R$ *by each organization unit "*. But, as we don't have information on how the organization is structured it will be of no interest to write such a query example. Therefore, these research results explained how the first research question is addressed where a comprehensive analysis among the variants is given.

This scenario showed that users can perform *Roll-up* and *Drill-down* operations to different hierarchy levels across different variants as demonstrated in Figure 8.2. In this figure user can drill-down (view data by increasing the level of details) from a generic process(either G*P_ReceiveInv*) or *GP_Pay* to a generic activity and then from a generic activity to specific elementary activities from the concrete variants. Or, apply roll-up operation in the opposite direction. These results can be interpreted to confirm that the second research question is addressed in consolidating variants into a dimension hierarchy where typical OLAP operations can be performed.

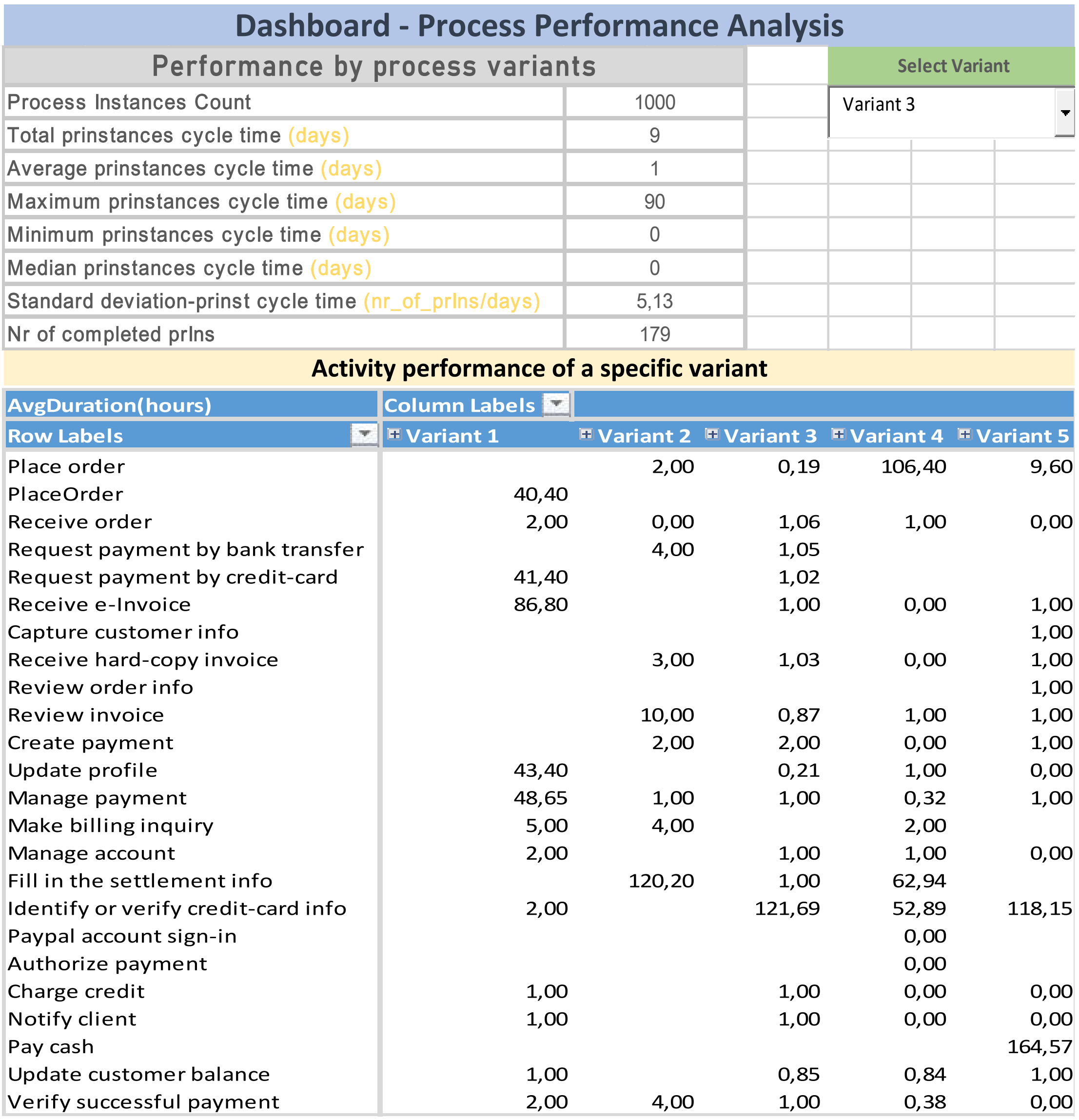


Y-Q-M-D	All

PatternAvgDuration(hours)	Column Labels				
Row Labels	Variant1	Variant2	Variant3	Variant4	Variant5
⊟process-BP5148					
⊟1					
⊟GA1: Pre-request payment					
Request payment by bank transfer		4,00	1,05		
Request payment by credit-card	41,40		1,02		
Receive e-Invoice	86,80		1,00		
Receive hard-copy invoice		3,00	1,03		
Review invoice		10,00	0,87		
Create payment		2,00	2,00		
Update profile	43,40		0,21		
Manage payment	48,65	1,00	1,00		
Make billing inquiry	5,00	4,00			
Manage account	2,00		1,00		
Fill in the settlement info		120,20	1,00		
Identify or verify credit-card info	2,00		121,69		
Charge credit	1,00		1,00		
Notify client	1,00		1,00		
Update customer balance	1,00		0,85		
Verify successful payment	2,00	4,00	1,00		
⊞GA2: Customer info					15,29
⊟2					
⊟GA3: Invoice type					
Receive e-Invoice	86,80		1,00	0,00	1,00
Receive hard-copy invoice		3,00	1,03	0,00	1,00
Review invoice		10,00	0,87	1,00	1,00
Create payment		2,00	2,00	0,00	1,00
Update profile	43,40		0,21	1,00	0,00
Manage payment	48,65	1,00	1,00	0,32	1,00
Make billing inquiry	5,00	4,00		2,00	
Manage account	2,00		1,00	1,00	0,00
Fill in the settlement info		120,20	1,00	62,94	
Identify or verify credit-card info	2,00		121,69	52,89	118,15
Paypal account sign-in				0,00	
Authorize payment				0,00	
Charge credit	1,00		1,00	0,00	0,00
Notify client	1,00		1,00	0,00	0,00
Pay cash					164,57
Update customer balance	1,00		0,85	0,84	1,00
Verify successful payment	2,00	4,00	1,00	0,38	0,00
⊟process-BP5149					
⊟3					
⊟GA4: Payment method					
Fill in the settlement info		120,20	1,00	62,94	
Identify or verify credit-card info	2,00		121,69	52,89	118,15
Paypal account sign-in				0,00	
Authorize payment				0,00	
Charge credit	1,00		1,00	0,00	0,00
Notify client	1,00		1,00	0,00	0,00
Pay cash					164,57
Update customer balance	1,00		0,85	0,84	1,00
Verify successful payment	2,00	4,00	1,00	0,38	0,00

Figure 8.2: Roll-up and Drill-down OLAP operations

Moreover, our approach offers as well the possibility to provide analysis results on a particular process or particular activity as typical BPMS offer. Here we show another dashboard as displayed in Figure 8.3



Dashboard - Process Performance Analysis

Performance by process variants

Select Variant: Variant 3

Performance by process variants	
Process Instances Count	1000
Total prinstances cycle time (days)	9
Average prinstances cycle time (days)	1
Maximum prinstances cycle time (days)	90
Minimum prinstances cycle time (days)	0
Median prinstances cycle time (days)	0
Standard deviation-prinst cycle time (nr_of_prIns/days)	5,13
Nr of completed prIns	179

Activity performance of a specific variant

AvgDuration(hours) — Column Labels

Row Labels	Variant 1	Variant 2	Variant 3	Variant 4	Variant 5
Place order		2,00	0,19	106,40	9,60
PlaceOrder	40,40				
Receive order	2,00	0,00	1,06	1,00	0,00
Request payment by bank transfer		4,00	1,05		
Request payment by credit-card	41,40		1,02		
Receive e-Invoice	86,80		1,00	0,00	1,00
Capture customer info					1,00
Receive hard-copy invoice		3,00	1,03	0,00	1,00
Review order info					1,00
Review invoice		10,00	0,87	1,00	1,00
Create payment		2,00	2,00	0,00	1,00
Update profile	43,40		0,21	1,00	0,00
Manage payment	48,65	1,00	1,00	0,32	1,00
Make billing inquiry	5,00	4,00		2,00	
Manage account	2,00		1,00	1,00	0,00
Fill in the settlement info		120,20	1,00	62,94	
Identify or verify credit-card info	2,00		121,69	52,89	118,15
Paypal account sign-in				0,00	
Authorize payment				0,00	
Charge credit	1,00		1,00	0,00	0,00
Notify client	1,00		1,00	0,00	0,00
Pay cash					164,57
Update customer balance	1,00		0,85	0,84	1,00
Verify successful payment	2,00	4,00	1,00	0,38	0,00

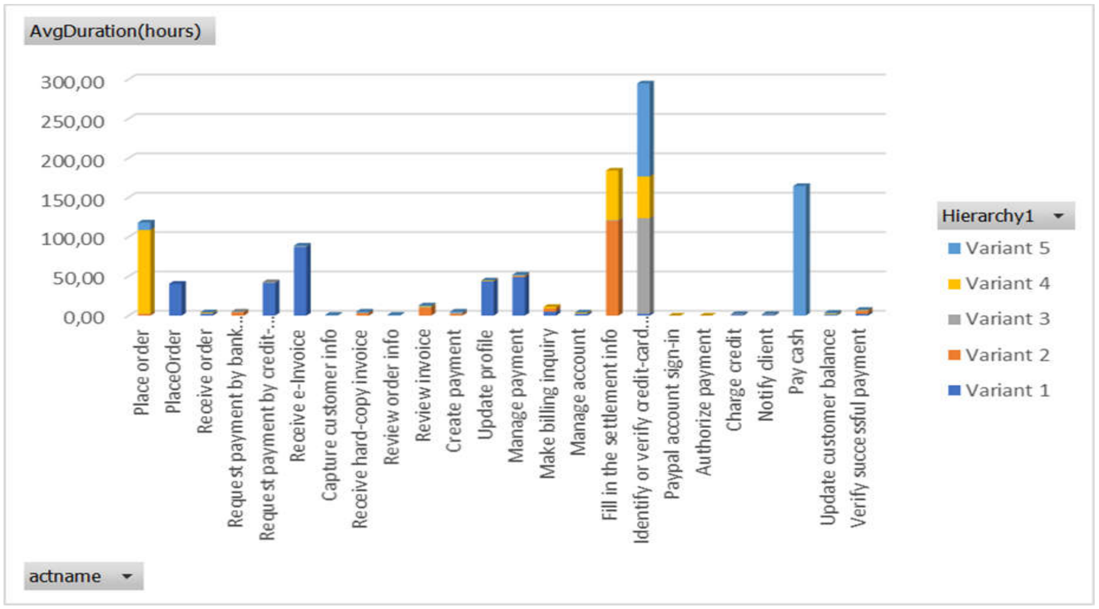
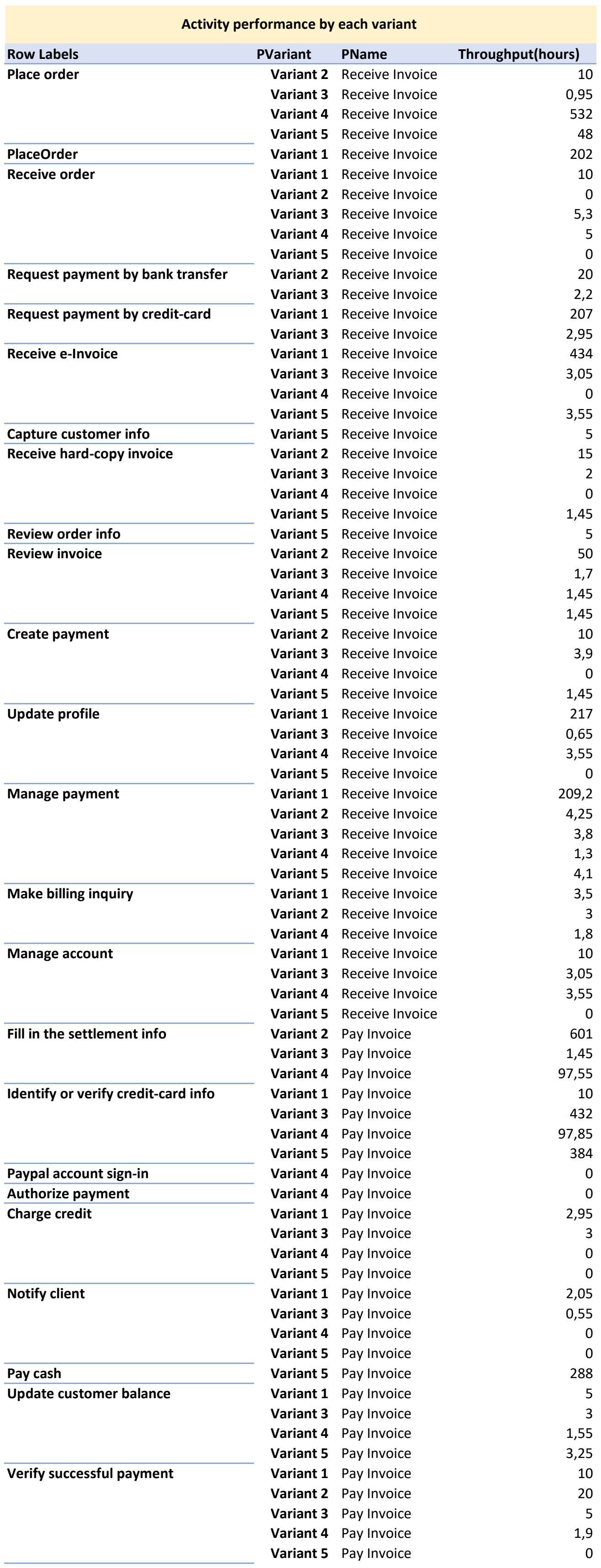


Figure 8.3: Dashboard of process performance analysis

Instead in Figure 8.4 we show duration in hours(Throughput) of each elementary activity across different processes so user can easily compare between similar activities.



Activity performance by each variant

Row Labels	PVariant	PName	Throughput(hours)
Place order	**Variant 2**	Receive Invoice	10
	Variant 3	Receive Invoice	0,95
	Variant 4	Receive Invoice	532
	Variant 5	Receive Invoice	48
PlaceOrder	**Variant 1**	Receive Invoice	202
Receive order	**Variant 1**	Receive Invoice	10
	Variant 2	Receive Invoice	0
	Variant 3	Receive Invoice	5,3
	Variant 4	Receive Invoice	5
	Variant 5	Receive Invoice	0
Request payment by bank transfer	**Variant 2**	Receive Invoice	20
	Variant 3	Receive Invoice	2,2
Request payment by credit-card	**Variant 1**	Receive Invoice	207
	Variant 3	Receive Invoice	2,95
Receive e-Invoice	**Variant 1**	Receive Invoice	434
	Variant 3	Receive Invoice	3,05
	Variant 4	Receive Invoice	0
	Variant 5	Receive Invoice	3,55
Capture customer info	**Variant 5**	Receive Invoice	5
Receive hard-copy invoice	**Variant 2**	Receive Invoice	15
	Variant 3	Receive Invoice	2
	Variant 4	Receive Invoice	0
	Variant 5	Receive Invoice	1,45
Review order info	**Variant 5**	Receive Invoice	5
Review invoice	**Variant 2**	Receive Invoice	50
	Variant 3	Receive Invoice	1,7
	Variant 4	Receive Invoice	1,45
	Variant 5	Receive Invoice	1,45
Create payment	**Variant 2**	Receive Invoice	10
	Variant 3	Receive Invoice	3,9
	Variant 4	Receive Invoice	0
	Variant 5	Receive Invoice	1,45
Update profile	**Variant 1**	Receive Invoice	217
	Variant 3	Receive Invoice	0,65
	Variant 4	Receive Invoice	3,55
	Variant 5	Receive Invoice	0
Manage payment	**Variant 1**	Receive Invoice	209,2
	Variant 2	Receive Invoice	4,25
	Variant 3	Receive Invoice	3,8
	Variant 4	Receive Invoice	1,3
	Variant 5	Receive Invoice	4,1
Make billing inquiry	**Variant 1**	Receive Invoice	3,5
	Variant 2	Receive Invoice	3
	Variant 4	Receive Invoice	1,8
Manage account	**Variant 1**	Receive Invoice	10
	Variant 3	Receive Invoice	3,05
	Variant 4	Receive Invoice	3,55
	Variant 5	Receive Invoice	0
Fill in the settlement info	**Variant 2**	Pay Invoice	601
	Variant 3	Pay Invoice	1,45
	Variant 4	Pay Invoice	97,55
Identify or verify credit-card info	**Variant 1**	Pay Invoice	10
	Variant 3	Pay Invoice	432
	Variant 4	Pay Invoice	97,85
	Variant 5	Pay Invoice	384
Paypal account sign-in	**Variant 4**	Pay Invoice	0
Authorize payment	**Variant 4**	Pay Invoice	0
Charge credit	**Variant 1**	Pay Invoice	2,95
	Variant 3	Pay Invoice	3
	Variant 4	Pay Invoice	0
	Variant 5	Pay Invoice	0
Notify client	**Variant 1**	Pay Invoice	2,05
	Variant 3	Pay Invoice	0,55
	Variant 4	Pay Invoice	0
	Variant 5	Pay Invoice	0
Pay cash	**Variant 5**	Pay Invoice	288
Update customer balance	**Variant 1**	Pay Invoice	5
	Variant 3	Pay Invoice	3
	Variant 4	Pay Invoice	1,55
	Variant 5	Pay Invoice	3,25
Verify successful payment	**Variant 1**	Pay Invoice	10
	Variant 2	Pay Invoice	20
	Variant 3	Pay Invoice	5
	Variant 4	Pay Invoice	1,9
	Variant 5	Pay Invoice	0

Figure 8.4: Info about each activity of different process variants

In the following we show process variant and performance analysis considering the real-life event logs of building permit applications.

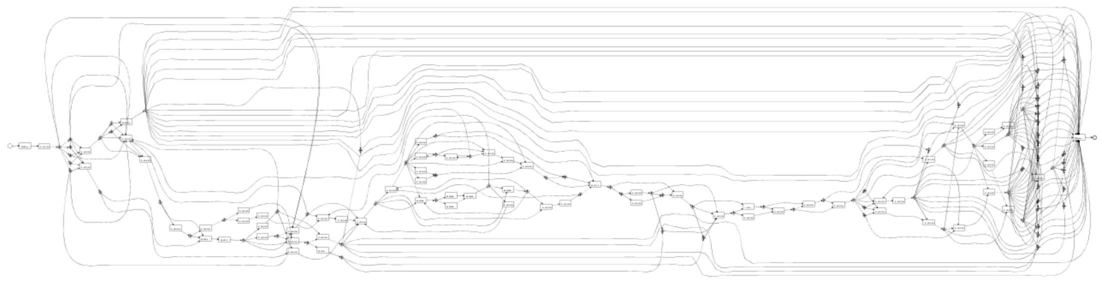
8.2 Case study 2: Real life event logs

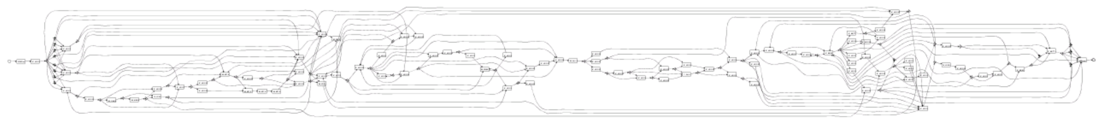
In this second case study we consider real-life event logs published in the repository of the Technical University of Eindhoven. They represent real event logs from 5 Dutch municipalities of 4TU-Center for Research Data repository. [2]
This data contains all building permit applications over a period of approximately four years. There are many different activities present, denoted by both codes (attribute concept:name) and labels, both in Dutch (attribute *taskNameNL*) and in English (attribute *taskNameEN*). The cases in the log contain information on the main application as well as objection procedures in various stages. Furthermore, information is available about the resource that carried out the task and on the cost of the application (attribute *SUMleges*) (Dongen and Boudewijn).

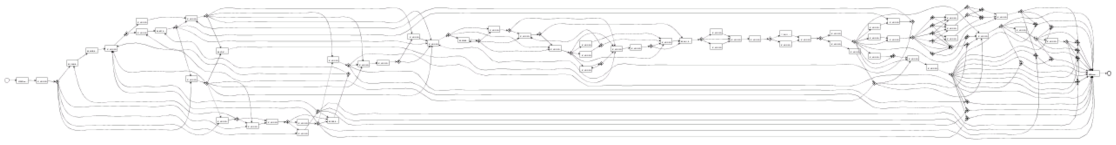
To mine a process model from each of the full logs yields to spaghetti-like models that are hard to interpret and compare as represented in Figure 8.5.

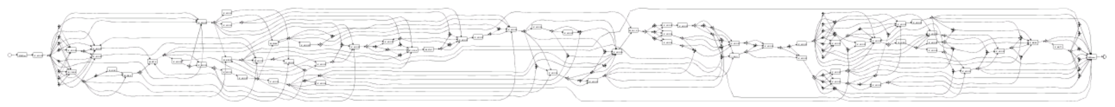
The processes in the five municipalities should be identical, but may differ slightly. Especially when changes are made to procedures, rules or regulations the time at which these changes are pushed into the five municipalities may differ.
To obtain readable process models we filtered the event logs using heuristic miner discovery algorithms using *PROM* tool.

[2]https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1


(a) Municipality 1


(b) Municipality 2


(c) Municipality 3


(d) Municipality 4

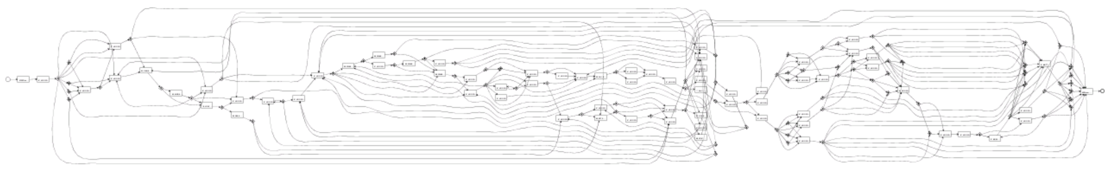

(e) Municipality 5

Figure 8.5: Process models mined from 5 event logs

We give a brief description of this case study as follows:

In the Netherlands a citizen or an enterprise needs a permit or other approval for a variety of activities that may have an impact on the environment or the use of land, e. g., a new building, demolishing a building, fire safety measures in a building, etc. We are interested only in the building permit applications. To apply for a building permit in all five municipalities there are two kind of regulations where two possible procedures are distinguished:

- regular procedure (Dutch: reguliere procedure)
- extensive procedure (Dutch: uitgebreide procedure)

Further, we annotate the commonalities and the differences as variation points among the five variants as follows:
We highlight common activities with a gray color. We present each of the variants in separate figures and describe the step activities of the building permit application process among the different variants. Every process variant starts with activity *register submission date request* as shown in variant 1 Figure 8.7. How they handle the submission differs in each of the variants because of the different activities involved with respective control flows. This marks the first variation point identified in our *reference process model* as displayed in Figure 8.13. In this reference process model we assign each activity a letter preceding the activity label. In variant 1 as shown in Figure 8.7 the following activities are performed:

- activity *A: send confirmation receipt* in parallel with *B: enter senddate acknowledgement*, i. e. A || B or
- activity *A: send confirmation receipt* in parallel with both *B: enter senddate acknowledgement* and *D: enter senddate procedure confirmation*, i. e. A || B || D or
- activity *C: phase application received* in parallel with both *D: enter senddate procedure confirmation* and *B: enter senddate acknowledgement*, i. e. C || D || B or
- activity *C: phase application received* in parallel with *B: enter senddate acknowledgement*, i. e. B || C or
- activity *C: phase application received* in parallel with *D: enter senddate procedure confirmation*, i. e. C || D

The second variation on all five variants occurs on the confirmation procedure part. For instance, in variant 1 either activity *F: send confirmation receipt finalize* or *send procedure confirmation* is performed. Based on the submitted documents the application may be cancelled or followed by *H: registration date publication* activity that publish the registered date of the permit application. If there any incorrect or missing information in it then activity *J: procedure change* is performed followed by two other parallel activities *L: send letter in progress*(to notify stakeholders for their submitted application) and *K: procedure change finalize*(final changes are stored).
Afterwards, another similar activity *N: article 34 WABO applies* in all the five variants follows, where WABO (Wet Algemene Bepalingen Omgevingsrecht) procedure allows that multiple permits related to one project to bundled into one environmental permit. The

application is sent to the competent authority, who determines whether the regular or extensive procedure needs to be followed. We give two line charts displayed in Figure 8.6 to show the differences on the number of these procedures and subprocess instances in order to have a better understanding of these issues.

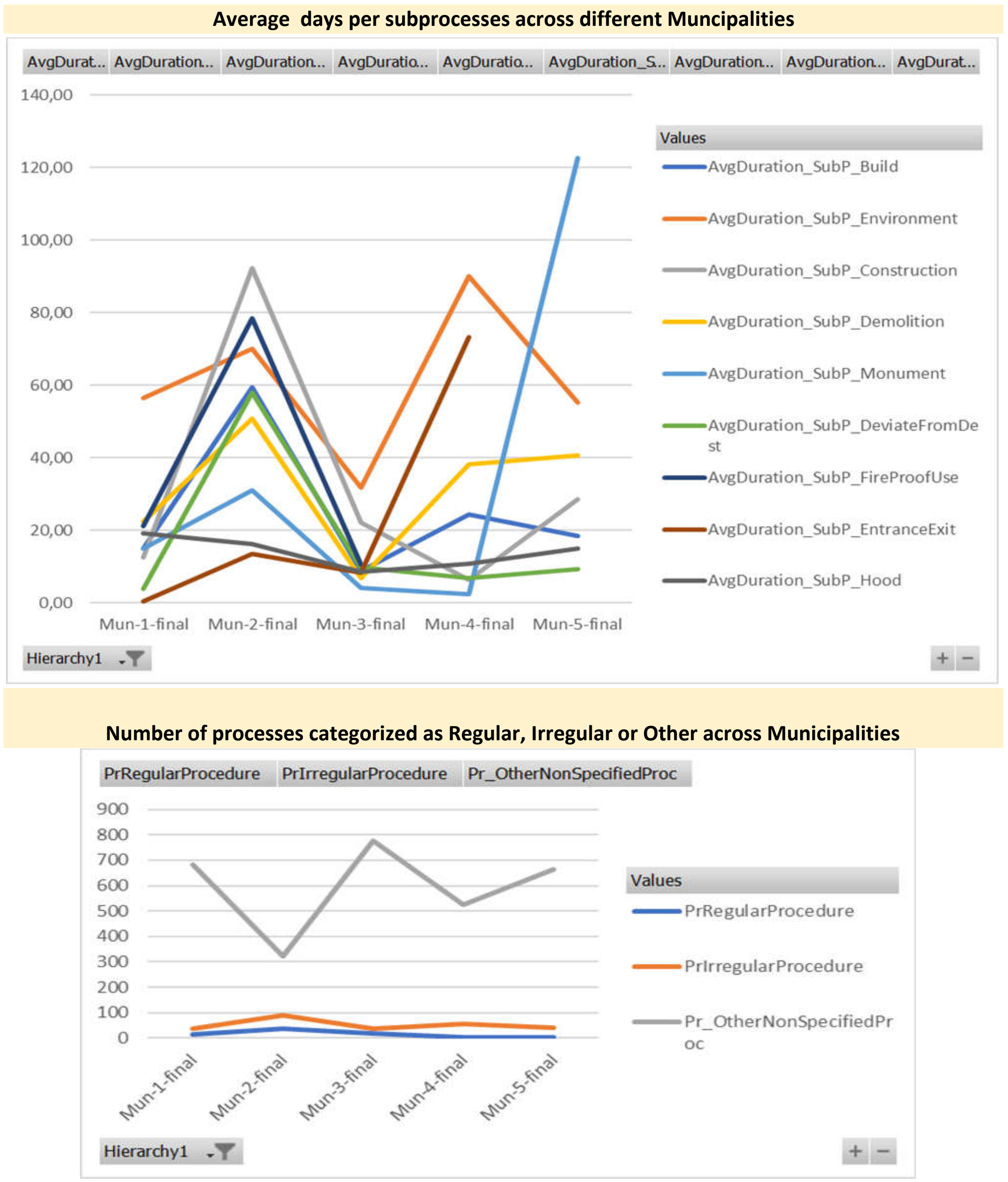


Figure 8.6: Process procedures and sub instances across Municipalities

As seen from the charts above *Mun-3* has the greatest number of *OtherNonSpecified* procedures whereas *Mun-2* the lowest number. Instead for *regular* and *irregular* proce-

dures the line is almost linear across all municipalities with a slight exception of *Mun-2* that counts a greater number of *irregular* procedures. Concerning average duration in days per each sub process clustered on the logs (as different types of permits) from the line chart we can infer that *Mun-2* and *Mun-4* need more time to complete a permit application process. The competent authority evaluates both the completeness and content of all subprocesses of the permit application. Once all subprocesses are handled and the applicable procedure is confirmed, the assessment phase can start.
Accordingly, we mark this phase as our third and last variation point in our reference process model. In each of the variants three activities are performed but slightly in a different order: *O: assessment of content completed* , *P: phase advice known*(a second phase starts in case advices are needed from the advisory board) and *Q:grounds for refusal*(documents the grounds in case of refusal). In variant 1, if the assessment of the content results complete and positive then activity *enter senddate decision environmental permit* is performed and indicates the end of the application process. Otherwise, the flow shifts to activities *P: phase advice known* and *Q:grounds for refusal* where as mentioned another phase starts where the advisory board should be asked to get their opinions. This goes through activity *Q:ask stakeholders views* followed by activity *W: by law* which is used when time limits is not exceeded and the procedure does not comply with the procedure that was originally designed by the ministry.
Then, activities *V: creating environmental permit decision* and *Z: decision date prior to decision* are performed in parallel. In case of positive decision from this second phase the process ends by executing the last activity *R: enter senddate decision environmental permit* otherwise activity *Z: decision date prior to decision* is followed and the process ends by executing the last activity *T: transcript decision environmental permit to stakeholders* and the decision is sent to all stakeholders.

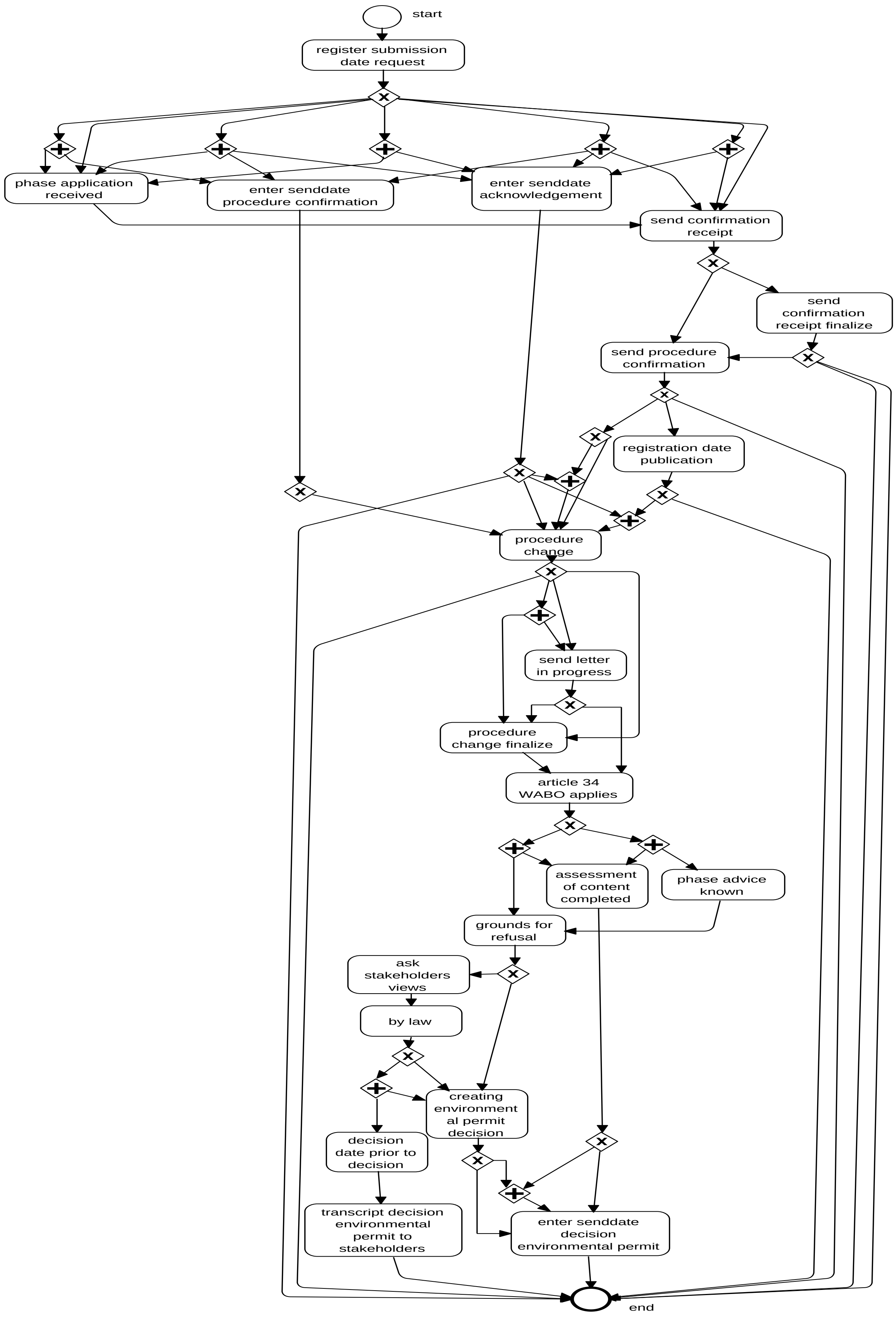

start
register submission date request
phase application received
enter senddate procedure confirmation
enter senddate acknowledgement
send confirmation receipt
send confirmation receipt finalize
send procedure confirmation
registration date publication
procedure change
send letter in progress
procedure change finalize
article 34 WABO applies
assessment of content completed
phase advice known
grounds for refusal
ask stakeholders views
by law
creating environmental permit decision
decision date prior to decision
transcript decision environmental permit to stakeholders
enter senddate decision environmental permit
end

Figure 8.7: The full process model of building permit application of Municipality 1

Whereas in variant 2 as shown in Figure 8.8 the following activities are performed:

- activity *A: send confirmation receipt* in parallel with *C: phase application received*, i. e. A || C or
- activity *A: send confirmation receipt* in parallel with both *B: enter senddate acknowledgement* and *D: enter senddate procedure confirmation*, i. e. A || B || D or
- activity *A: send confirmation receipt* in parallel with *B: enter senddate acknowledgement*, i. e. A || B or
- activity *A: send confirmation receipt* in parallel with both *B: enter senddate acknowledgement* and *C: phase application received*, i. e. A || B || C

In contrast with variant 1, variant 2 in the confirmation procedure part constitutes of other two activities: *I: terminate on request* and *AC: register deadline.* We didn't annotate them as another variation point because they don't influence the flow of rest of activities in the process itself.
Moreover, during the assessment of content variant 2 performs two other activities *AB: objection lodged against decision* and *Y: set phase: phase permitting irrevocable* to handle situations where the application can be disputed within the municipality itself without involving an appeal case in court. Afterwards, takes the final decision which is irrevocable.

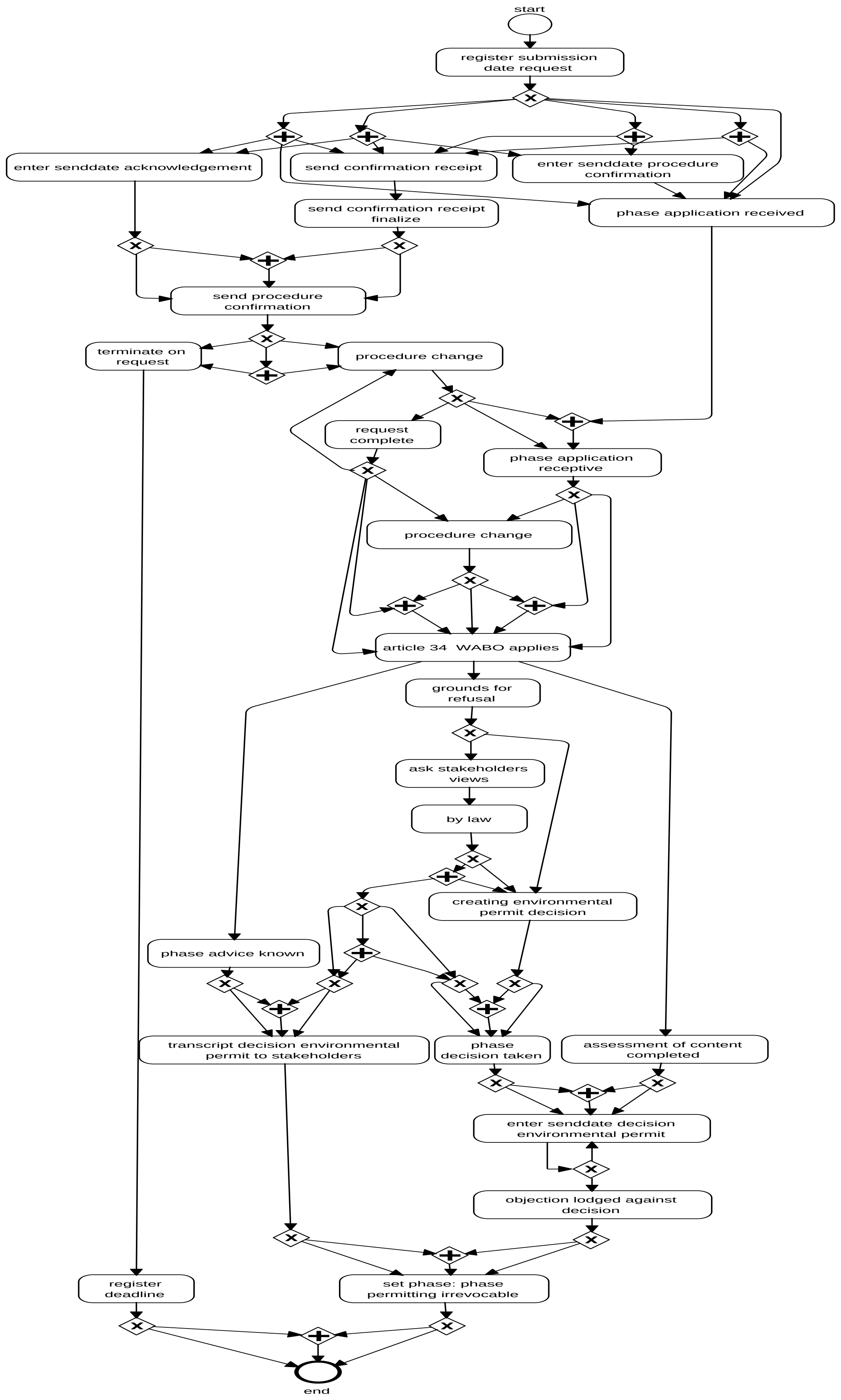

start
register submission date request
enter senddate acknowledgement
send confirmation receipt
enter senddate procedure confirmation
send confirmation receipt finalize
phase application received
send procedure confirmation
terminate on request
procedure change
request complete
phase application receptive
procedure change
article 34 WABO applies
grounds for refusal
ask stakeholders views
by law
creating environmental permit decision
phase advice known
transcript decision environmental permit to stakeholders
phase decision taken
assessment of content completed
enter senddate decision environmental permit
objection lodged against decision
register deadline
set phase: phase permitting irrevocable
end

Figure 8.8: The full process model of building permit application of Municipality 2

In variant 3 as shown in Figure 8.9 activity *E: applicant is stakeholder* is performed; whereas activity *D: enter senddate procedure confirmation* is not. During the submission phase the following activities are performed:

- activity *A: send confirmation receipt* in parallel with *B: enter senddate acknowledgement*, i. e. A || B or
- activity *C: phase application received* in parallel with *B: enter senddate acknowledgement*, i. e. B || C or
- activity *C: phase application received* in parallel with both *B: enter senddate acknowledgement* and *E: applicant is stakeholder*, i. e. B || C || E or
- activity *A: send confirmation receipt* in parallel with both *B: enter senddate acknowledgement* and *E: applicant is stakeholder*, i. e. A || B || E or
- activity *B: enter senddate acknowledgement* in parallel with *E: applicant is stakeholder* , i. e. B || E or
- activity *A: send confirmation receipt* in parallel with *E: applicant is stakeholder* , i. e. A || E

Moreover, activity *procedure change finalize* is performed after considering the *Q: grounds for refusal* activity as well followed by activity *N: article 34 WABO applies.* In this variant, activities are more evenly distributed and number of gateways is smaller comparing to other variants thus affecting the flow of the work.

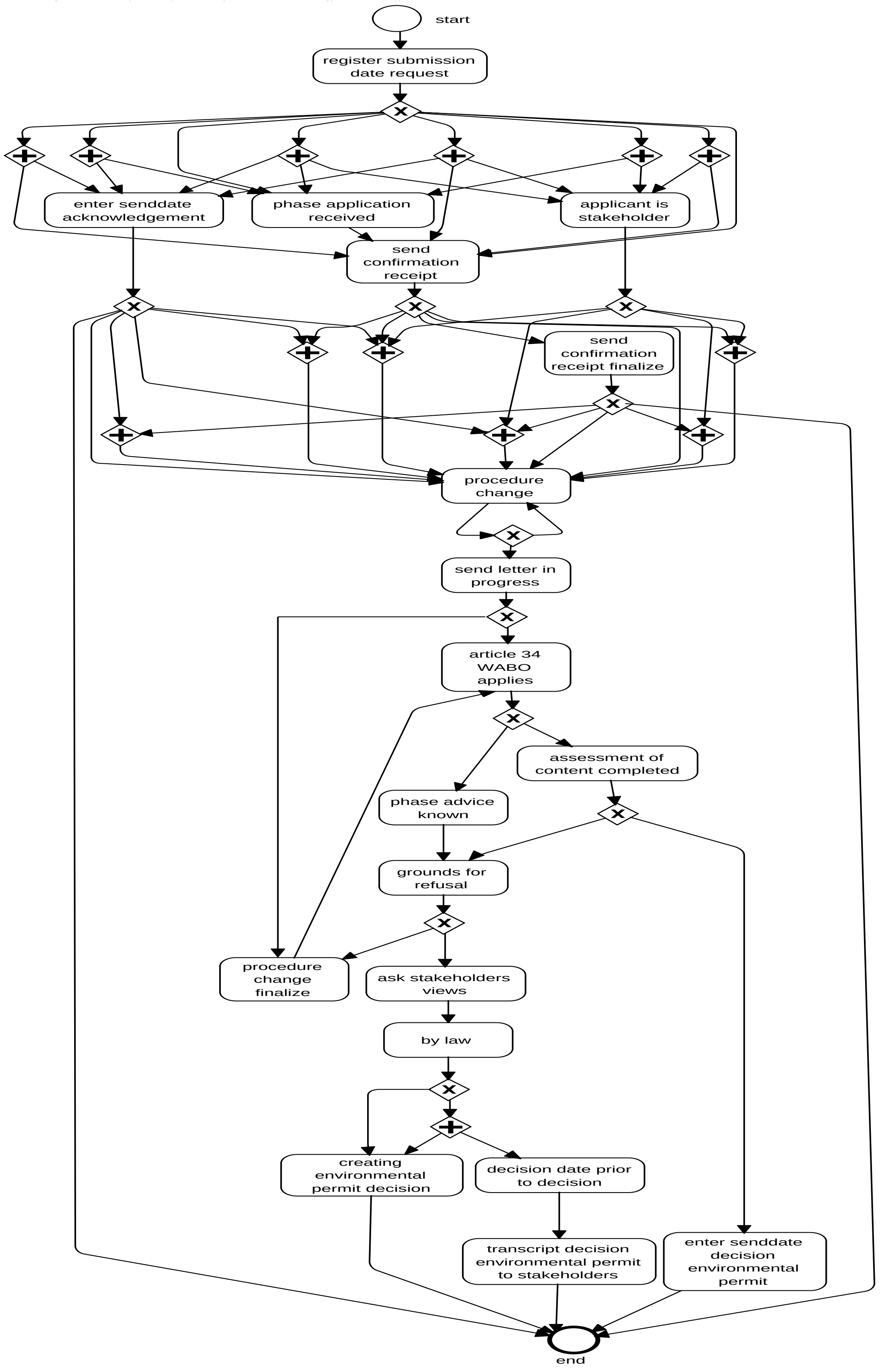

start
register submission date request
enter senddate acknowledgement
phase application received
applicant is stakeholder
send confirmation receipt
send confirmation receipt finalize
procedure change
send letter in progress
article 34 WABO applies
assessment of content completed
phase advice known
grounds for refusal
procedure change finalize
ask stakeholders views
by law
creating environmental permit decision
decision date prior to decision
transcript decision environmental permit to stakeholders
enter senddate decision environmental permit
end

Figure 8.9: The full process model of building permit application of Municipality 3

In variant 4 as shown in Figure 8.10 the following activities are performed in parallel: Similar to variant 2, this variant as well handles the objections against decisions by performing activity *AB: objection lodged against decision* to handle cases where a claim is appealed against the taken decision. In opposite to all other four variants, activity *G: send procedure confirmation* marks the end of the process which most probably means that is used to handle only cases when the permit application is rejected. Obviously, as this is a specialized activity of *GA2* (defined in Figure 8.12) the number of activities after substitution operations (direct and non-direct) will be performed will contain only this activity and no other ones.

- activity *A: send confirmation receipt* in parallel with *E: applicant is stakeholder*, i. e. A || E or
- activity *A: send confirmation receipt* in parallel with *C: phase application received*, i. e. A || C or
- activity *A: send confirmation receipt* in parallel with *C: phase application received* and *D: enter senddate procedure confirmation* and *E: applicant is stakeholder*, i. e. A || C || D || E or
- activity *A: send confirmation receipt* in parallel with both *C: phase application received* and *D: enter senddate procedure confirmation*, i. e. A || C || D

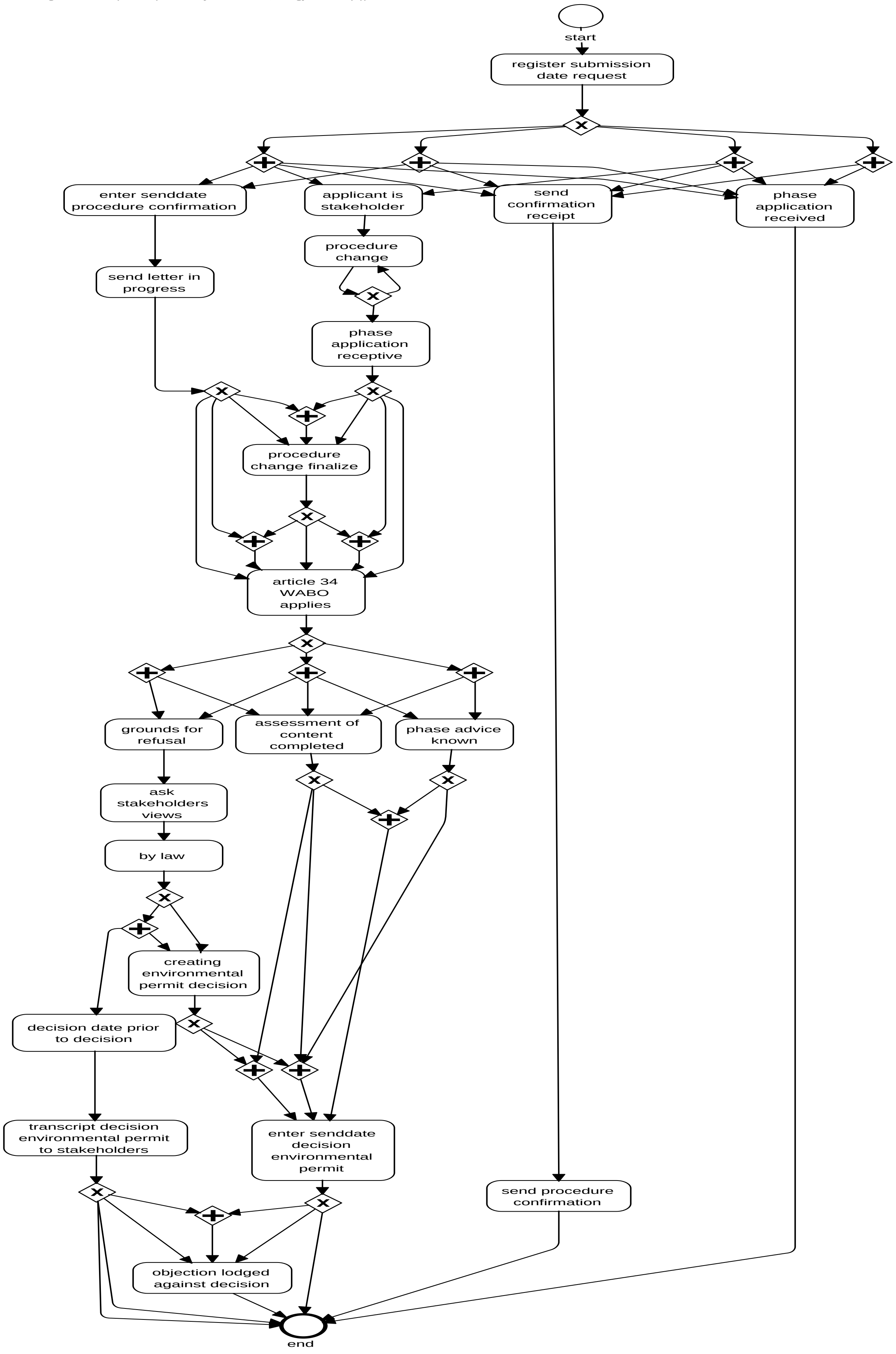

start
register submission date request
enter senddate procedure confirmation
applicant is stakeholder
send confirmation receipt
phase application received
procedure change
send letter in progress
phase application receptive
procedure change finalize
article 34 WABO applies
grounds for refusal
assessment of content completed
phase advice known
ask stakeholders views
by law
creating environmental permit decision
decision date prior to decision
transcript decision environmental permit to stakeholders
enter senddate decision environmental permit
send procedure confirmation
objection lodged against decision
end

Figure 8.10: The full process model of building permit application of Municipality 4

Finally, in variant 5 as shown in Figure 8.11 for the submission phase are performed the following activities:

- activity *A: send confirmation receipt* in parallel with both *B: enter senddate acknowledgement* and *D: enter senddate procedure confirmation*, i. e. A || B || D or
- activity *A: send confirmation receipt* in parallel with both *B: enter senddate acknowledgement* and *C: phase application received* , i. e. A || B || C or
- activity *A: send confirmation receipt* in parallel with *D: enter senddate procedure confirmation*, i. e. A || D or
- activity *A: send confirmation receipt* in parallel with *C: phase application received*, i. e. A || C

Similar to variant 2, this variant handle as well the claim appeals using both activities *AB: objection lodged against decision* and *Y: set phase: phase permitting irrevocable.* In the last variation point, similar activities such as: *V: creating environmental permit decision*, *R: enter senddate decision environmental permit*, *U: ask stakeholders views* and *W: by law* are shared among the different variants.

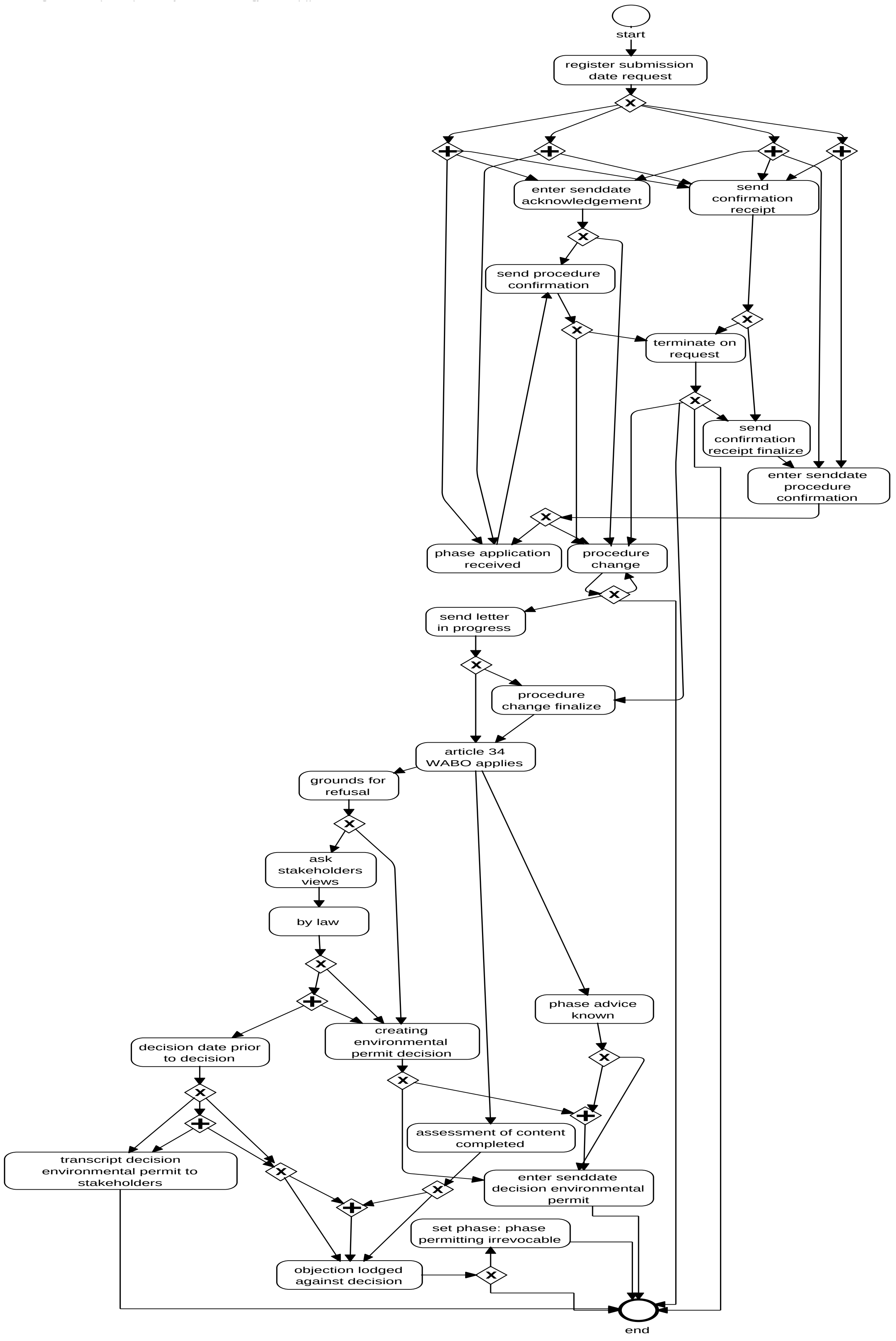

start
register submission date request
enter senddate acknowledgement
send confirmation receipt
send procedure confirmation
terminate on request
send confirmation receipt finalize
enter senddate procedure confirmation
phase application received
procedure change
send letter in progress
procedure change finalize
article 34 WABO applies
grounds for refusal
ask stakeholders views
by law
phase advice known
creating environmental permit decision
decision date prior to decision
assessment of content completed
transcript decision environmental permit to stakeholders
enter senddate decision environmental permit
set phase: phase permitting irrevocable
objection lodged against decision
end

Figure 8.11: The full process model of building permit application of Municipality 5

After giving the brief description on all the five process variants we explain in the next pages how we model the reference process model as a generic process to capture all the variants by defining three GAs.

As mentioned above, after annotating differences among the five process variants of the building permit application we manage to identify three variation points so-called Generic Activities (GA) highlighted with a yellow color Figure 8.12. Three GAs identified are: *GA1: Submission type*, *GA2: Confirmation procedure* and *GA3: Assessment of submission content*. Specialized activities of *GA1* with green highlighted color are six customized subprocesses named:

- *SP1: A,B,(C,D,E)*,
- *SP2: B,C,(D,E)*,
- *SP3: C,D*,
- *SP4: A,C,(D,E)*,
- *SP5: A,D,(C,E)* and
- *SP6: E,(A,C)*

We reduced the number of these customized subprocesses to six instead of thirteen based on the total combinations of the involved activities (i. e., A, B, C, D, E) in this phase, in order to make this reference model less complex. Therefore, we capture all different control flows by using AND, XOR split/joins gateways as already depicted in the following figure. Whereas, specialized activities of *GA2* are two elementary activities named: *F: send confirmation receipt finalize* and *G: send procedure confirmation*.
And finally, specialized activities of *GA3* are three customized subprocesses based on the ordering of flow of the relevant activities *O*, *P*, *Q* :

- *SP7: (O,P,Q)*,
- *SP8: O,(P,Q)* and
- *SP9: O,P,Q*

We annotate these specialization relationships by using a stereotype *variant_specialization* in the dashed connector line with an arrow-head directed to the generic activity.

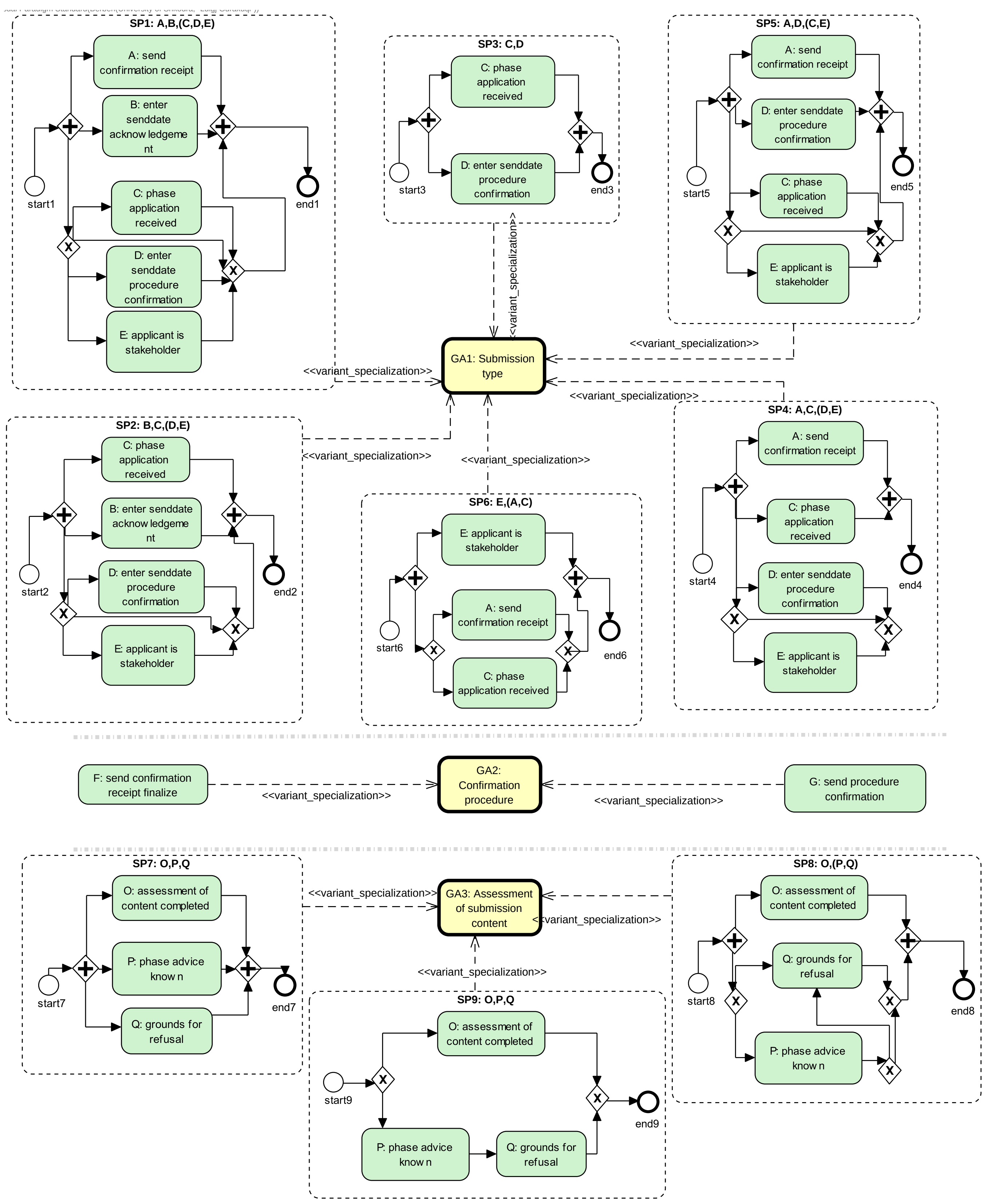

SP1: A,B,(C,D,E)
A: send confirmation receipt
B: enter senddate acknowledgement
C: phase application received
D: enter senddate procedure confirmation
E: applicant is stakeholder
start1
end1
SP3: C,D
start3
end3
SP5: A,D,(C,E)
start5
end5
GA1: Submission type
<<variant_specialization>>
SP2: B,C,(D,E)
start2
end2
SP6: E,(A,C)
start6
end6
SP4: A,C,(D,E)
start4
end4
F: send confirmation receipt finalize
GA2: Confirmation procedure
G: send procedure confirmation
SP7: O,P,Q
O: assessment of content completed
P: phase advice known
Q: grounds for refusal
start7
end7
GA3: Assessment of submission content
SP8: O,(P,Q)
start8
end8
SP9: O,P,Q
start9
end9

Figure 8.12: Generic Activities and the specialized activities and subprocesses

8.2.1 Generic process of building permit application

In this section, we build the generic process with generic activities displayed as follows: We could have built the reference model by modelling only GAs without the other activities that are shared among the five process variants because certainly they are being captured after applying non-direct specializations of each GA based on *Breadth_ First_ Search* procedure.

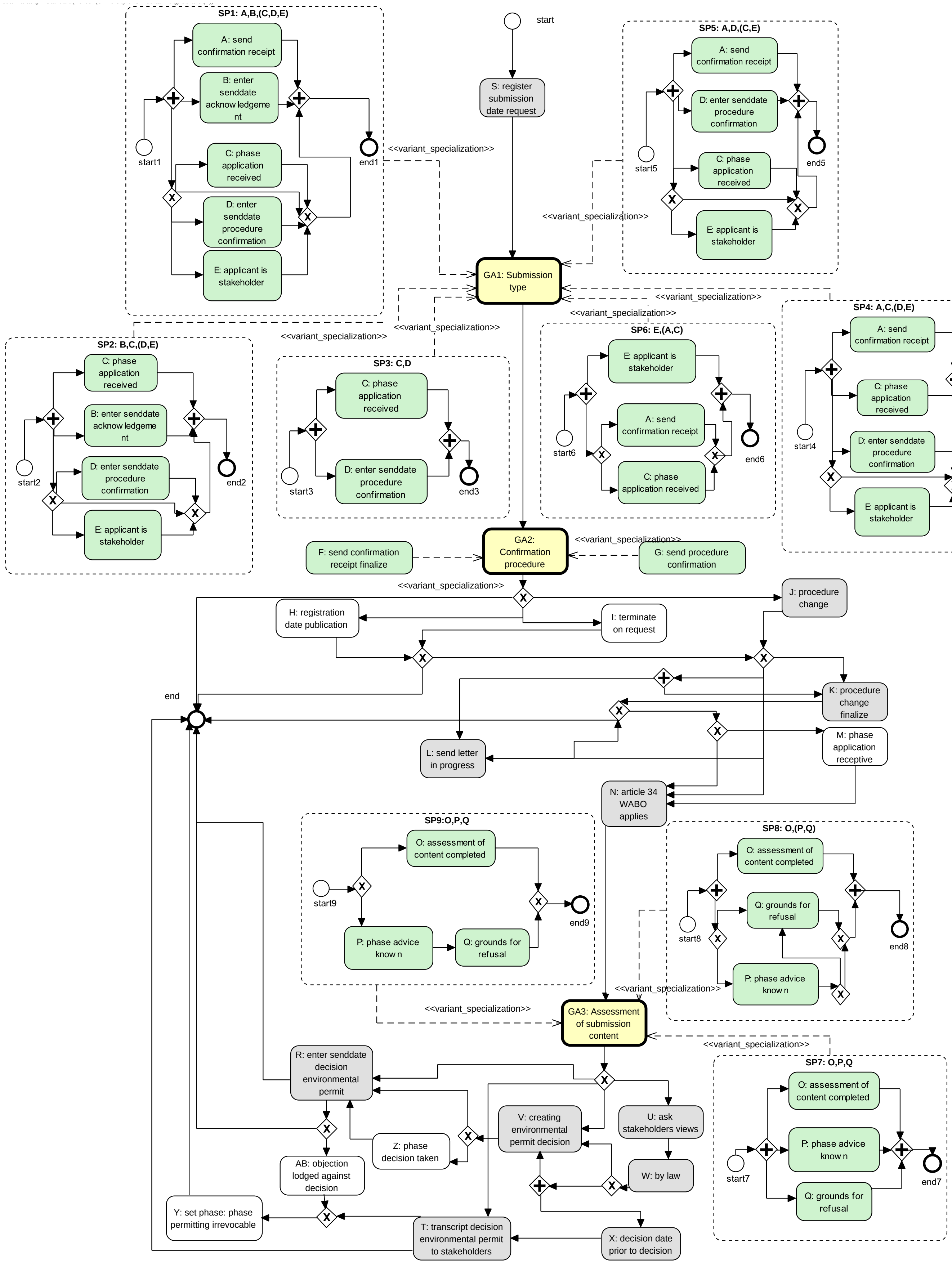

SP1: A,B,(C,D,E)
A: send confirmation receipt
B: enter senddate acknowledgement
C: phase application received
D: enter senddate procedure confirmation
E: applicant is stakeholder
start1
end1
start
S: register submission date request
SP5: A,D,(C,E)
start5
end5
<<variant_specialization>>
GA1: Submission type
SP2: B,C,(D,E)
SP3: C,D
SP4: A,C,(D,E)
SP6: E,(A,C)
start2
end2
start3
end3
start4
end4
start6
end6
GA2: Confirmation procedure
F: send confirmation receipt finalize
G: send procedure confirmation
H: registration date publication
I: terminate on request
J: procedure change
K: procedure change finalize
L: send letter in progress
M: phase application receptive
N: article 34 WABO applies
end
SP9:O,P,Q
SP8: O,(P,Q)
SP7: O,P,Q
O: assessment of content completed
P: phase advice known
Q: grounds for refusal
start9
end9
start8
end8
start7
end7
GA3: Assessment of submission content
R: enter senddate decision environmental permit
U: ask stakeholders views
V: creating environmental permit decision
W: by law
Z: phase decision taken
AB: objection lodged against decision
Y: set phase: phase permitting irrevocable
T: transcript decision environmental permit to stakeholders
X: decision date prior to decision

Figure 8.13: Generic process of building permit application

Whereas in table 8.1 we give a representation of tuples of how specialization relationships between activities and generic activities for this reference process model (contains only one generic processes) is stored in our relational database. Activity name's attribute records task code values such as *G: 01_HOOFD_065_1* (i. e.,)and *F: 01_HOOFD_-030_1* highlighted with a light yellow color in the table below.

Table 8.1: Specialization relationships between activities and generic activities of reference permit process

ActivityName	GenericActivityName	ProcessName
SP1: A,B,(C,D,E)	GA1: Submission type	RefPM_PermitApp
SP2: B,C,(D,E)	GA1: Submission type	RefPM_PermitApp
SP3: C,D	GA1: Submission type	RefPM_PermitApp
SP4: A,C,(D,E)	GA1: Submission type	RefPM_PermitApp
SP5: A,D,(C,E)	GA1: Submission type	RefPM_PermitApp
SP6: E,(A,C)	GA1: Submission type	RefPM_PermitApp
G: 01_HOOFD_065_1	GA2: Confirmation procedure	RefPM_PermitApp
F: 01_HOOFD_030_1	GA2: Confirmation procedure	RefPM_PermitApp
SP7: O,P,Q	GA3: Assessment of submission content	RefPM_PermitApp
SP8: O,(P,Q)	GA3: Assessment of submission content	RefPM_PermitApp
SP9: (O,P),Q	GA3: Assessment of submission content	RefPM_PermitApp

If we bound each specialized activity from the generic processes to an activity of a concrete process we get some other tuples as displayed in Table 8.2(variants 1-3) and Table 8.3(variants 4-5).
In these two tables we show a representation of tuples of how specialization relationships between activities and generic activities for each process variant are retrieved in our relational database. We show both attributes of activity codes and activity names (in English) to make it easy for reader to understand. Of course, in each of the concrete processes i. e., in all five variants these activities are uniquely identified by their *ActId*. We give a full list of all activity code labelled with respective activity names both in English and Dutch language in Appendix Chapter B. Moreover, if we compare these results with the above table in Table 8.1 we see that all specialized subprocesses are bounded with their direct elementary activities that belong to respective variants. The order on which they are executed can be revealed after getting information from respective event logs.

Table 8.2: Specialization relationships between activities and generic activities for each process variant (1-3)

	ActivityCodeName	ActivityEnName	GenericActivity
Process Variant 1 (Mun 1)	01_HOOFD_015	phase application received	GA1: Submission type
	01_HOOFD_020	send confirmation receipt	GA1: Submission type
	01_HOOFD_030_2	enter senddate acknowledgement	GA1: Submission type
	01_HOOFD_065_2	enter senddate procedure confirmation	GA1: Submission type
	01_HOOFD_030_1	send confirmation receipt finalize	GA2: Confirmation procedure
	01_HOOFD_065_1	send procedure confirmation	GA2: Confirmation procedure
	01_HOOFD_370	assessment of content completed	GA3: Assessment of submission content
	01_HOOFD_375	phase advice known	GA3: Assessment of submission content
	01_HOOFD_380	grounds for refusal	GA3: Assessment of submission content
Process Variant 2 (Mun 2)	01_HOOFD_015	phase application received	GA1: Submission type
	01_HOOFD_020	send confirmation receipt	GA1: Submission type
	01_HOOFD_030_2	enter senddate acknowledgement	GA1: Submission type
	01_HOOFD_065_2	enter senddate procedure confirmation	GA1: Submission type
	01_HOOFD_030_1	send confirmation receipt finalize	GA2: Confirmation procedure
	01_HOOFD_065_1	send procedure confirmation	GA2: Confirmation procedure
	01_HOOFD_370	assessment of content completed	GA3: Assessment of submission content
	01_HOOFD_375	phase advice known	GA3: Assessment of submission content
	01_HOOFD_380	grounds for refusal	GA3: Assessment of submission content
Process Variant 3 (Mun 3)	01_HOOFD_015	phase application received	GA1: Submission type
	01_HOOFD_020	send confirmation receipt	GA1: Submission type
	01_HOOFD_030_2	enter senddate acknowledgement	GA1: Submission type
	03_GBH_005	applicant is stakeholder	GA1: Submission type
	01_HOOFD_030_1	send confirmation receipt finalize	GA2: Confirmation procedure
	01_HOOFD_370	assessment of content completed	GA3: Assessment of submission content
	01_HOOFD_375	phase advice known	GA3: Assessment of submission content
	01_HOOFD_380	grounds for refusal	GA3: Assessment of submission content

Table 8.3: Specialization relationships between activities and generic activities for each process variant (4-5)

	ActivityCodeName	ActivityEngName	GenericActivity
Process Variant 4 (Mun 4)	01_HOOFD_015	phase application received	GA1: Submission type
	01_HOOFD_020	send confirmation receipt	GA1: Submission type
	01_HOOFD_065_2	enter senddate procedure confirmation	GA1: Submission type
	03_GBH_005	applicant is stakeholder	GA1: Submission type
	01_HOOFD_065_1	send procedure confirmation	GA2: Confirmation procedure
	01_HOOFD_370	assessment of content completed	GA3: Assessment of submission content
	01_HOOFD_375	phase advice known	GA3: Assessment of submission content
	01_HOOFD_380	grounds for refusal	GA3: Assessment of submission content
Process Variant 5 (Mun 5)	01_HOOFD_015	phase application received	GA1: Submission type
	01_HOOFD_020	send confirmation receipt	GA1: Submission type
	01_HOOFD_030_2	enter senddate acknowledgement	GA1: Submission type
	01_HOOFD_065_2	enter senddate procedure confirmation	GA1: Submission type
	01_HOOFD_030_1	send confirmation receipt finalize	GA2: Confirmation procedure
	01_HOOFD_065_1	send procedure confirmation	GA2: Confirmation procedure
	01_HOOFD_370	assessment of content completed	GA3: Assessment of submission content
	01_HOOFD_375	phase advice known	GA3: Assessment of submission content
	01_HOOFD_380	grounds for refusal	GA3: Assessment of submission content

Then, we derive process variants hierarchy after applying direct and non-direct specializations of GAs from generic processes by executing Algorithm 2. We start by substituting *GA1: Submission type* to get all direct elementary activities from different process variants. Then, we explore all other activities in a breadth first search approach until the very end of the process.

So, we basically define the GA1 as a process pattern that after first activity *S: register submission date request* is enacted it can be followed by any other group of activities in a specific sequential order until the end of the process. In analogy to regular expressions we can translate this pattern in: $S, *$, where $*$ -means 0 or more.

We give a graphical representation of this pattern in Figure 8.14. As displayed in the figure, *GA1* is the highest level of the hierarchy where all activities from the five variants (process models in a shadowed gray color) are consolidated.

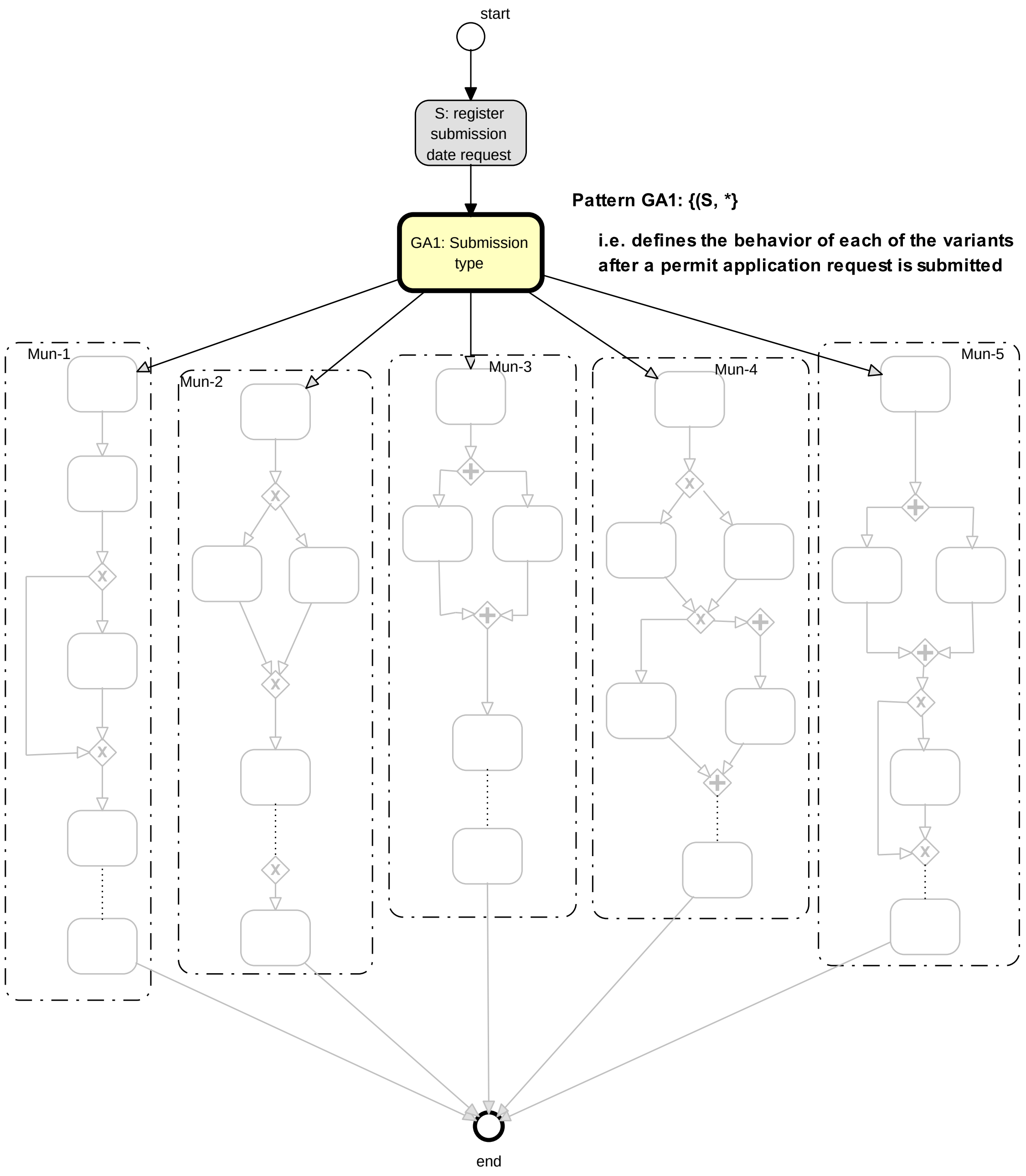

start
S: register submission date request
Pattern GA1: {(S, *}
GA1: Submission type
i.e. defines the behavior of each of the variants after a permit application request is submitted
Mun-1
Mun-2
Mun-3
Mun-4
Mun-5
end

Figure 8.14: Graphical representation of process pattern S, *

In similar way we give a graphical representation of the second pattern in Figure 8.15. *GA2 pattern*: $S, (A, B, C, D, E), *$ means that starts with activity S, followed by activities A, or B, or C, or D, or E, in any order and then followed by more activities up to the end. As displayed in the figure, *GA2* is positioned at a lower level of the hierarchy then *GA1* where specialized activities from this point onwards are consolidated.

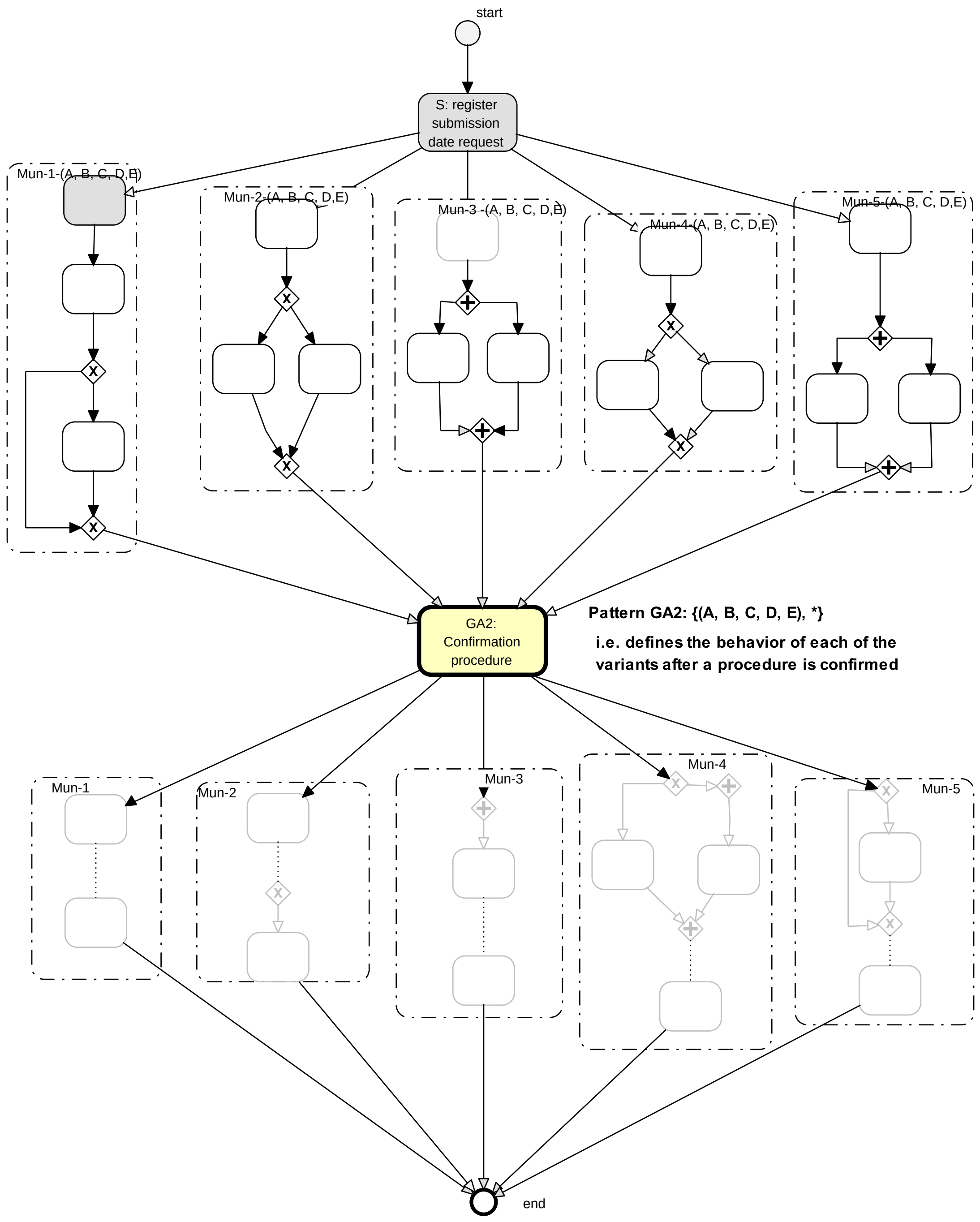

start
S: register submission date request
Mun-1-(A, B, C, D,E)
Mun-2-(A, B, C, D,E)
Mun-3 -(A, B, C, D,E)
Mun-4-(A, B, C, D,E)
Mun-5-(A, B, C, D,E)
GA2: Confirmation procedure
Pattern GA2: {(A, B, C, D, E), *}
i.e. defines the behavior of each of the variants after a procedure is confirmed
Mun-1
Mun-2
Mun-3
Mun-4
Mun-5
end

Figure 8.15: Graphical representation of process pattern S, (A, B, C, D, E), *

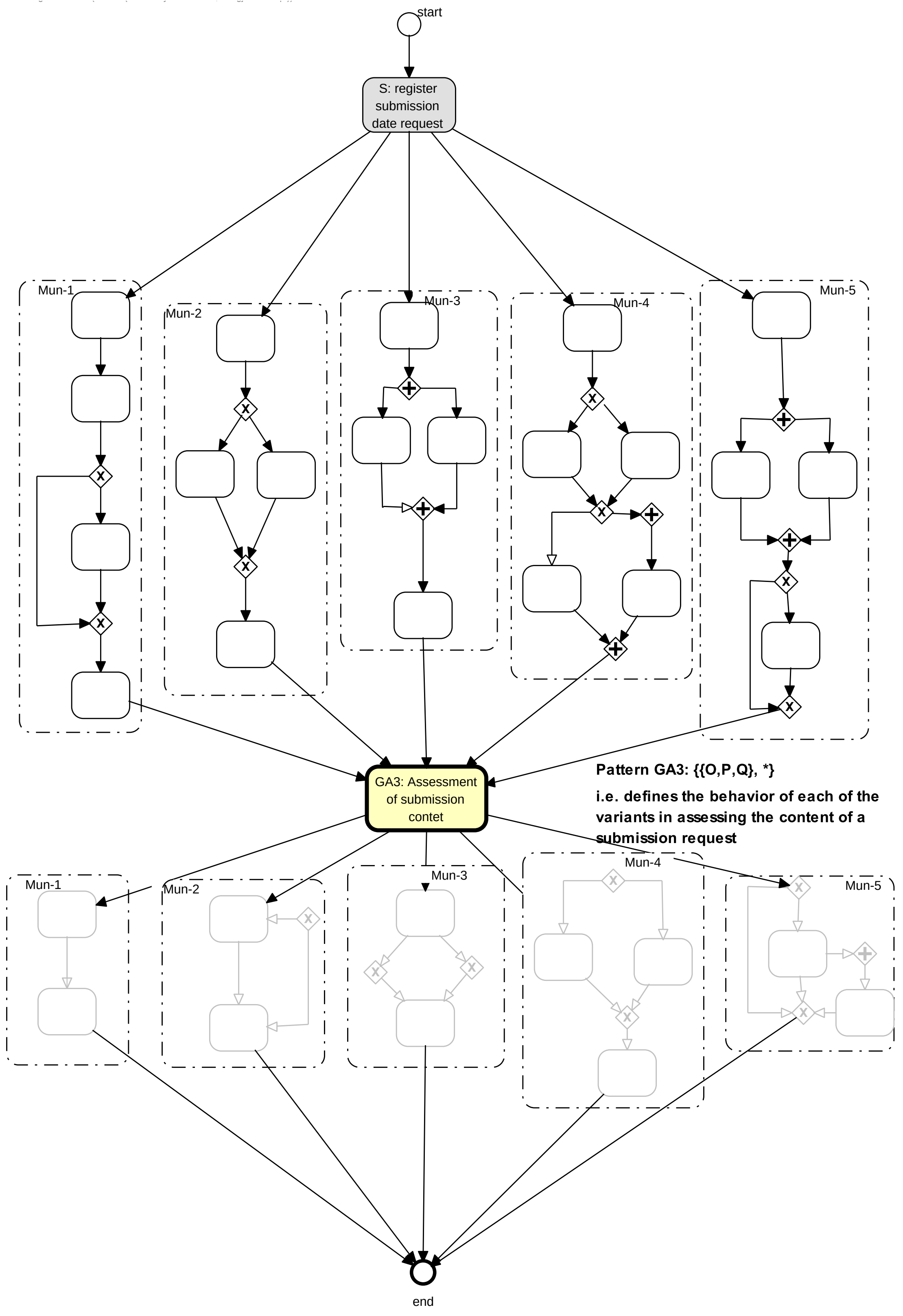

start
S: register submission date request
Mun-1
Mun-2
Mun-3
Mun-4
Mun-5
GA3: Assessment of submission contet
Pattern GA3: {{O,P,Q}, *}
i.e. defines the behavior of each of the variants in assessing the content of a submission request
Mun-1
Mun-2
Mun-3
Mun-4
Mun-5
end

Figure 8.16: Graphical representation of process pattern S, (A, B, C, D, E), (F, G),(J, K, L, M, N), *

And we show the last we give a graphical representation of the third pattern in Figure 8.16. *GA3 pattern*: $S, (A, B, C, D, E), (F, G), (J, K, L, M, N), *$ means that starts with activity S, followed by activities either (A, B, C, D, E) in any order, followed by either F or G, then J, K, L, M, N (in any order) and lastly followed by any other activities up to the end. As displayed in the figure, *GA3* is positioned at the lowest level of the hierarchy where specialized activities from this point onwards are consolidated.

In the following we give an example of process warehouse instances of permit application process.

8.2.2 Process warehouse instances

Before showing some instances of the process warehouse for the building permit application we display an excerpt of the five event logs. Furthermore, we explain some of the adaptations we make to the Figure 5.1 process warehouse schema. In the following Table 8.4 we show an excerpt of the five event logs we parsed and imported to our dB. The *StepInstance* table contains 67.443 rows in total. In contrast with case study 1 with the simulated logs, the event log files of this real life case study contained much more information. For this reason, we populated with records table *cEInstance*(to store output gateway values, e. g., *True* or *False*) and alter table *ProcessIntance* by adding other attributes e. g., *PrInsStatus*, *PrIns_ Parts*, *SubPrIns_ Included* etc. in order to capture all the data.
Another difference between these two cases studies is that in the second case study the event log files contain only "complete" events and not "assign" or "start" events. But in each of the step instances there is also another associated attribute that records the start time of each activity instance.

Table 8.4: An excerpt of event logs from case study 2

StepInsId	StepInsName	StepInsExeTime(s)	StepId	ProcessInstanceId	ProcessId	ProcessName
10951	01_HOOFD_370 (assessment of content completed)	2926	node_79f48ce1-78fa-4842-b53a-0f5e74226bc5	2817543	proc_-1374099955	Mun-1
10953	01_HOOFD_370 (assessment of content completed)	5515	node_79f48ce1-78fa-4842-b53a-0f5e74226bc5	2832460	proc_-1374099955	Mun-1
[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]
18105	01_HOOFD_370 (assessment of content completed)	2638	node_0de3f4e6-d904-47cf-9f27-7bd37c39cc6b	5979313	proc_-1679653381	Mun-2
14067	01_HOOFD_480 (by law)	142	node_80b889db-e4f5-4955-abe7-de3fb291ae08	19914886	proc_-1679653381	Mun-2
[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]
30129	01_HOOFD_370 (assessment of content completed)	1	node_01f37af7-0c2a-4fec-b283-e009e06bea9e	3121492	proc_-316325276	Mun-3
31264	01_HOOFD_370 (assessment of content completed)	550	node_01f37af7-0c2a-4fec-b283-e009e06bea9e	4582439	proc_-316325276	Mun-3
[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]
48325	01_HOOFD_370 (assessment of content completed)	1	node_0e968545-23b8-4920-84f6-5ea8e8d0601f	4258780	proc_-895511059	Mun-4
48598	01_HOOFD_370 (assessment of content completed)	1896	node_0e968545-23b8-4920-84f6-5ea8e8d0601f	5238596	proc_-895511059	Mun-4
49643	01_HOOFD_015 (phase application received)	10418	node_267826c5-8dc7-4b66-89bf-8ad9af3b1869	4473109	proc_-895511059	Mun-4
[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]
51923	01_HOOFD_370 (assessment of content completed)	17639	node_9ec1c634-454c-4f59-8e40-b2068cc743ab	3462724	proc_-1624093678	Mun-5
63879	01_HOOFD_-510_2 (enter senddate decision environmental permit)	32	node_f5712e4d-3316-4a90-a718-45e26f18cd45	4914544	proc_-1624093678	Mun-5
[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]	[. . .]

As shown from the table above we highlight with a light cyan color the same activity instances (e. g., *01_HOOFD_370 (assessment of content completed)* that are enacted along different process variants together with an info of their execution time and in which process instance they occur. Considering what we explain throughout the first case study we emphasis again the fact that in the same process different instances of the same activity might occur that belongs to different process instances.

Now, we address the adaptations made to the PW schema in order to fit the second scenario. We describe a list of these changes as follows:

- We alter dimension *Time* to aggregate by *hours* and not *minute* to avoid this table gets too big in size. This is because we have to populate this dimension with records for 4 years and not only 1 year as it was the case with the first case study. So, this means that 8760 records (365 days x 24 h) of time table will be generated for a year, whereas in total for 4 years will be: 35064 because 2012 is a leap year with 366 days. For example, for the date *2010-01-01* we have 24 tuples, from *FullDateTime*(2010-01-01 00:00:00.000), *Hour*(0),*FullDate*(2010-01-01) up to *FullDateTime*(2010-01-01 23:00:00.000), *Hour*(23), *FullDate*(2010-01-01). More details on the data generated for the *Date* and *Time* dimensions we give in the section:

- We have no information on which city this data came from, so *Geolocation* dimension cannot be populated with real data but we inserted some fictitious data in it. For instance, we know that this municipalities are based in Netherlands as a country, whereas for city we just write values such as *city 1* etc. and for the organization chart we can think that in each of the Municipality Offices there are some departments such as *Constructing Sector*, *Human Resources Department* and *Building Construction Bureau* (sub organization unit of *Constructing Sector*) where *Municipality Office* is in the top of these organization hierarchies.

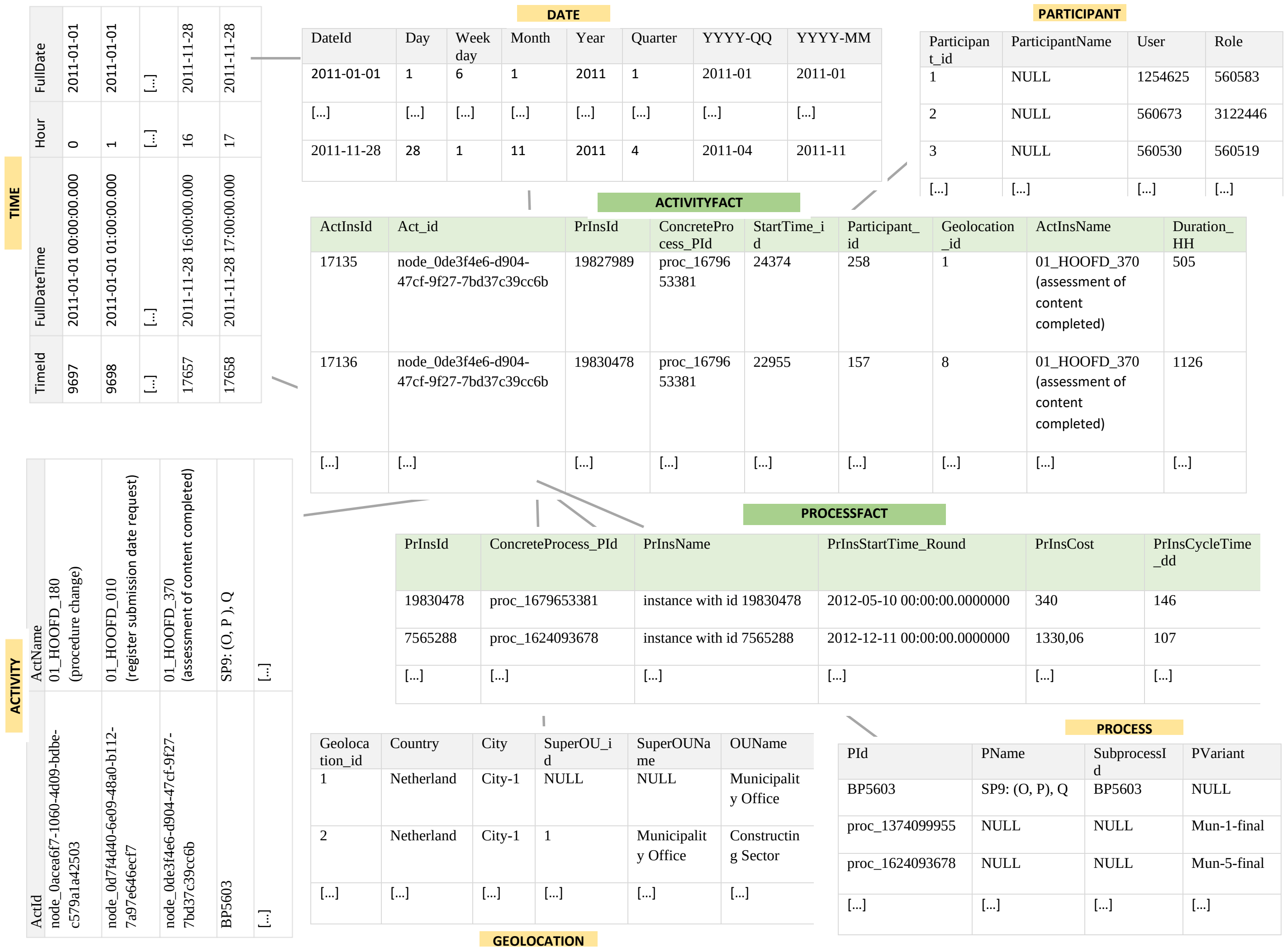
- *Participant* dimension is populated with data extracted from the logs which are anonymized a priori. For instance, instead of a participant name, a combination of numbers is used instead, e. g., *2670601*. Moreover, roles are not so clearly defined; we can assume that there are three type of resources:

  - *Monitoring Participant* which is responsible to monitor a typical work; most likely it has to have a functional expertise and not linked to a specific activity;
  - *Responsible Participant* is linked to the process instance in most of the cases; it can be some kind of a boss;
  - *Participant* which usually handles the work, e. g., an activity. But, still there is not a clear evidence that these roles are properly defined, because in many cases a monitoring participant is the same with a responsible participant and sometimes a participant is a monitoring participant. Therefore, we cannot argue if there is a coordinating mechanism between different organization units and participants involved (Peter and Liese, 2015)

- When we load the data from the data source we consider a different timeunit of *Duration* attribute as a measure to calculate *days* of the executed activities instead of *hours* or *minutes.* The reason behind it is that usually a building permit application requires at least 8 weeks to process the standard case and other 6 weeks if the permit application needs consultation from advisory board; so in total approximately 14 weeks to complete a specific permit application.

Accordingly, we give an excerpt of instances of the process warehouse for the building permit application as shown in Figure 8.17. In the middle of the figure there are two fact table instances *ActivityFact* and *ProcessFact* where instances about respective activity instances and process instances are given. And in the top, bottom and left part of the figure we position the relevant dimension table instances, such as: *Participant*, *Geolocation*,

Time, *Date*, *Process* and *Activity*.
We use as measures those already defined in Chapter 7 and we give relevant statistics in the next section.


TIME
DATE
PARTICIPANT
ACTIVITYFACT
PROCESSFACT
ACTIVITY
GEOLOCATION
PROCESS

TimeId	FullDateTime	Hour	FullDate
9697	2011-01-01 00:00:00.000	0	2011-01-01
9698	2011-01-01 01:00:00.000	1	2011-01-01
[...]	[...]	[...]	[...]
17657	2011-11-28 16:00:00.000	16	2011-11-28
17658	2011-11-28 17:00:00.000	17	2011-11-28

DateId	Day	Week day	Month	Year	Quarter	YYYY-QQ	YYYY-MM
2011-01-01	1	6	1	2011	1	2011-01	2011-01
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]
2011-11-28	28	1	11	2011	4	2011-04	2011-11

Participant_id	ParticipantName	User	Role
1	NULL	1254625	560583
2	NULL	560673	3122446
3	NULL	560530	560519
[...]	[...]	[...]	[...]

ActInsId	Act_id	PrInsId	ConcreteProcess_PId	StartTime_id	Participant_id	Geolocation_id	ActInsName	Duration_HH
17135	node_0de3f4e6-d904-47cf-9f27-7bd37c39cc6b	19827989	proc_1679653381	24374	258	1	01_HOOFD_370 (assessment of content completed)	505
17136	node_0de3f4e6-d904-47cf-9f27-7bd37c39cc6b	19830478	proc_1679653381	22955	157	8	01_HOOFD_370 (assessment of content completed)	1126
[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]	[...]

PrInsId	ConcreteProcess_PId	PrInsName	PrInsStartTime_Round	PrInsCost	PrInsCycleTime_dd
19830478	proc_1679653381	instance with id 19830478	2012-05-10 00:00:00.0000000	340	146
7565288	proc_1624093678	instance with id 7565288	2012-12-11 00:00:00.0000000	1330,06	107
[...]	[...]	[...]	[...]	[...]	[...]

ActId	ActName
node_0acea6f7-1060-4d09-bdbe-c579a1a42503	01_HOOFD_180 (procedure change)
node_0d7f4d40-6e09-48a0-b112-7a97e646ecf7	01_HOOFD_010 (register submission date request)
node_0de3f4e6-d904-47cf-9f27-7bd37c39cc6b	01_HOOFD_370 (assessment of content completed)
BP5603	SP9: (O, P), Q
[...]	[...]

Geolocation_id	Country	City	SuperOU_id	SuperOUName	OUName
1	Netherland	City-1	NULL	NULL	Municipality Office
2	Netherland	City-1	1	Municipality Office	Construction Sector
[...]	[...]	[...]	[...]	[...]	[...]

PId	PName	SubprocessId	PVariant
BP5603	SP9: (O, P), Q	BP5603	NULL
proc_1374099955	NULL	NULL	Mun-1-final
proc_1624093678	NULL	NULL	Mun-5-final
[...]	[...]	[...]	[...]

Figure 8.17: Instances of the process warehouse for the building permit application

8.2.3 Analysis results

By using our solution we can answer different type of queries that are interesting for business users. First, we address the achievement of core objectives we mention at the Introduction chapter and then verify that our solution can answer as well other queries provided by conventional BPM systems. We give a comprehensive analysis of our approach divided in the following two sections: *Process pattern analysis* and *Process analysis*. The former one aims to answer queries related to different patterns among different variants and the latter one aims to answer typical queries on a standalone process.

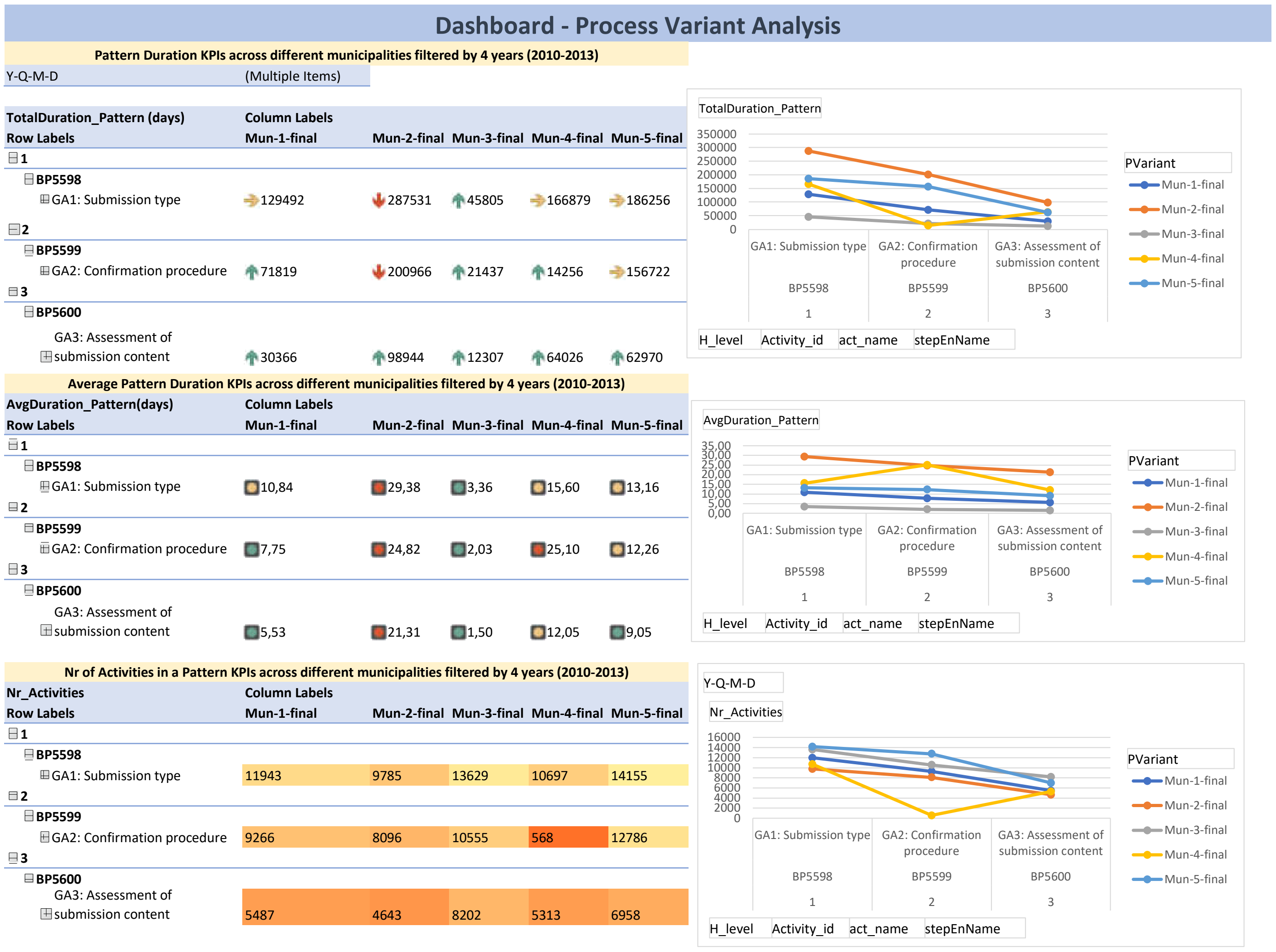
Process pattern analysis consists of :

- compute different aggregation measures to different process patterns
- give evidence that *roll-up* and *drill-down* OLAP operations can be performed on different process patterns by consolidating variants to an hierarchy
- offer the possibility to compare different patterns from different variants

Based on the three identified variation points among the five variants we can write some ad-hoc queries that might be of interest to know for assessing their respective performance and give suggestions for efficient improvements. We write some queries as follows:

- Find duration time of the permit applications processes based on the submitted type of request, which corresponds with *Pattern GA1*.
- Find average duration time of the permit applications processes based on the confirmed procedure, which corresponds with *Pattern GA2*.
- Find number of participants for each process pattern etc.

Initially we present different aggregation measures calculated across different variants as displayed in Figure 8.18 through an interactive dashboard in an excel spreadsheet.



Dashboard - Process Variant Analysis

Pattern Duration KPIs across different municipalities filtered by 4 years (2010-2013)

Y-Q-M-D (Multiple Items)

TotalDuration_Pattern (days)	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊞GA1: Submission type	➡129492	⬇287531	⬆45805	➡166879	➡186256
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	⬆71819	⬇200966	⬆21437	⬆14256	➡156722
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	⬆30366	⬆98944	⬆12307	⬆64026	⬆62970

Average Pattern Duration KPIs across different municipalities filtered by 4 years (2010-2013)

AvgDuration_Pattern(days)	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊞GA1: Submission type	🟧10,84	🟥29,38	🟩3,36	🟧15,60	🟧13,16
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	🟩7,75	🟥24,82	🟩2,03	🟥25,10	🟧12,26
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	🟩5,53	🟥21,31	🟩1,50	🟧12,05	🟩9,05

Nr of Activities in a Pattern KPIs across different municipalities filtered by 4 years (2010-2013)

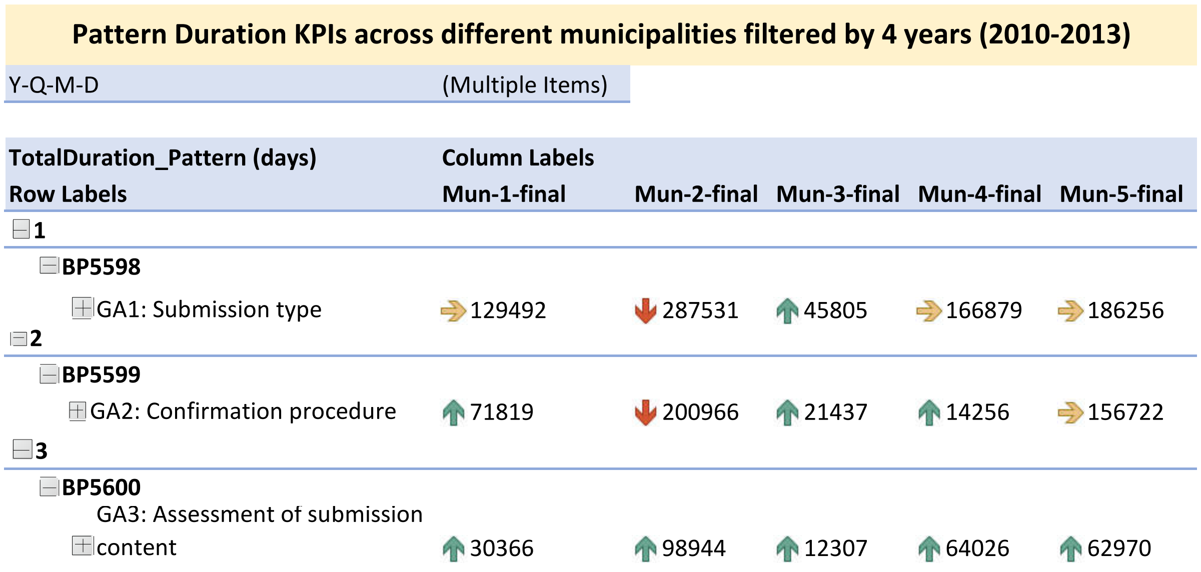
Nr_Activities	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊞GA1: Submission type	11943	9785	13629	10697	14155
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	9266	8096	10555	568	12786
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	5487	4643	8202	5313	6958

Figure 8.18: Dashboard of process variant analysis

As we already stated we use *PowerPivot* in Excel to perform powerful data analysis where our data is process-related. Based on different calculation columns and measures we created we build different Pivot tables to answer different queries.
In the dashboard are summarized different aggregated measures with corresponding lines charts. To explain more in detail the results we show them in separated figures. In Figure 8.19 we show results about duration(days) of the three process patterns across the five municipality variants for the period of 4 years from 2010 to 2013. User can filter these results to only a specific year/quarter/month or date.
To help users to quickly evaluate the current values a KPI is created where a target value is given and then a status threshold between a low and high threshold is specified. An alternative way to show visual measures of performance is trough a *conditional-if* rules. The rule in this case is: if value $<$ 33% i. e. 50000 days for a total of 4 years (1461 days) or 30 days in a year then *PatternDuration* metric is good, else if $>=$67% is bad and

in-between, i. e., >=36% and <67%. So, each value is associated with an arrow icon to indicate either positive performance (up green arrow), poor performance (down red arrow) and in-between (right orange arrow).

From the chart given below the table results we can conclude that Municipality-3 is performing better in each of the patterns so it requires less time to finish processing a permit application. Whereas Municipality-2 has the worst performance in all the first two patterns except for the third one which deals with handling the assessment of the submitted application. As, *GA1* pattern involves all activities from the submission request to the very end of the process we can consider it for a general performance for each of the variants. And *GA2* metrics emphasise the fact how each of the variants proceed to confirm next step activities in handling the application. So, from the values can be inferred that Mun-1, Mun-3 and Mun-5 are successfully handling this phase contrary to Mun-2 whereas Mun-4 is in-between.



Pattern Duration KPIs across different municipalities filtered by 4 years (2010-2013)

Y-Q-M-D (Multiple Items)

TotalDuration_Pattern (days)	Column Labels				
Row Labels	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**
⊟ **1**					
⊟ **BP5598**					
⊞ GA1: Submission type	➔ 129492	⬇ 287531	⬆ 45805	➔ 166879	➔ 186256
⊟ **2**					
⊟ **BP5599**					
⊞ GA2: Confirmation procedure	⬆ 71819	⬇ 200966	⬆ 21437	⬆ 14256	➔ 156722
⊟ **3**					
⊟ **BP5600**					
⊞ GA3: Assessment of submission content	⬆ 30366	⬆ 98944	⬆ 12307	⬆ 64026	⬆ 62970

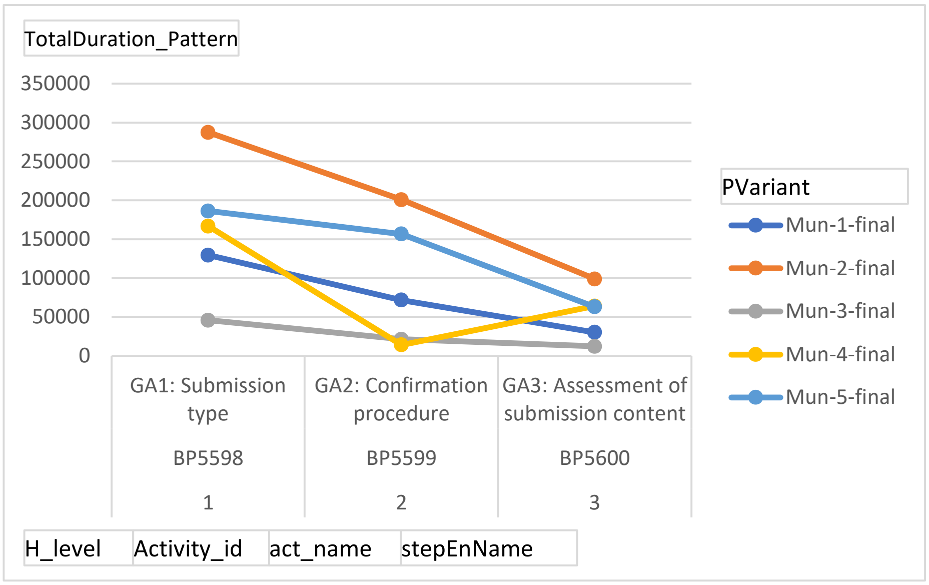
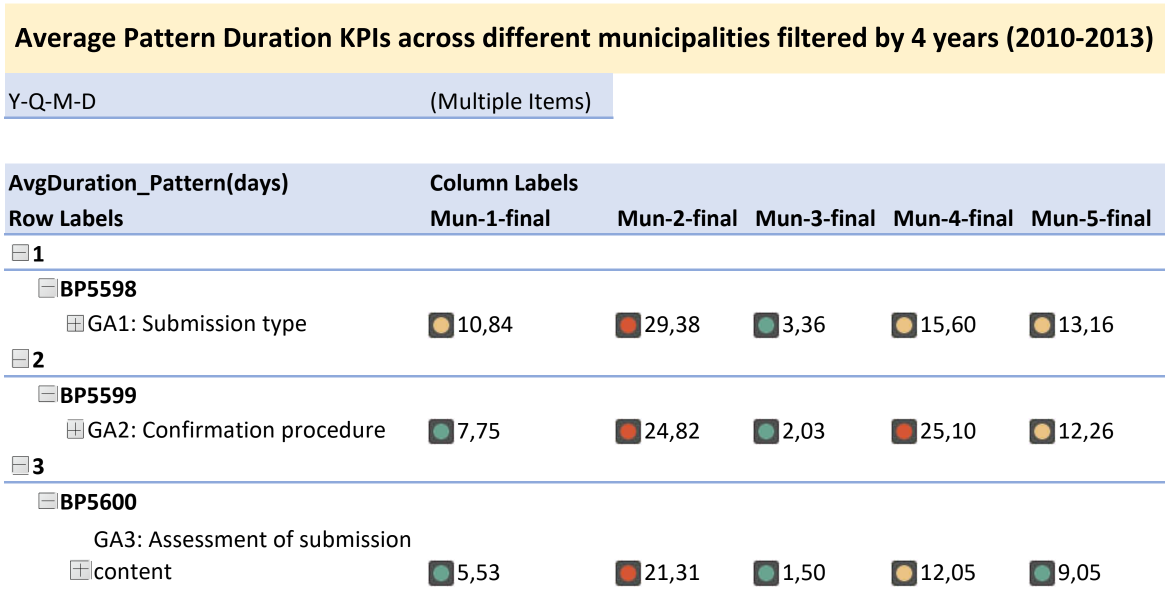


Figure 8.19: Pattern duration across five variants municipalities

Whereas in Figure 8.20 we show results about average duration of a pattern. Again, Mun-3 has the best performance followed up by Mun-1. Whereas Mun-5 and Mun-4 are somehow in-between whereas Mun-2 has again the worst performance in all the three patterns.



Average Pattern Duration KPIs across different municipalities filtered by 4 years (2010-2013)

Y-Q-M-D	(Multiple Items)

AvgDuration_Pattern(days)	Column Labels				
Row Labels	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**
⊟**1**					
⊟**BP5598**					
⊞GA1: Submission type	🟡10,84	🔴29,38	🟢3,36	🟡15,60	🟡13,16
⊟**2**					
⊟**BP5599**					
⊞GA2: Confirmation procedure	🟢7,75	🔴24,82	🟢2,03	🔴25,10	🟡12,26
⊟**3**					
⊟**BP5600**					
⊞GA3: Assessment of submission content	🟢5,53	🔴21,31	🟢1,50	🟡12,05	🟢9,05

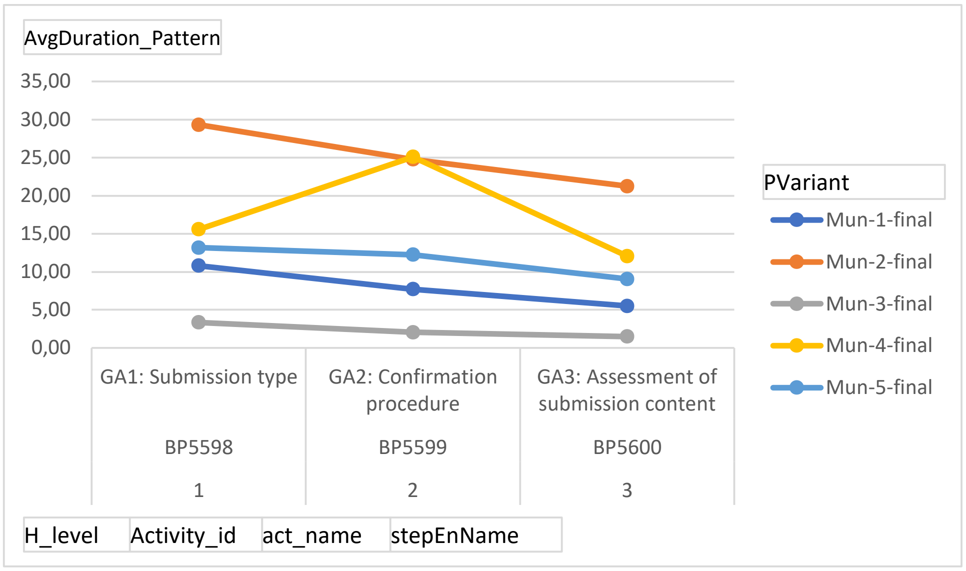
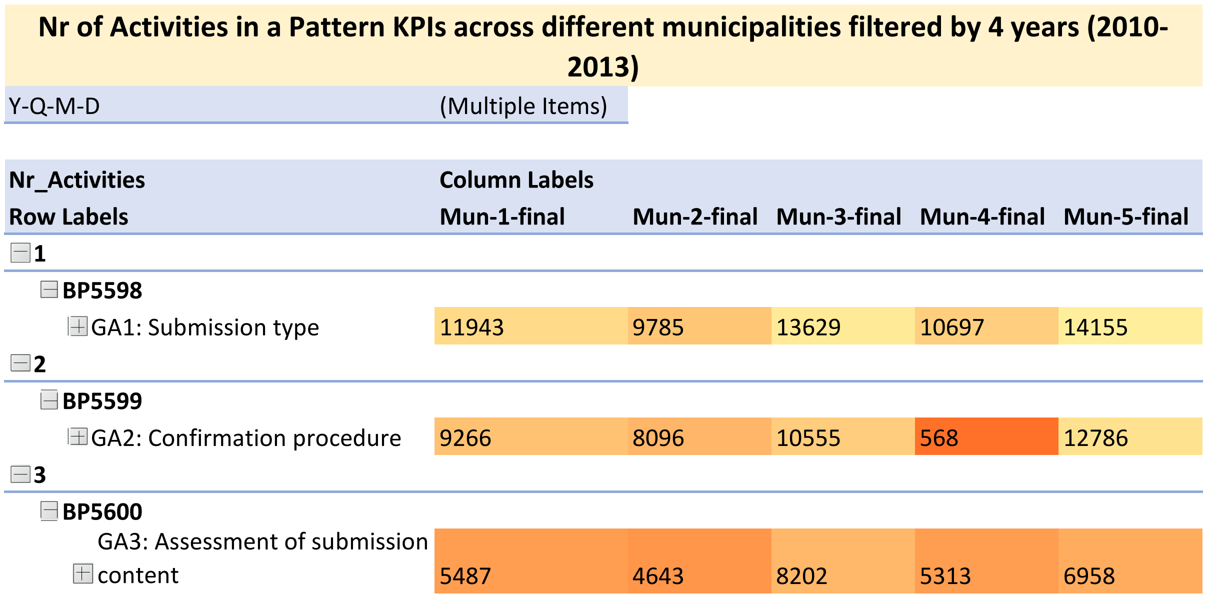


Figure 8.20: Average pattern duration across five variants municipalities

And finally, in Figure 8.21 we show results about the number of activity instances occurred in each process variant in four years. As we can see from the chart Mun-5 has the greatest number (highlighted with a yellow color) of activity instances (considering *GA1 pattern*) whereas Mun-4 has the smallest number (highlighted with an orange color) followed up by Mun-2. Whereas, Mun-3 and Mun-1 has a moderated number of activity instances. If we look at the different patterns, specifically to the
emphGA3 pattern we can see that in Mun-3 the number of activities is greater than all

other four variants. Therefore, based on to the two previous results we can presume that Mun-3 is performing better because when it comes to the assessment of the permit application i. e., *GA3 pattern* they maximise the number of activity instances in order to successfully finish it.



Nr of Activities in a Pattern KPIs across different municipalities filtered by 4 years (2010-2013)

Y-Q-M-D (Multiple Items)

Nr_Activities	Column Labels				
Row Labels	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**
⊟ **1**					
⊟ **BP5598**					
⊞ GA1: Submission type	11943	9785	13629	10697	14155
⊟ **2**					
⊟ **BP5599**					
⊞ GA2: Confirmation procedure	9266	8096	10555	568	12786
⊟ **3**					
⊟ **BP5600**					
⊞ GA3: Assessment of submission content	5487	4643	8202	5313	6958

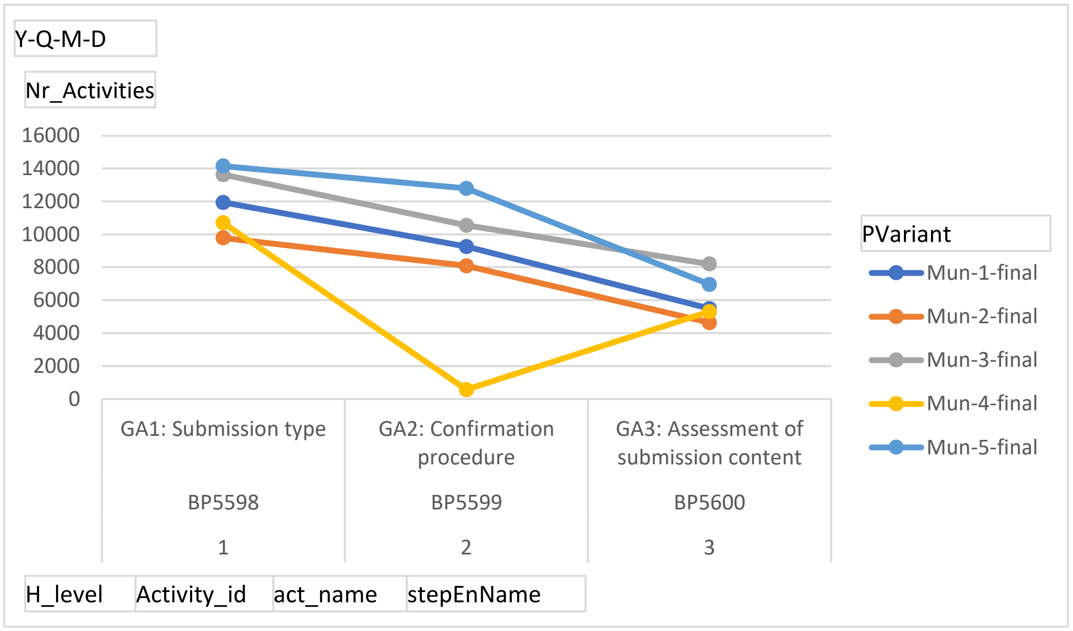
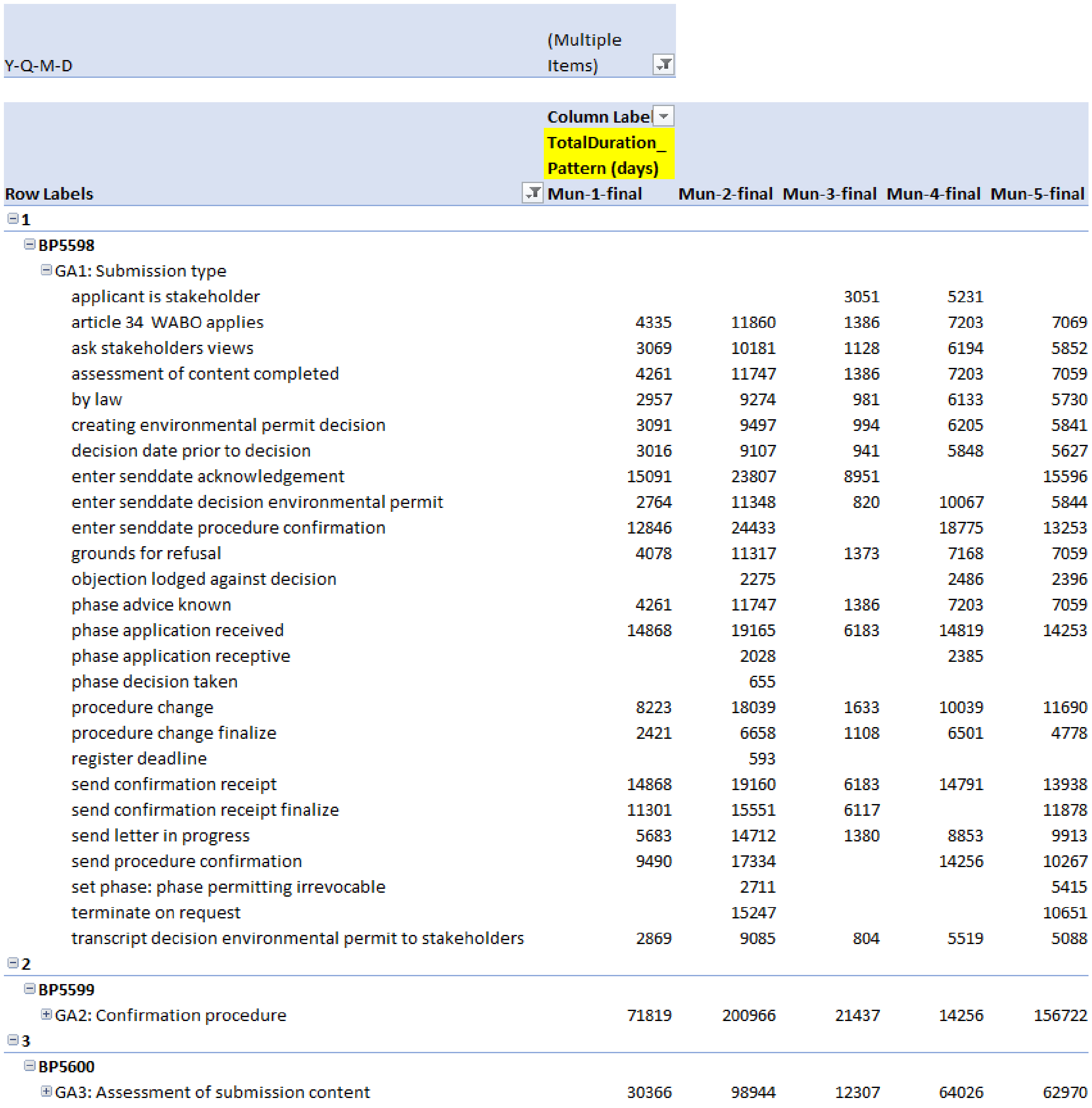


Figure 8.21: Pattern duration across five variants municipalities

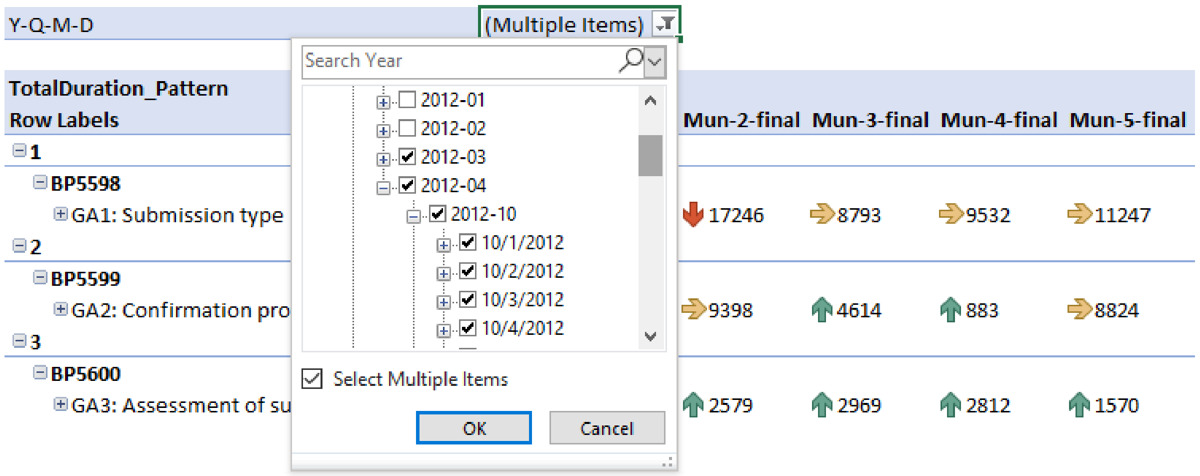
We give a demonstration as well on how *roll-up* and *drill-down* OLAP operations can be performed as displayed in the Figure 8.22



Y-Q-M-D	(Multiple Items)				
	Column Labels				
	TotalDuration_Pattern (days)				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊟GA1: Submission type					
applicant is stakeholder			3051	5231	
article 34 WABO applies	4335	11860	1386	7203	7069
ask stakeholders views	3069	10181	1128	6194	5852
assessment of content completed	4261	11747	1386	7203	7059
by law	2957	9274	981	6133	5730
creating environmental permit decision	3091	9497	994	6205	5841
decision date prior to decision	3016	9107	941	5848	5627
enter senddate acknowledgement	15091	23807	8951		15596
enter senddate decision environmental permit	2764	11348	820	10067	5844
enter senddate procedure confirmation	12846	24433		18775	13253
grounds for refusal	4078	11317	1373	7168	7059
objection lodged against decision		2275		2486	2396
phase advice known	4261	11747	1386	7203	7059
phase application received	14868	19165	6183	14819	14253
phase application receptive		2028		2385	
phase decision taken		655			
procedure change	8223	18039	1633	10039	11690
procedure change finalize	2421	6658	1108	6501	4778
register deadline		593			
send confirmation receipt	14868	19160	6183	14791	13938
send confirmation receipt finalize	11301	15551	6117		11878
send letter in progress	5683	14712	1380	8853	9913
send procedure confirmation	9490	17334		14256	10267
set phase: phase permitting irrevocable		2711			5415
terminate on request		15247			10651
transcript decision environmental permit to stakeholders	2869	9085	804	5519	5088
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	71819	200966	21437	14256	156722
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	30366	98944	12307	64026	62970

Figure 8.22: *Roll-up* and *Drill-down* OLAP operations

As shown in the above figure there are three levels of hierarchy that corresponds to the three GAs identified and if we want to explore activities that are consolidated to each of them we can click + button in each of the row labels in the excel sheet. We show the expanded activities of *GA1: Submission type* where total duration of each of these activities during 4 years (from 2010-2013) are measured. We can filter these values for a specific year by selecting the filter option on the cell *Multiple items* that corresponds to the hierarchy attribute *Y-Q-M-D*. Moreover, as date values are consolidated to this hierarchy the user can filter values to different levels, from year to quarter, or quarter to month, and month to a specific date as shown in Figure 8.23 In the above table there are some *blank* cells that corresponds to a specific activity to a specific variant municipality. This means that this activity is not part of that variant but it belongs to the variant where a cell value is given.



Y-Q-M-D (Multiple Items)

TotalDuration_Pattern Row Labels	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1				
⊟BP5598				
⊞GA1: Submission type	↓17246	→8793	→9532	→11247
⊟2				
⊟BP5599				
⊞GA2: Confirmation pro	→9398	↑4614	↑883	→8824
⊟3				
⊟BP5600				
⊞GA3: Assessment of su	↑2579	↑2969	↑2812	↑1570

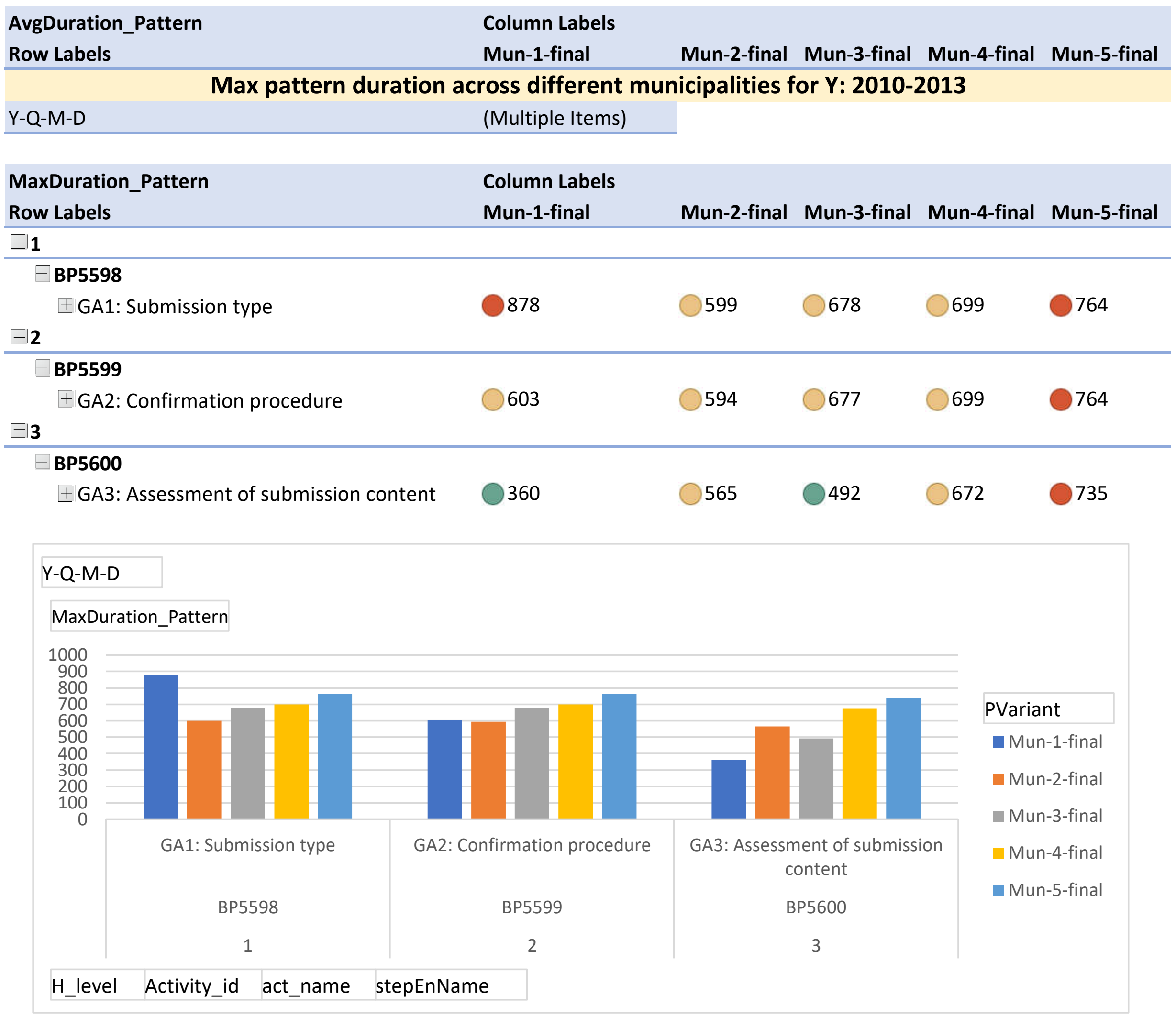
Search Year
- ☐ 2012-01
- ☐ 2012-02
- ☑ 2012-03
- ☑ 2012-04
  - ☑ 2012-10
    - ☑ 10/1/2012
    - ☑ 10/2/2012
    - ☑ 10/3/2012
    - ☑ 10/4/2012

☑ Select Multiple Items

Figure 8.23: Filter patterns based on different *date* hierarchy values

In the same pivot table we added other aggregated measures such as *MaxPatternDuration* and *Nr_Participants* in a pattern but due to page margins we display them in separate figures as follows.

In Figure 8.24 we give results about max duration of different patterns across different variants. From the above figure we can say Mun-5 reports the greatest values for max duration, so, they received the upper limit on the amount of time needed to execute a permit application. This means that this maximum was due to the amount of time of activities involved along the pattern. We can say that there is a slight difference in values or even equal values such as Mun-3, Mun-4, Mun-5 between GA1 and GA2 which means that the maximum time was caused by activities that handle the "confirmation procedure" (as GA2 pattern activities are enacted after GA1) and the sequential ones in the control flow up to the third pattern GA3. As illustrated in the associated chart Mun-1 and Mun-3 report the smallest values of this metric in the GA3 pattern on how to process the assessment of the request.



AvgDuration_Pattern	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final

Max pattern duration across different municipalities for Y: 2010-2013

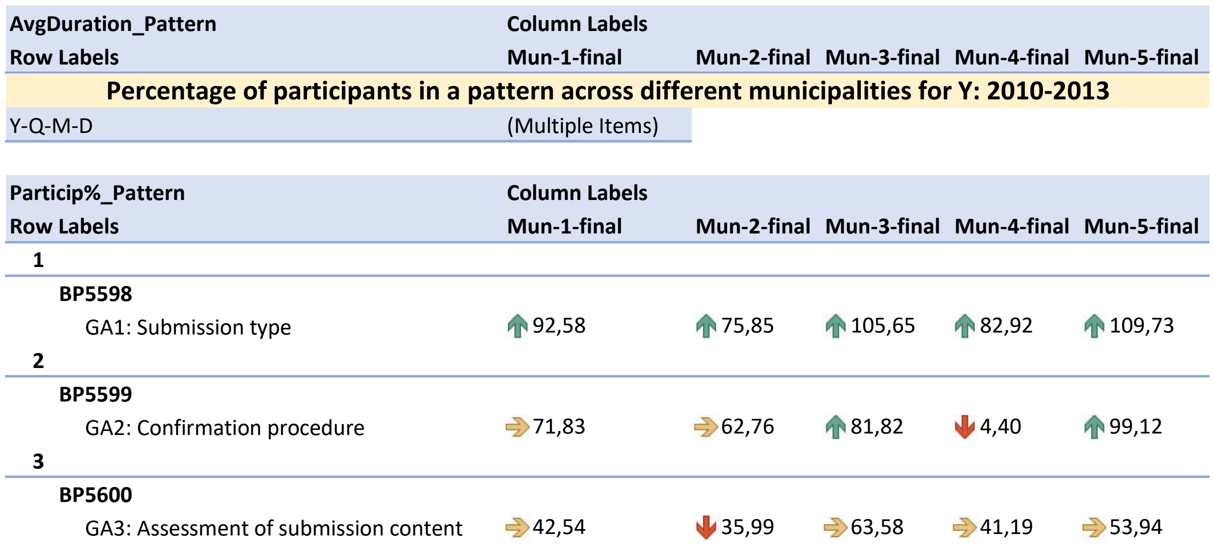
Y-Q-M-D	(Multiple Items)

MaxDuration_Pattern	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊞GA1: Submission type	🔴878	🟠599	🟠678	🟠699	🔴764
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	🟠603	🟠594	🟠677	🟠699	🔴764
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	🟢360	🟠565	🟢492	🟠672	🔴735

Figure 8.24: Max pattern duration across five variants municipalities

Another important statistics related to participants/agents that perform activities about variants municipalities performance is given in Figure 8.25. In this figure, we show values that report the percentage of participants for each pattern and across different variants.

If we consider GA1 as the highest level in the hierarchy we see that Mun-3 and Mun-5 have more than 100% of the participants executing the activities(utilization of people is quite effective); followed by Mun-1, Mun-4 and the last one Mun-2. Whereas for GA2 and GA3 pattern Mun-2 and Mun-4 report the smallest values in participants percentage which can be consider a good argument to point out the bad performance from the previous metrics as well.



AvgDuration_Pattern	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final

Percentage of participants in a pattern across different municipalities for Y: 2010-2013

Y-Q-M-D	(Multiple Items)

Particip%_Pattern	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
1					
BP5598					
GA1: Submission type	↑ 92,58	↑ 75,85	↑ 105,65	↑ 82,92	↑ 109,73
2					
BP5599					
GA2: Confirmation procedure	→ 71,83	→ 62,76	↑ 81,82	↓ 4,40	↑ 99,12
3					
BP5600					
GA3: Assessment of submission content	→ 42,54	↓ 35,99	→ 63,58	→ 41,19	→ 53,94

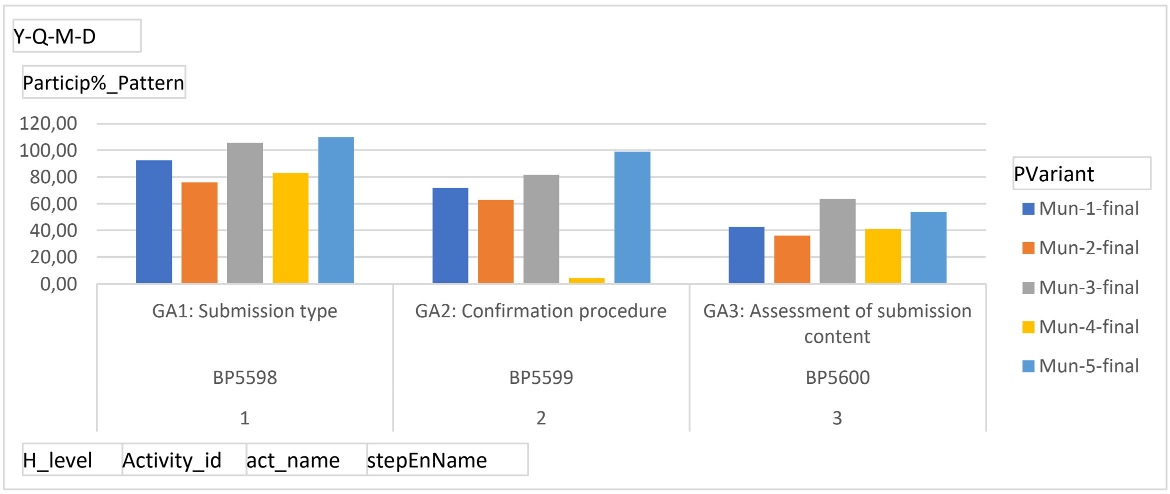


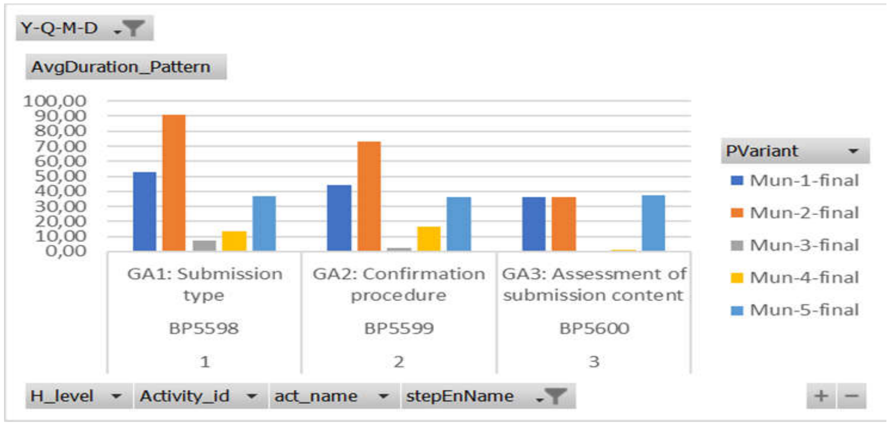
Figure 8.25: Percentage of participants in a pattern across five variants municipalities

It is important for this report as well that there exists 76 total participants among the five variants. But, as recorded from the event logs role assignments between those participants differs and are reported in total 376 relationships between responsible participants and monitoring participants.
Moreover, we show average duration pattern results in Figure 8.26 across multiple variants for two specific years too, e. g., the first one 2010 and the last one 2013.

Average Pattern Duration KPIs across different municipalities for Y: 2010

Y-Q-M-D	2010

AvgDuration_Pattern	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊞GA1: Submission type	🟠52,58	🔴90,68	🟢7,07	🟢13,36	🟠36,84
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	🟠44,45	🔴72,87	🟢2,24	🟢16,44	🟠36,40
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	🟠36,44	🟠36,32	🟢0,23	🟢1,29	🟠37,50



Average Pattern Duration KPIs across different municipalities for Y: 2013

Y-Q-M-D	2013

AvgDuration_Pattern	Column Labels				
Row Labels	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
⊟1					
⊟BP5598					
⊞GA1: Submission type	🟠1,01	🔴1,82	🟢0,60	🟠0,66	🟢0,62
⊟2					
⊟BP5599					
⊞GA2: Confirmation procedure	🟢0,30	🔴1,25	🟢0,23	🟢0,16	🟢0,19
⊟3					
⊟BP5600					
⊞GA3: Assessment of submission content	🟢0,08	🔴1,30	🟢0,09	🟢0,49	🟢0,10

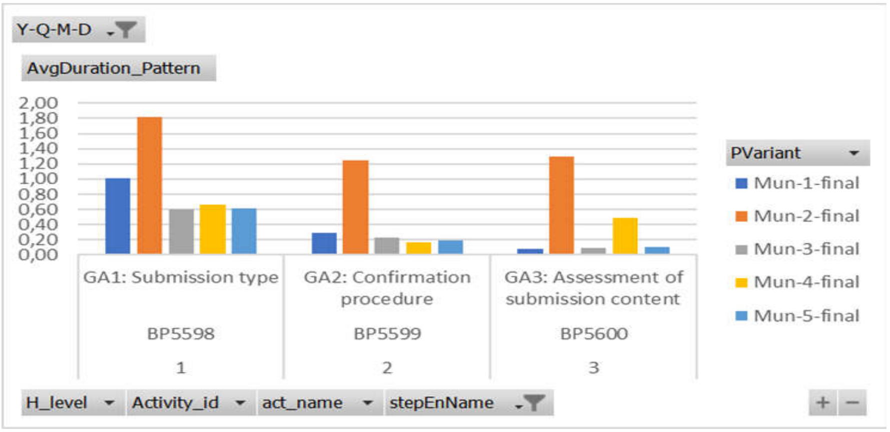


Figure 8.26: Average duration pattern across five variant for two specific years

As seen from the charts, in 2010, Mun-3 has the best performance, followed by Mun-4, whereas Mun-1 and Mun-5 have moderated values instead Mun-2 has the worst performance. Concerning results for 2013, Mun-5 instead has the smallest average duration for all three patterns, followed by Mun-3 and then Mun-1 and Mun-4 have both a slightly better performance especially in the two last patterns, i. e., *GA2* and *GA3*. This means that they have to change only the first part of the permit application which is about on how to handle the submission request.
Also, we should emphasis the fact that during 2013 there were two participants (participant: *185* and *73*) who moved to work on Mun-5 from Mun-2 and Mun-4 respectively. It seems the reason behind it was the relocation of these two offices. It is significant to acknowledge that municipalities during the last year have improved their process performance.

Whereas the rest of statistics concerning all variants but for other years we show them in the Appendix section Chapter B.
Based on the aforementioned metrics about different patterns across different variants we can derive other metrics as in the following Figure 8.27: As depicted in figure below we derive other patterns, such as *Pattern x = a-b* which consists of all specialized activities of *GA1* up to the point of *GA2*, and *Pattern y =b-c* where *a=all activities starting from GA1 variation point up to the end of the process*, whereas *b=all activities starting from GA2 variation point up to the end of the process* and *c=all activities starting from GA3 variation point up to the end of the process.* So, we can argue how these variants deal with submission type process context where activities *A, B, C, D, E* and procedure confirmation context where activities *F, G and sequential ones up to the GA3 point* are defined and executed in each of the variants.

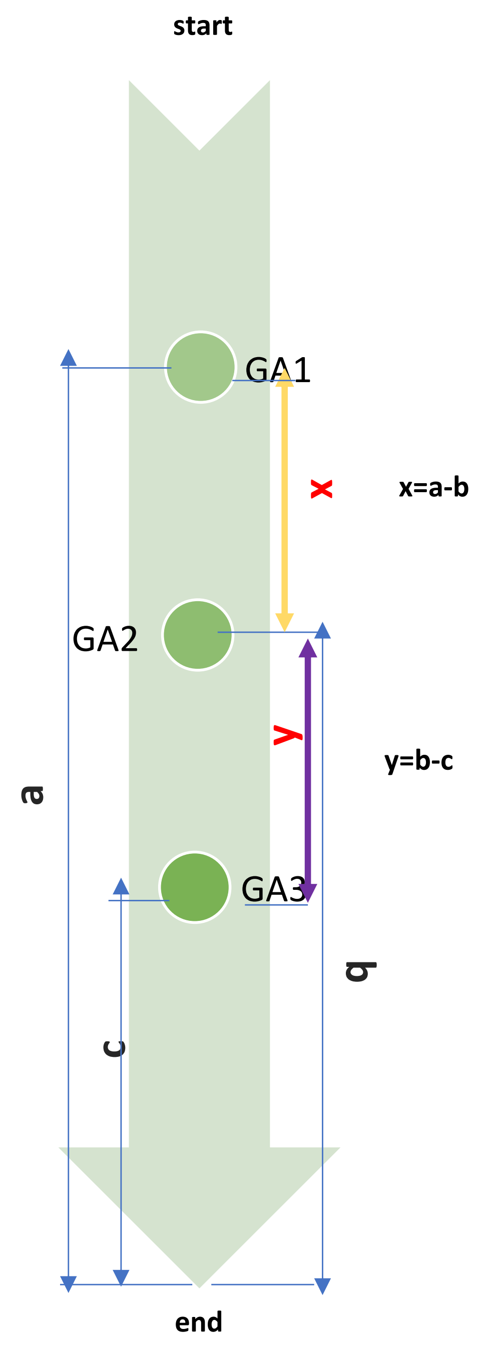
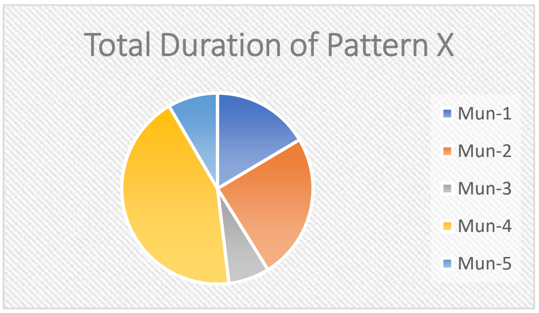


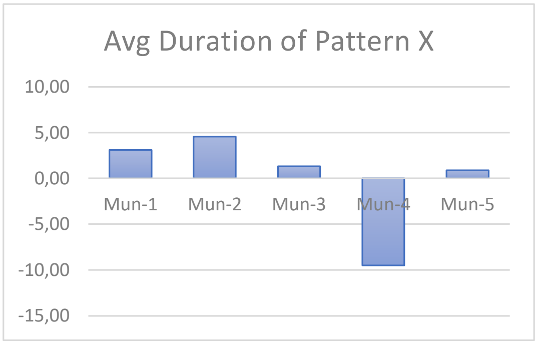
Figure 8.27: Some derived patterns

According to the formulas we defined we can derive relevant information based on two different patterns, X and Y. We calculate two different measures: total and average duration of Pattern X as displayed in Figure 8.28



Total Duration of specialized activities of GA1 up to GA2 for a period of 4 years: 2010-2013

Mun-1	Mun-2	Mun-3	Mun-4	Mun-5
57673	86565	24368	152623	29534



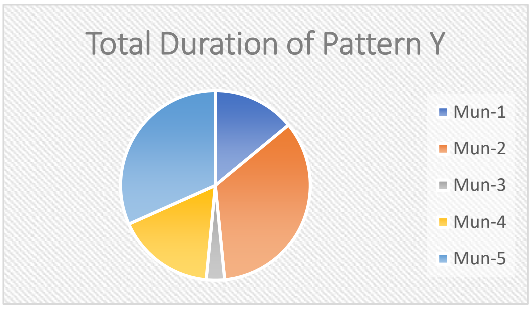
Average Duration of specialized activities of GA1 up to GA2 for a period of 4 years: 2010-2013

Mun-1	Mun-2	Mun-3	Mun-4	Mun-5
3,09	4,56	1,33	-9,50	0,90

Figure 8.28: Total and average duration of *submission type* activities as specialized activities of *GA1* up to *GA2*

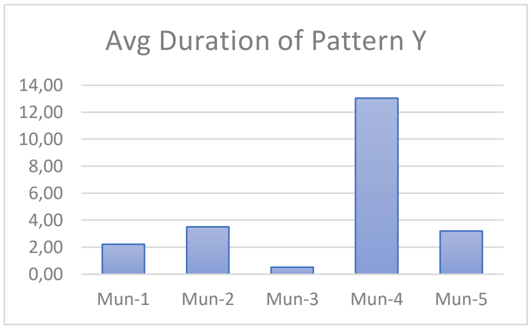
From the results visualized in the above charts we can report the lowest absolute value for *pattern x* for Mun-3 thus achieving the best performance, followed by Mun-5. Mun-1 and Mun-2 are not as good as Mun-3 but not as bad as Mun-4 which reports the greatest total duration value and the absolute average duration value among the other 4 variants.

Whereas, in Figure 8.29 we show results of total (in days) and average duration of *Pattern Y*.



Total Duration of specialized activities of GA2 up to GA3 for a period of 4 years: 2010-2013

Mun-1	Mun-2	Mun-3	Mun-4	Mun-5
41453	101975	9130	-49770	93752



Average Duration of specialized activities of GA1 up to GA2 for a period of 4 years: 2010-2013

Mun-1	Mun-2	Mun-3	Mun-4	Mun-5
2,22	3,51	0,53	13,05	3,21

Figure 8.29: Total and average duration of *confirmation procedure* activities as specialized activities of *GA2* up to *GA3*

From the results visualized in the above charts we can report the lowest absolute value for *pattern y* for Mun-3 thus achieving the best performance. But, if we consider the negative value (but we take in consideration the absolute value) of Mun-4 we can reason that the negative value behind it stems as the difference between *GA2* and *GA3* values ($GA3>GA2$). Therefore, only very few activities are executed as specialized activities of *GA2* and then the process ends. So, activities were heavily distributed mostly among the first and the last pattern. Whereas, Mun-1 and Mun-5 report in-between values and Mun-2 report the worst performance in activities lasting for too long.

Using the statistics we explained in this section business users already gained deeper and more fined grained insights on different patterns across different variants. Having this information available for handling building permit application in different process contexts is immensely valuable so the demand for process improvement is ever more feasible. To summarize we conclude the following effective improvements based on the previous described statistics analysis:

- **Mun-1** has not so good performance on total duration (in days) and average duration for activities comprised in the first pattern: *pattern x*. Whereas, reports a better performance on these metrics for the second pattern: *pattern y*. And for the last one, pattern *GA3* is slightly better. An effective improvement may be suggested to the first and second pattern on trying to change their process behaviour compared to Mun-3.
- **Mun-2** has the worst performance on total duration (in days) and average duration

among the three process patterns. Therefore a lot of improvements should be foreseen for this variant in all the three patterns, especially in the third pattern on handling the assessment of a permit application where the relevant activities are lasting for too long and thus with a greater average time's value.

- **Mun-3** has the best performance on total duration (in days) among the three process patterns whereas only good performance on average duration for *pattern x*. Therefore, the process context of *submission type* requests can be improved by following the example of Mun-5 in this regard. As well, how to better utilize people in performing their work can be further improved especially people handling activities of *pattern y*. Therefore, it can slightly improve on increasing the participant percentage for handling *confirmation procedure* process context compare to Mun-5.

- **Mun-4** has the greatest value on total duration (in days) of *pattern x*, so they are having problem in dealing with citizens submission requests. Whereas, *pattern y* they report a negative value because activities involved here are quite a few compared too many activities in the next pattern *GA3*. A reason behind is that after executing activities part of *pattern y* the process flow is cancelled or ended. Thus, some improvements might be consider definitely in *pattern y* on how to properly handle the procedure confirmation flow.

- **Mun-5** has a good performance on all the three patterns during the four years period. Some effective improvements might be consider in the third pattern *GA3*.

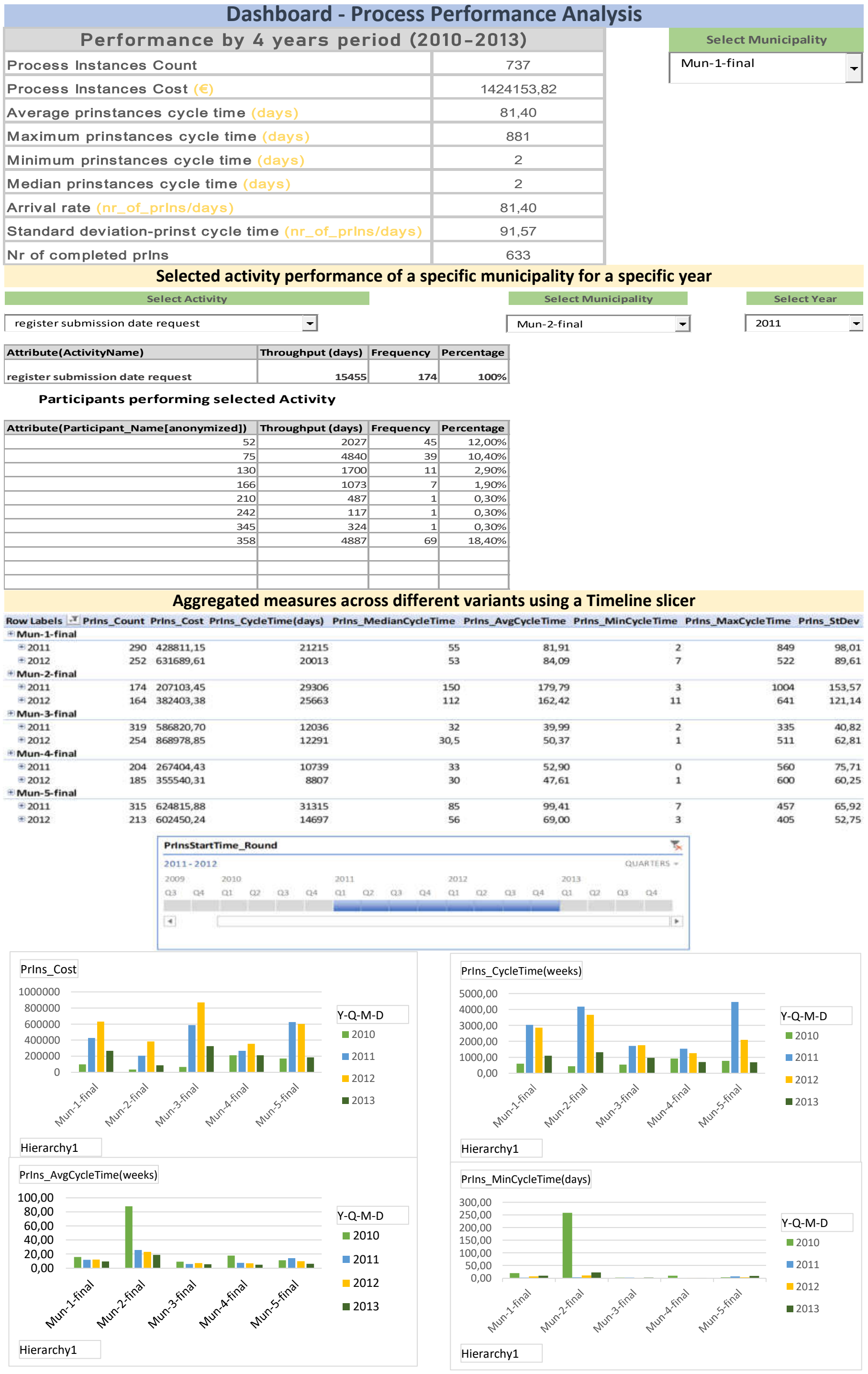
In the next section we demonstrate statistics on a standalone process.

Process analysis consists of :

- compute different aggregation measures to a particular process or activity
- compare between activity/process throughput/cycle time, frequency and cost across variants

Initially we give a demonstration of different aggregation measures calculated for a specific process as displayed in Figure 8.30 through a dashboard in an excel spreadsheet.
In the top part of this interactive dashboard we show table results on process performance by a particular municipality for a four year period (2010-2014).
In the middle we report results for a particular activity in a specific process performed in a specific year.
And last in the bottom one we show some of the aggregated measures such as *activity/process cycle time, process cost*[3] *, arrival rate etc.* for different municipalities filtered by different years using a time-line slicer where the user can select a datetime value based on the attribute *PrInsStartTime* rounded value. These datetime values can be selected and filtered either by quarter, years, month or specific days.

[3]Process Cost seems to refer to permit fee citizens have to pay instead of the real internal cost



Dashboard - Process Performance Analysis

Performance by 4 years period (2010-2013)	
Process Instances Count	737
Process Instances Cost (€)	1424153,82
Average prinstances cycle time (days)	81,40
Maximum prinstances cycle time (days)	881
Minimum prinstances cycle time (days)	2
Median prinstances cycle time (days)	2
Arrival rate (nr_of_prIns/days)	81,40
Standard deviation-prinst cycle time (nr_of_prIns/days)	91,57
Nr of completed prIns	633

Select Municipality: Mun-1-final

Selected activity performance of a specific municipality for a specific year

Select Activity: register submission date request
Select Municipality: Mun-2-final
Select Year: 2011

Attribute(ActivityName)	Throughput (days)	Frequency	Percentage
register submission date request	15455	174	100%

Participants performing selected Activity

Attribute(Participant_Name[anonymized])	Throughput (days)	Frequency	Percentage
52	2027	45	12,00%
75	4840	39	10,40%
130	1700	11	2,90%
166	1073	7	1,90%
210	487	1	0,30%
242	117	1	0,30%
345	324	1	0,30%
358	4887	69	18,40%

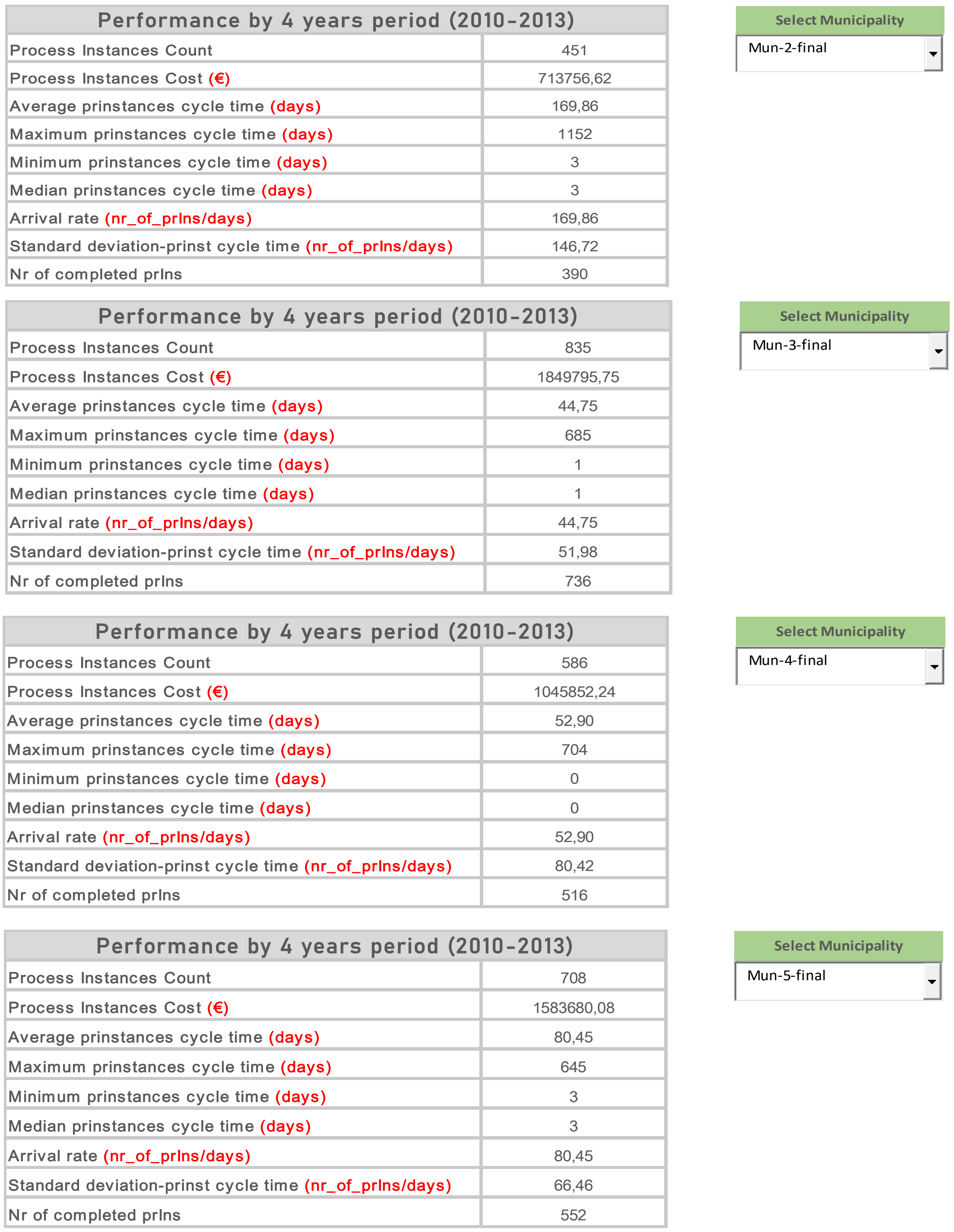
Aggregated measures across different variants using a Timeline slicer

Row Labels	PrIns_Count	PrIns_Cost	PrIns_CycleTime(days)	PrIns_MedianCycleTime	PrIns_AvgCycleTime	PrIns_MinCycleTime	PrIns_MaxCycleTime	PrIns_StDev
Mun-1-final								
2011	290	428811,15	21215	55	81,91	2	849	98,01
2012	252	631689,61	20013	53	84,09	7	522	89,61
Mun-2-final								
2011	174	207103,45	29306	150	179,79	3	1004	153,57
2012	164	382403,38	25663	112	162,42	11	641	121,14
Mun-3-final								
2011	319	586820,70	12036	32	39,99	2	335	40,82
2012	254	868978,85	12291	30,5	50,37	1	511	62,81
Mun-4-final								
2011	204	267404,43	10739	33	52,90	0	560	75,71
2012	185	355540,31	8807	30	47,61	1	600	60,25
Mun-5-final								
2011	315	624815,88	31315	85	99,41	7	457	65,92
2012	213	602450,24	14697	56	69,00	3	405	52,75

Figure 8.30: Dashboard of process performance analysis

We give all of the above results in separated figures as follows:
In Figure 8.31 we show other four remaining municipalities performance for the four year period. As reported the highest value of process instances cost belongs to Mun-3, followed

by Mun-5 and Mun-1 and Mun-4. Whereas, Mun-2 reports the lowest value on the cost of the process instances which can be a strong argument to reason about the negative overall process performance of this variant.
If we consider measure *nr of completed process* Mun-3 again reports the highest value on this numbers, followed by Mun-1, Mun-5 and Mun-4. Again, Mun-2 has the lowest value on this measure as well. And finally, one important measure that is of very much important when analysing process performance is *average process instance cycle time or process instance duration*. Based on these values against different municipalities we can conclude that Mun-3 has the lowest average duration of process instances, followed by Mun-4. Mun-5 and Mun-1 report in-between values whereas Mun-2 report the highest average duration translating into a bad performance of this process.



Performance by 4 years period (2010-2013)	
Process Instances Count	451
Process Instances Cost (€)	713756,62
Average prinstances cycle time (days)	169,86
Maximum prinstances cycle time (days)	1152
Minimum prinstances cycle time (days)	3
Median prinstances cycle time (days)	3
Arrival rate (nr_of_prIns/days)	169,86
Standard deviation-prinst cycle time (nr_of_prIns/days)	146,72
Nr of completed prIns	390

Select Municipality

Mun-2-final

Performance by 4 years period (2010-2013)	
Process Instances Count	835
Process Instances Cost (€)	1849795,75
Average prinstances cycle time (days)	44,75
Maximum prinstances cycle time (days)	685
Minimum prinstances cycle time (days)	1
Median prinstances cycle time (days)	1
Arrival rate (nr_of_prIns/days)	44,75
Standard deviation-prinst cycle time (nr_of_prIns/days)	51,98
Nr of completed prIns	736

Select Municipality

Mun-3-final

Performance by 4 years period (2010-2013)	
Process Instances Count	586
Process Instances Cost (€)	1045852,24
Average prinstances cycle time (days)	52,90
Maximum prinstances cycle time (days)	704
Minimum prinstances cycle time (days)	0
Median prinstances cycle time (days)	0
Arrival rate (nr_of_prIns/days)	52,90
Standard deviation-prinst cycle time (nr_of_prIns/days)	80,42
Nr of completed prIns	516

Select Municipality

Mun-4-final

Performance by 4 years period (2010-2013)	
Process Instances Count	708
Process Instances Cost (€)	1583680,08
Average prinstances cycle time (days)	80,45
Maximum prinstances cycle time (days)	645
Minimum prinstances cycle time (days)	3
Median prinstances cycle time (days)	3
Arrival rate (nr_of_prIns/days)	80,45
Standard deviation-prinst cycle time (nr_of_prIns/days)	66,46
Nr of completed prIns	552

Select Municipality

Mun-5-final

Figure 8.31: Aggregated measures for specific municipalities for four years period

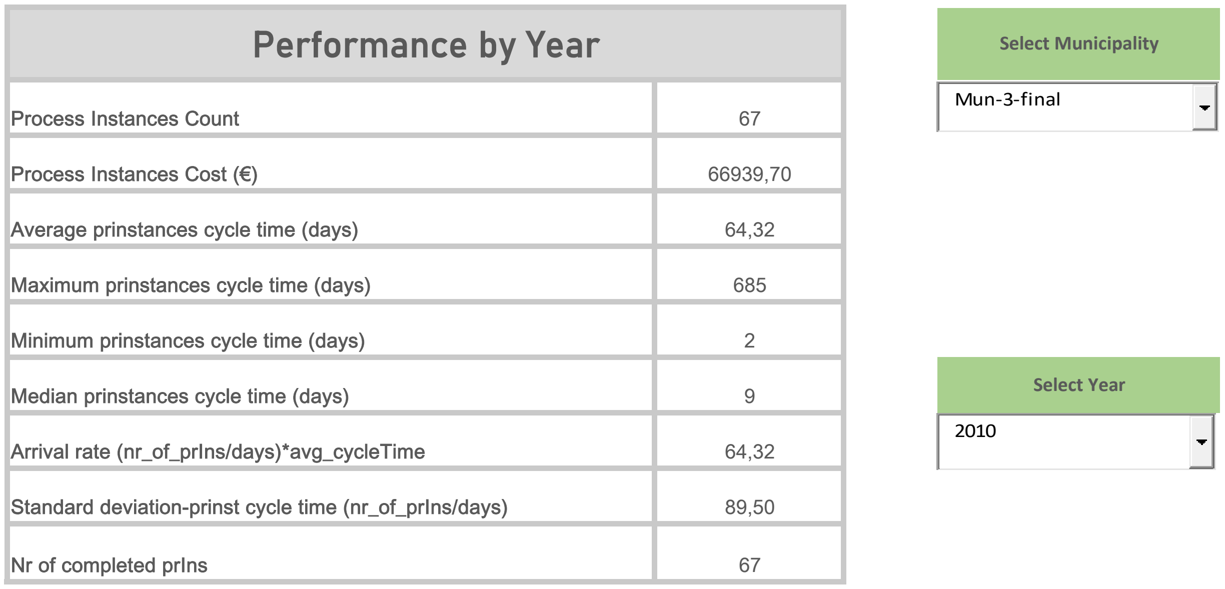
In Figure 8.31 we show process performance based on different aggregated measures such as:

- $PrIns\_AvgTime(days)$, average cycle time (duration) of process instances occurred in a process for four years period

- *PrIns_MinTime(days)*, minimum cycle time (duration) of process instances occurred in a process for four years period
- *PrIns_MaxTime(days)*, maximum cycle time (duration) of process instances occurred in a process for four years period
- *PrIns_MedianTime(days)*, median cycle time (duration) of process instances occurred in a process for four years period
- *PrIns_StDev*, the standard deviation of process instances occurred in a process for four years period
- *PrIns_ArrivalRate*, arrival rate of of process instances occurred in a process for four years period
- *PrIns_Cost*, cost of permit fee citizens have to pay
- *PrIns_completed_Nr*, number of completed process instances for four years period

All these reported measures are automatically refreshed based on the municipality the user has selected in the listbox in the top.

We give a different way of reporting these results by specific year where user can select which year he/she wants to show information as displayed in Figure 8.32



Performance by Year	
Process Instances Count	67
Process Instances Cost (€)	66939,70
Average prinstances cycle time (days)	64,32
Maximum prinstances cycle time (days)	685
Minimum prinstances cycle time (days)	2
Median prinstances cycle time (days)	9
Arrival rate (nr_of_prIns/days)*avg_cycleTime	64,32
Standard deviation-prinst cycle time (nr_of_prIns/days)	89,50
Nr of completed prIns	67

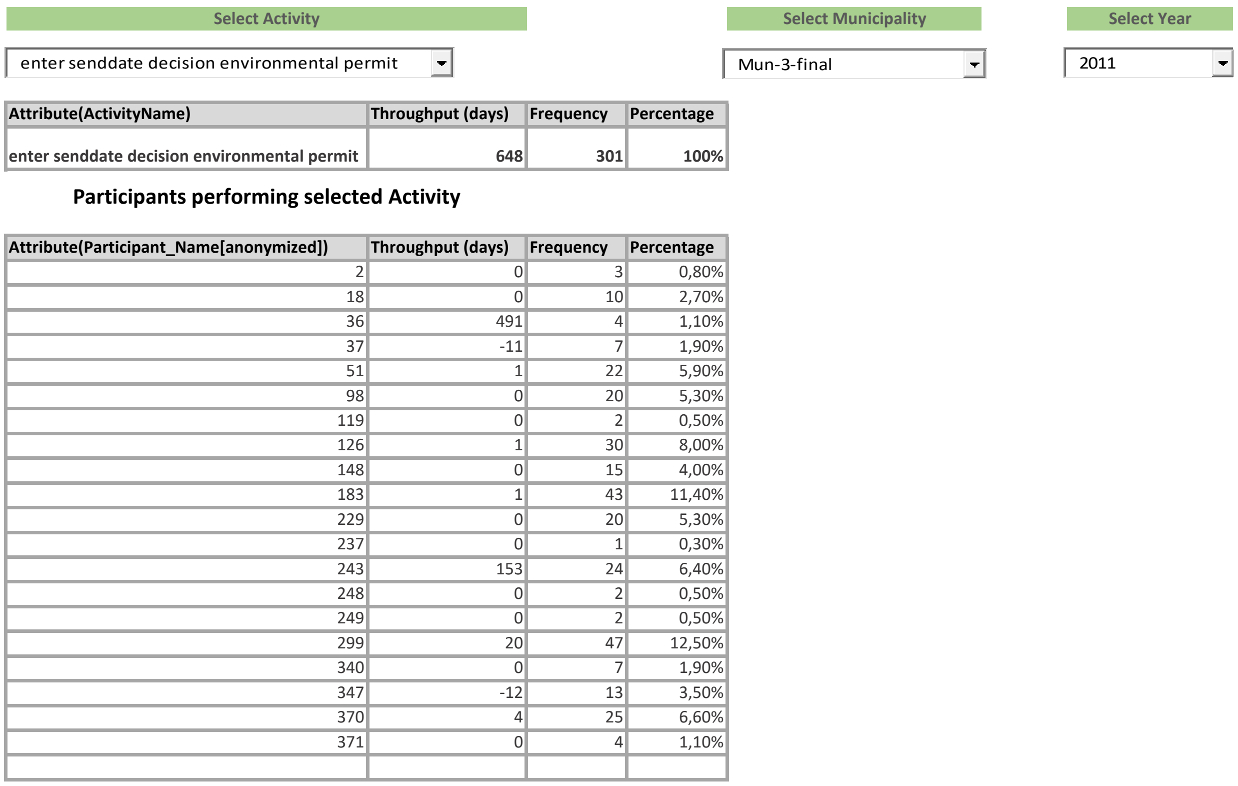
Select Municipality: Mun-3-final

Select Year: 2010

Figure 8.32: Aggregated measures for specific municipalities for specific year

In this figure, user can select which municipality by which year he/she wants to show results and the values on the row tables are automatically refreshed.

In the middle of the dashboard we show activity performance results which we give in more detailed in the following Figure 8.33. In this interactive spreadsheet user can select which specific activity he is interested from which municipality and for which year. We give results based on some measures such as *Throughput*, *Frequency*, *Percentage* for the selected activity. Afterwards, participants that perform the selected activities are listed, associated with some other information concerning the days spent for a specific participant to complete an activity, the relevant frequency and participant percentage.



Select Activity	Select Municipality	Select Year
enter senddate decision environmental permit	Mun-3-final	2011

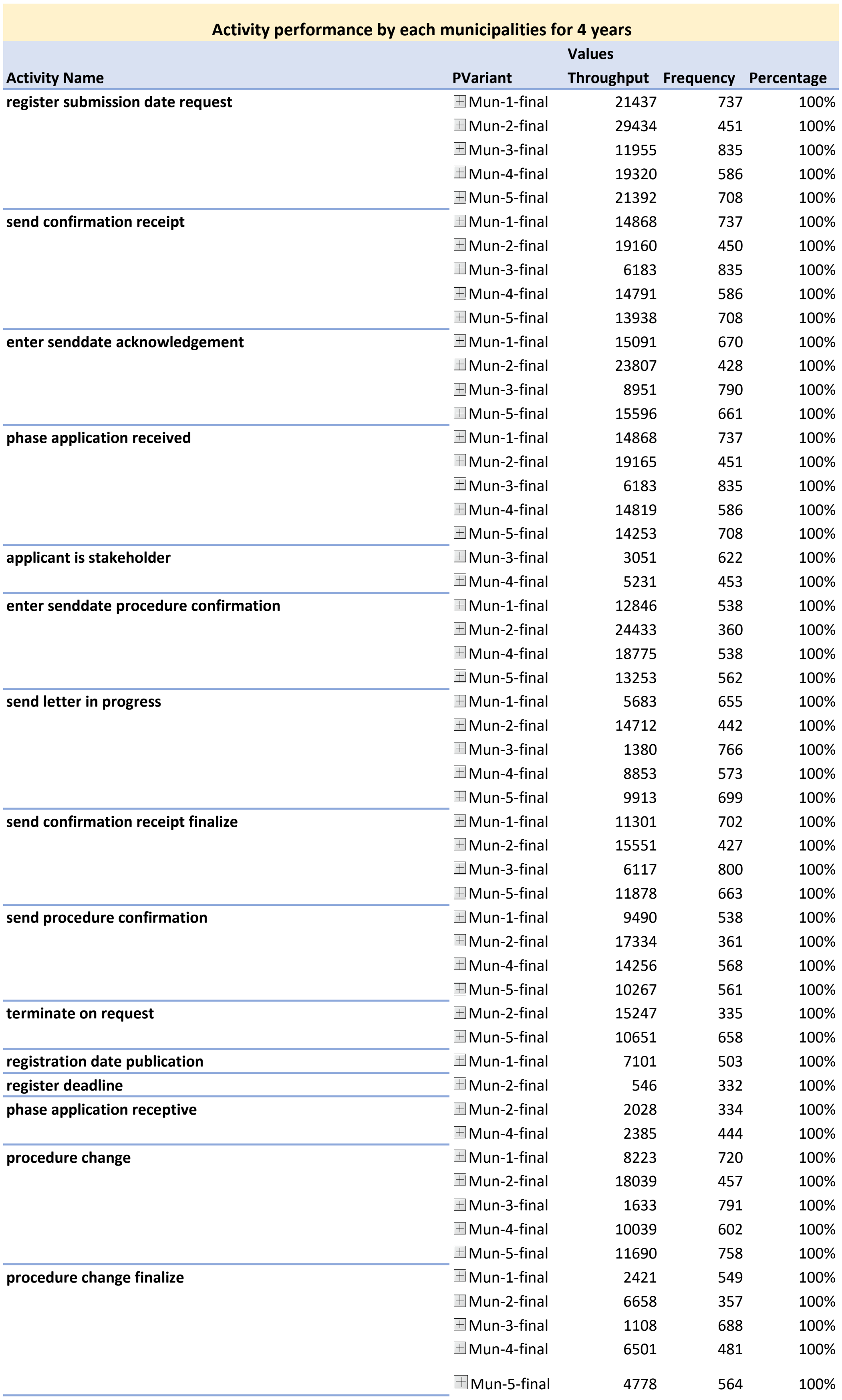
Attribute(ActivityName)	Throughput (days)	Frequency	Percentage
enter senddate decision environmental permit	648	301	100%

Participants performing selected Activity

Attribute(Participant_Name[anonymized])	Throughput (days)	Frequency	Percentage
2	0	3	0,80%
18	0	10	2,70%
36	491	4	1,10%
37	-11	7	1,90%
51	1	22	5,90%
98	0	20	5,30%
119	0	2	0,50%
126	1	30	8,00%
148	0	15	4,00%
183	1	43	11,40%
229	0	20	5,30%
237	0	1	0,30%
243	153	24	6,40%
248	0	2	0,50%
249	0	2	0,50%
299	20	47	12,50%
340	0	7	1,90%
347	-12	13	3,50%
370	4	25	6,60%
371	0	4	1,10%

Figure 8.33: Performance for the selected activity of a specific Municipality in a specific year

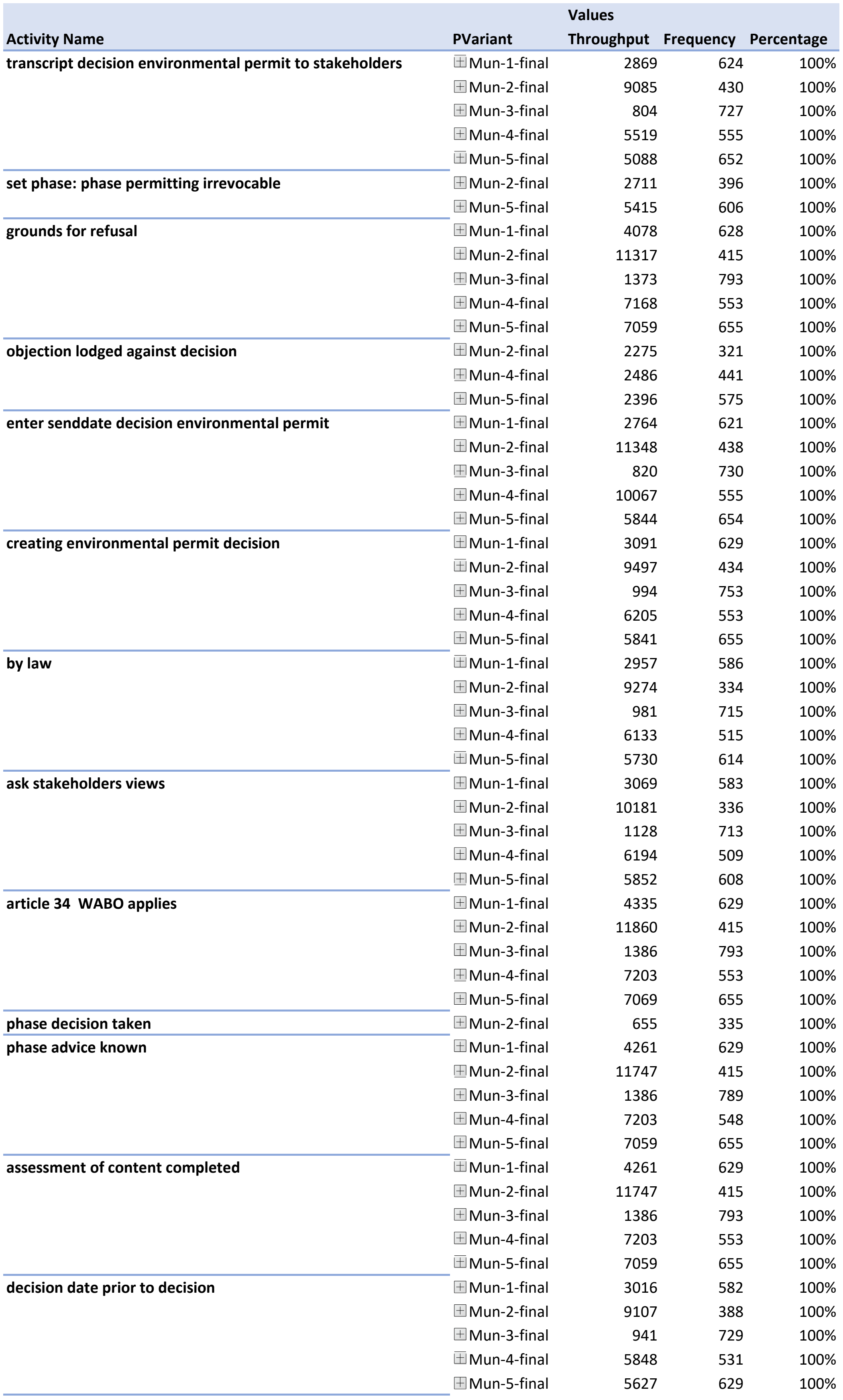
Whereas, in the following two consecutive figures: Figure 8.34 and Figure 8.35 we show all individual activity performance for each municipality for four years period.



Activity performance by each municipalities for 4 years

		Values		
Activity Name	**PVariant**	**Throughput**	**Frequency**	**Percentage**
register submission date request	Mun-1-final	21437	737	100%
	Mun-2-final	29434	451	100%
	Mun-3-final	11955	835	100%
	Mun-4-final	19320	586	100%
	Mun-5-final	21392	708	100%
send confirmation receipt	Mun-1-final	14868	737	100%
	Mun-2-final	19160	450	100%
	Mun-3-final	6183	835	100%
	Mun-4-final	14791	586	100%
	Mun-5-final	13938	708	100%
enter senddate acknowledgement	Mun-1-final	15091	670	100%
	Mun-2-final	23807	428	100%
	Mun-3-final	8951	790	100%
	Mun-5-final	15596	661	100%
phase application received	Mun-1-final	14868	737	100%
	Mun-2-final	19165	451	100%
	Mun-3-final	6183	835	100%
	Mun-4-final	14819	586	100%
	Mun-5-final	14253	708	100%
applicant is stakeholder	Mun-3-final	3051	622	100%
	Mun-4-final	5231	453	100%
enter senddate procedure confirmation	Mun-1-final	12846	538	100%
	Mun-2-final	24433	360	100%
	Mun-4-final	18775	538	100%
	Mun-5-final	13253	562	100%
send letter in progress	Mun-1-final	5683	655	100%
	Mun-2-final	14712	442	100%
	Mun-3-final	1380	766	100%
	Mun-4-final	8853	573	100%
	Mun-5-final	9913	699	100%
send confirmation receipt finalize	Mun-1-final	11301	702	100%
	Mun-2-final	15551	427	100%
	Mun-3-final	6117	800	100%
	Mun-5-final	11878	663	100%
send procedure confirmation	Mun-1-final	9490	538	100%
	Mun-2-final	17334	361	100%
	Mun-4-final	14256	568	100%
	Mun-5-final	10267	561	100%
terminate on request	Mun-2-final	15247	335	100%
	Mun-5-final	10651	658	100%
registration date publication	Mun-1-final	7101	503	100%
register deadline	Mun-2-final	546	332	100%
phase application receptive	Mun-2-final	2028	334	100%
	Mun-4-final	2385	444	100%
procedure change	Mun-1-final	8223	720	100%
	Mun-2-final	18039	457	100%
	Mun-3-final	1633	791	100%
	Mun-4-final	10039	602	100%
	Mun-5-final	11690	758	100%
procedure change finalize	Mun-1-final	2421	549	100%
	Mun-2-final	6658	357	100%
	Mun-3-final	1108	688	100%
	Mun-4-final	6501	481	100%
	Mun-5-final	4778	564	100%

Figure 8.34: Activity performance by each municipalities for four years period (1)



		Values		
Activity Name	PVariant	Throughput	Frequency	Percentage
transcript decision environmental permit to stakeholders	Mun-1-final	2869	624	100%
	Mun-2-final	9085	430	100%
	Mun-3-final	804	727	100%
	Mun-4-final	5519	555	100%
	Mun-5-final	5088	652	100%
set phase: phase permitting irrevocable	Mun-2-final	2711	396	100%
	Mun-5-final	5415	606	100%
grounds for refusal	Mun-1-final	4078	628	100%
	Mun-2-final	11317	415	100%
	Mun-3-final	1373	793	100%
	Mun-4-final	7168	553	100%
	Mun-5-final	7059	655	100%
objection lodged against decision	Mun-2-final	2275	321	100%
	Mun-4-final	2486	441	100%
	Mun-5-final	2396	575	100%
enter senddate decision environmental permit	Mun-1-final	2764	621	100%
	Mun-2-final	11348	438	100%
	Mun-3-final	820	730	100%
	Mun-4-final	10067	555	100%
	Mun-5-final	5844	654	100%
creating environmental permit decision	Mun-1-final	3091	629	100%
	Mun-2-final	9497	434	100%
	Mun-3-final	994	753	100%
	Mun-4-final	6205	553	100%
	Mun-5-final	5841	655	100%
by law	Mun-1-final	2957	586	100%
	Mun-2-final	9274	334	100%
	Mun-3-final	981	715	100%
	Mun-4-final	6133	515	100%
	Mun-5-final	5730	614	100%
ask stakeholders views	Mun-1-final	3069	583	100%
	Mun-2-final	10181	336	100%
	Mun-3-final	1128	713	100%
	Mun-4-final	6194	509	100%
	Mun-5-final	5852	608	100%
article 34 WABO applies	Mun-1-final	4335	629	100%
	Mun-2-final	11860	415	100%
	Mun-3-final	1386	793	100%
	Mun-4-final	7203	553	100%
	Mun-5-final	7069	655	100%
phase decision taken	Mun-2-final	655	335	100%
phase advice known	Mun-1-final	4261	629	100%
	Mun-2-final	11747	415	100%
	Mun-3-final	1386	789	100%
	Mun-4-final	7203	548	100%
	Mun-5-final	7059	655	100%
assessment of content completed	Mun-1-final	4261	629	100%
	Mun-2-final	11747	415	100%
	Mun-3-final	1386	793	100%
	Mun-4-final	7203	553	100%
	Mun-5-final	7059	655	100%
decision date prior to decision	Mun-1-final	3016	582	100%
	Mun-2-final	9107	388	100%
	Mun-3-final	941	729	100%
	Mun-4-final	5848	531	100%
	Mun-5-final	5627	629	100%

Figure 8.35: Activity performance by each municipalities for four years period (2)

To summarize all activity performance of different municipalities either by Year or by four years period we display them in respective figures Figure 8.36 and Figure 8.37. Moreover, we create charts based on the respective values.

Y-Q-M-D	2011

TotalDuration(days)	Column Labels				
Activity Name	Mun-1-final	Mun-2-final	Mun-3-final	Mun-4-final	Mun-5-final
register submission date request	10549	15455	5005	11313	12468
send confirmation receipt	6960	11859	2818	10937	9541
enter senddate acknowledgement	7740	12984	3792		11180
phase application received	6960	11859	2818	10965	9541
applicant is stakeholder			276	2353	
enter senddate procedure confirmation	7149	13517		12927	9758
send letter in progress	3405	11692	750	7784	7823
send confirmation receipt finalize	6033	9809	2806		8803
send procedure confirmation	4787	11383		11165	7339
terminate on request		11021			7738
registration date publication	1623				
register deadline		147			
phase application receptive		563		1248	
procedure change	4917	13173	883	7927	8574
procedure change finalize	1482	5387	750	5574	4265
transcript decision environmental permit to stakeholders	2444	8602	696	5009	4073
set phase: phase permitting irrevocable		2388			5166
grounds for refusal	2825	9437	750	6128	5630
objection lodged against decision		2275		1907	2396
enter senddate decision environmental permit	2078	8815	648	6769	4447
creating environmental permit decision	2479	8197	709	5626	4651
by law	2390	8213	709	5626	4561
ask stakeholders views	2427	9073	710	5721	4683
article 34 WABO applies	2624	9885	750	6163	5640
phase decision taken		147			
phase advice known	2546	9867	750	6163	5630
assessment of content completed	2546	9867	750	6163	5630
decision date prior to decision	2444	8346	703	5275	4481

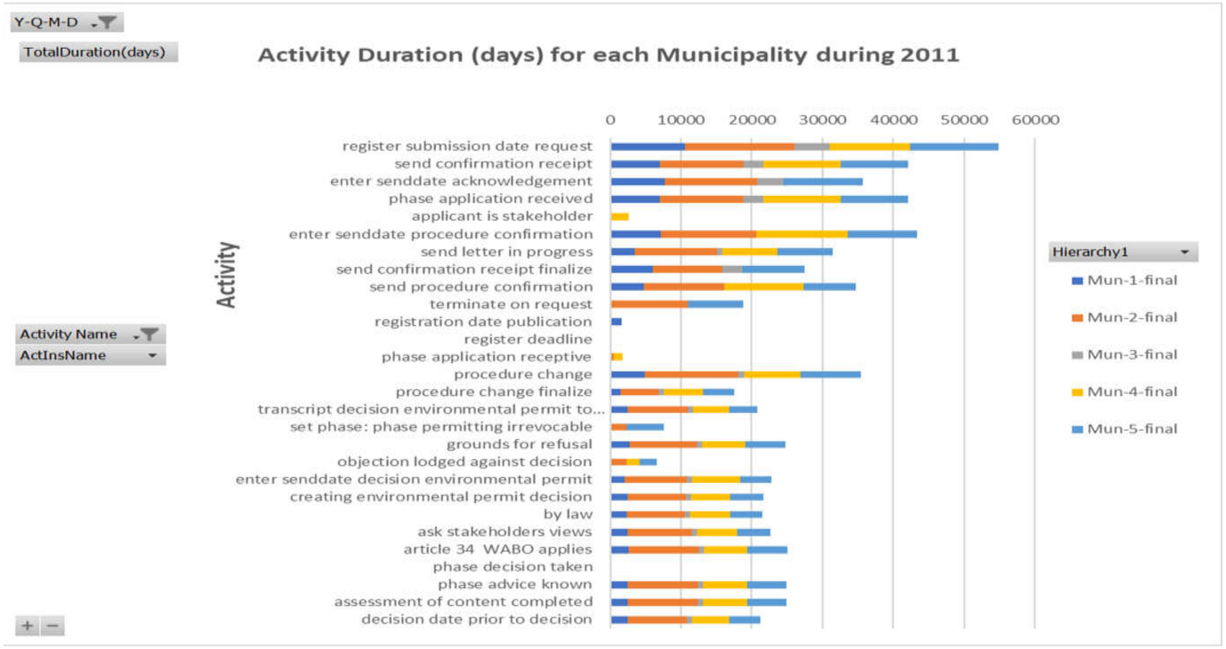


Figure 8.36: Activity performance measures of municipalities for specific year

TotalDuration(days)	Column Labels				
Activity Name	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**
register submission date request	21437	29434	11955	19320	21392
send confirmation receipt	14868	19160	6183	14791	13938
enter senddate acknowledgement	15091	23807	8951		15596
phase application received	14868	19165	6183	14819	14253
applicant is stakeholder			3051	5231	
enter senddate procedure confirmation	12846	24433		18775	13253
send letter in progress	5683	14712	1380	8853	9913
send confirmation receipt finalize	11301	15551	6117		11878
send procedure confirmation	9490	17334		14256	10267
terminate on request		15247			10651
registration date publication	7101				
register deadline		546			
phase application receptive		2028		2385	
procedure change	8223	18039	1633	10039	11690
procedure change finalize	2421	6658	1108	6501	4778
transcript decision environmental permit to stakeholders	2869	9085	804	5519	5088
set phase: phase permitting irrevocable		2711			5415
grounds for refusal	4078	11317	1373	7168	7059
objection lodged against decision		2275		2486	2396
enter senddate decision environmental permit	2764	11348	820	10067	5844
creating environmental permit decision	3091	9497	994	6205	5841
by law	2957	9274	981	6133	5730
ask stakeholders views	3069	10181	1128	6194	5852
article 34 WABO applies	4335	11860	1386	7203	7069
phase decision taken		655			
phase advice known	4261	11747	1386	7203	7059
assessment of content completed	4261	11747	1386	7203	7059
decision date prior to decision	3016	9107	941	5848	5627

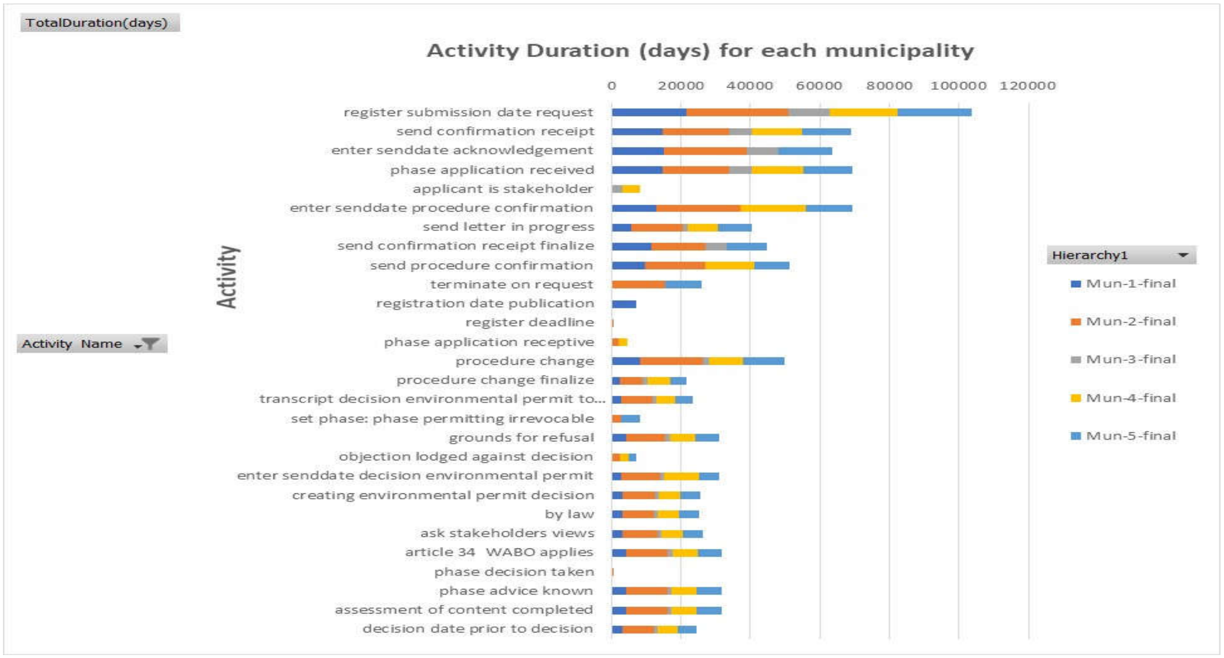
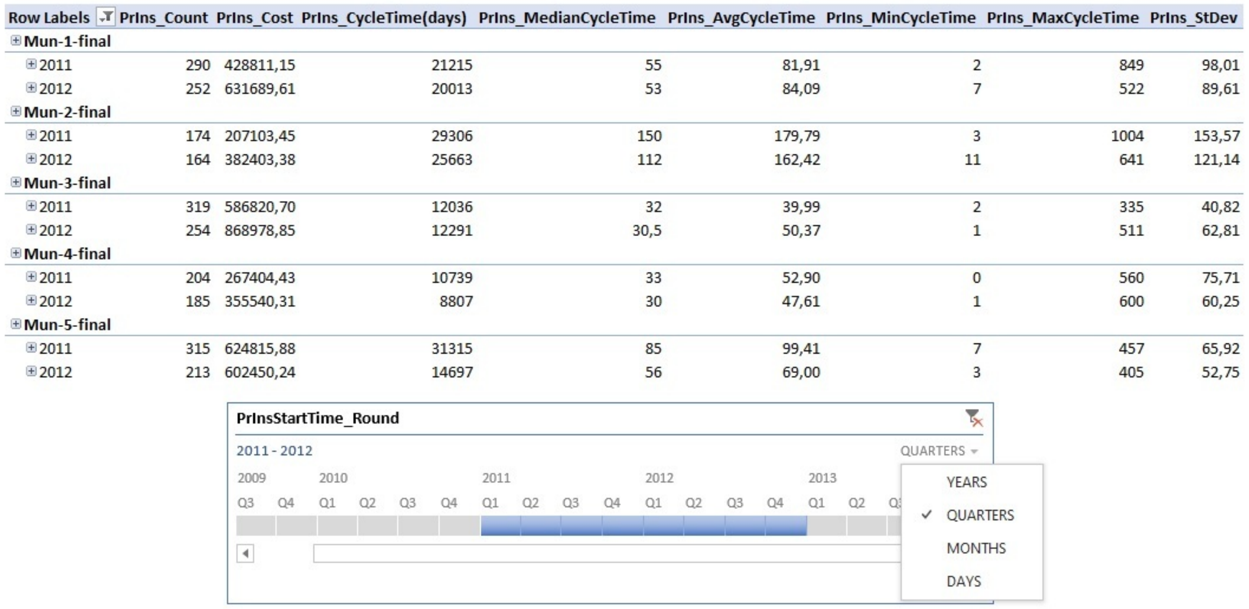


Figure 8.37: Activity performance measures of municipalities for four years period

And yet another convenient way on showing results is by using a timeline slicer as shown in Figure 8.38.



Row Labels	PrIns_Count	PrIns_Cost	PrIns_CycleTime(days)	PrIns_MedianCycleTime	PrIns_AvgCycleTime	PrIns_MinCycleTime	PrIns_MaxCycleTime	PrIns_StDev
Mun-1-final								
2011	290	428811,15	21215	55	81,91	2	849	98,01
2012	252	631689,61	20013	53	84,09	7	522	89,61
Mun-2-final								
2011	174	207103,45	29306	150	179,79	3	1004	153,57
2012	164	382403,38	25663	112	162,42	11	641	121,14
Mun-3-final								
2011	319	586820,70	12036	32	39,99	2	335	40,82
2012	254	868978,85	12291	30,5	50,37	1	511	62,81
Mun-4-final								
2011	204	267404,43	10739	33	52,90	0	560	75,71
2012	185	355540,31	8807	30	47,61	1	600	60,25
Mun-5-final								
2011	315	624815,88	31315	85	99,41	7	457	65,92
2012	213	602450,24	14697	56	69,00	3	405	52,75

PrInsStartTime_Round

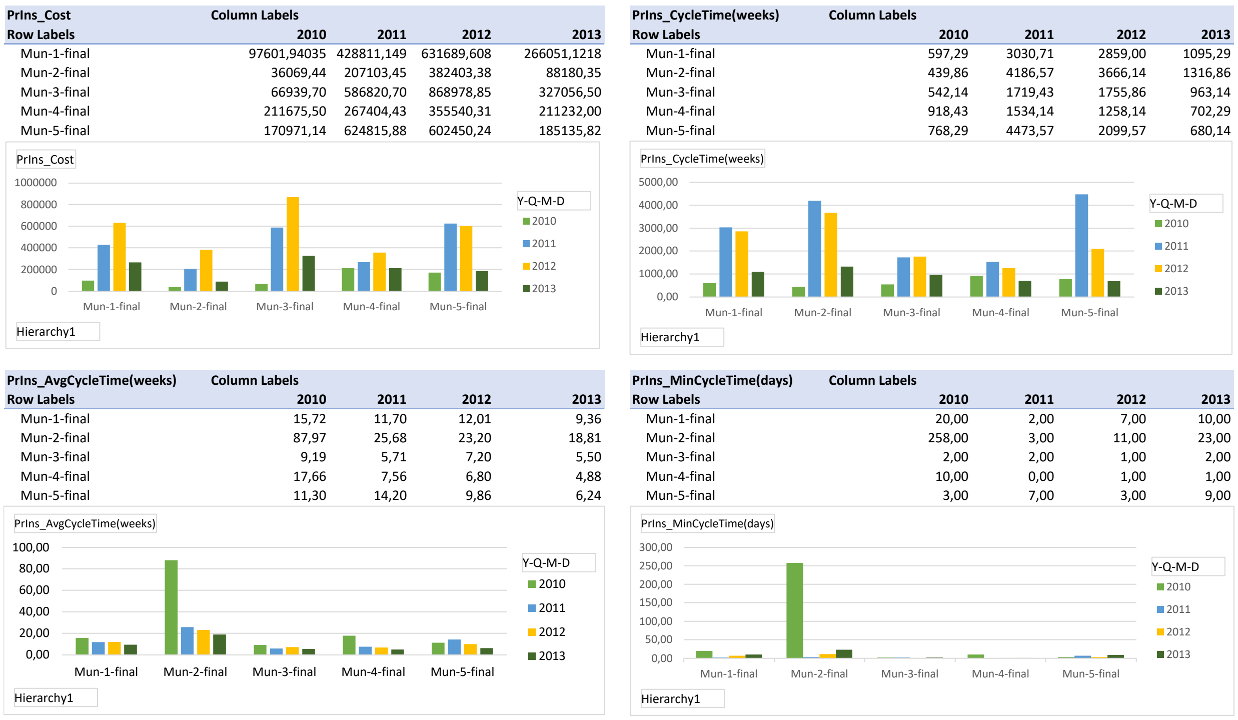
2011 - 2012

QUARTERS

Figure 8.38: Aggregated measures for specific municipalities for specific dates using a timeline slicer

Here, user can filter the *Y-Q-M-D* hierarchy button to select which attribute he/she wants to show for a datetime and then select which parts of the slider he is interested to show results of. For instance, in the figure the slider is divided by 4 quarters per year as we filter the data based on *Quarter* attribute.

And last figure with charts positioned in the bottom of the aforementioned dashboard displays tables and respective charts which report some useful information on time and cost of each municipality process for specific years. These charts are shown in Figure 8.39. User can filter also on a single municipality or different multiple items and for which date time, i. e., by Year, Quarter, Month or Days value. And based on what the user wants to select the respective charts will be generated automatically.



PrIns_Cost	Column Labels			
Row Labels	2010	2011	2012	2013
Mun-1-final	97601,94035	428811,149	631689,608	266051,1218
Mun-2-final	36069,44	207103,45	382403,38	88180,35
Mun-3-final	66939,70	586820,70	868978,85	327056,50
Mun-4-final	211675,50	267404,43	355540,31	211232,00
Mun-5-final	170971,14	624815,88	602450,24	185135,82

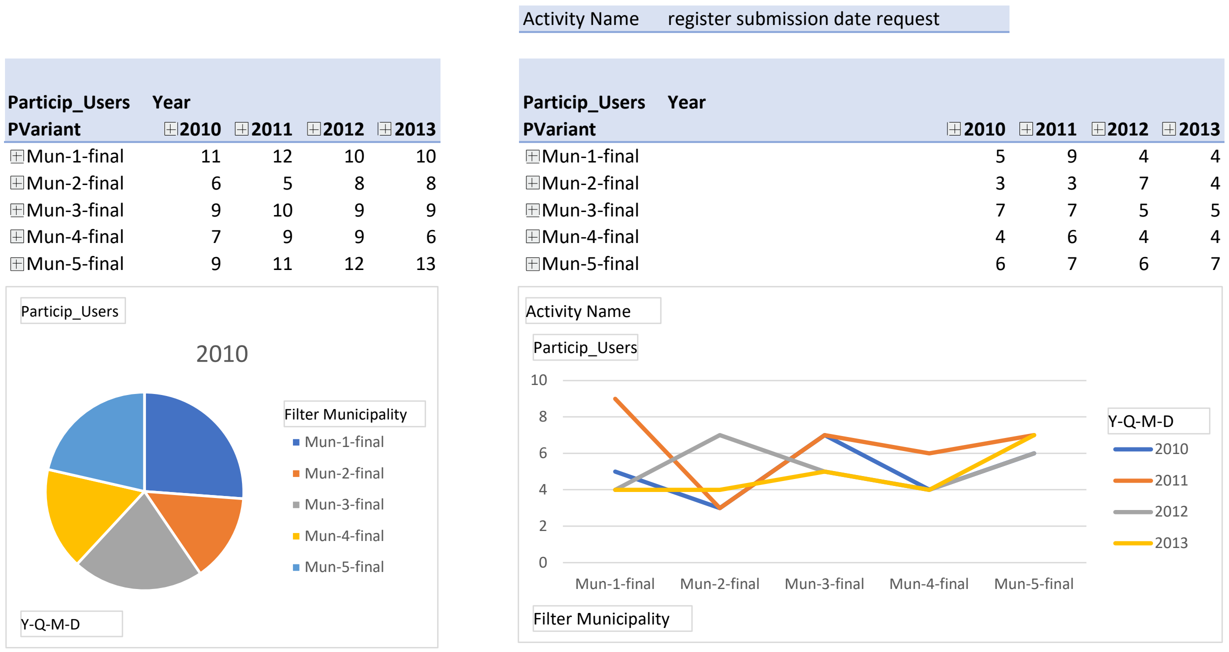
PrIns_CycleTime(weeks)	Column Labels			
Row Labels	2010	2011	2012	2013
Mun-1-final	597,29	3030,71	2859,00	1095,29
Mun-2-final	439,86	4186,57	3666,14	1316,86
Mun-3-final	542,14	1719,43	1755,86	963,14
Mun-4-final	918,43	1534,14	1258,14	702,29
Mun-5-final	768,29	4473,57	2099,57	680,14

PrIns_AvgCycleTime(weeks)	Column Labels			
Row Labels	2010	2011	2012	2013
Mun-1-final	15,72	11,70	12,01	9,36
Mun-2-final	87,97	25,68	23,20	18,81
Mun-3-final	9,19	5,71	7,20	5,50
Mun-4-final	17,66	7,56	6,80	4,88
Mun-5-final	11,30	14,20	9,86	6,24

PrIns_MinCycleTime(days)	Column Labels			
Row Labels	2010	2011	2012	2013
Mun-1-final	20,00	2,00	7,00	10,00
Mun-2-final	258,00	3,00	11,00	23,00
Mun-3-final	2,00	2,00	1,00	2,00
Mun-4-final	10,00	0,00	1,00	1,00
Mun-5-final	3,00	7,00	3,00	9,00

Figure 8.39: Time and cost related measures for municipalities and their respective charts

Mun-3 reports the best performance based on time aggregated measures whereas *Mun-2* the worst one for four years period. Instead, *Mun-3*report the highest value for process cost whereas *Mun-2* the lowest value, followed by *Mun-4* and *Mun-1* for the overall period of 4 years. The minimum cycle time is recorded for *Mun-3* and the greatest value for this measure instead is recorded for *Mun-2* specifically for the year:2010.
Moreover, in our approach user has the possibility to check the allocation of participants performing different activities in a particular process. Therefore, in Figure 8.40 we give results on how each municipality is handling the utilization of their personnel. Here, in the left part we show results about number of participants assigned to perform activities for each municipality and for specific years. We should stress we refer to participants here by distinct users not the number of participants with a specific role assigned. Visual values can be represented through a pie chart as given in the figure; user can filter for a specific year. Whereas, in the right part we drop another filter field, i. e. attribute *ActivityName* (e. g., register submission request activity in the filtered field) to show the relevant number of participants executing the filtered activity for each of the municipalities. Also, a line chart represents visually the values in the table. Based on these values, *Mun-1* has the greatest nr of distinct users assigned during year:2011 and *Mun-2* has the lowest ones instead. Based on four years period we can say that *Mun-5* and *Mun-1* reports the greatest nr of users performing activities; followed by *Mun-3* and then *Mun-4*; whereas *Mun-2* has again the lowest values. We have to highlight that *Mun-3* reports the highest percentage of the participants, i. e., more than 100%, which means that even though they do not rely on a larger number of personnel (users) they still outperform their-selves in their responsibility of role assignments to execute different activities.



Activity Name register submission date request

Particip_Users PVariant	2010	2011	2012	2013
Mun-1-final	11	12	10	10
Mun-2-final	6	5	8	8
Mun-3-final	9	10	9	9
Mun-4-final	7	9	9	6
Mun-5-final	9	11	12	13

Particip_Users PVariant	2010	2011	2012	2013
Mun-1-final	5	9	4	4
Mun-2-final	3	3	7	4
Mun-3-final	7	7	5	5
Mun-4-final	4	6	4	4
Mun-5-final	6	7	6	7

Figure 8.40: Number of participants

According to process and activity performance of each municipalities we can conclude the following:

- *Mun-3* has the best performance on time related measures, a moderated number of users but utilized more than 100%
- *Mun-5* has a good performance on time related measures, the highest number of users working in the municipality but not utilized in their maximum capabilities
- *Mun-1* has a moderated performance on time related measures as well, a considerable number of users but not fully utilized during all years.
- *Mun-4* not so good performance on time related measures, a low number of users assigned to perform the work they are responsible to.
- *Mun-2* very low performance on time related measures, the lowest number of assigned users

In the next and final section we give conclusions and suggestions for further improvements on our prototype.

CHAPTER 9

Conclusions and Future Work

Chapter 9 concludes and summarises all contributions of this research work.

This thesis proposed a process warehouse approach to efficiently and effectively analyse a family of process variants. Process variants are defined as sets of similar process models that may evolve over time because of the adjustments made to the same business process in a given domain. In our study we considered two different scenarios of these variants: the customer invoice payment process variants as our motivation example and the building permit application process variants as a real-life case study. In practice a large number of variants occurs increasing thus modelling and maintaining efforts which is both time-consuming and error-prone task. Current business process management systems and traditional process warehouses lack on adequately abstracting and consolidating all variants into one generic process model, to provide the possibility to distinguish and compare among different parts of different variants. This shortcoming affects decision making of business analysts for a specific process context.

As a consequence, analysing and comparing these multiple variants within a common IT system becomes quite hard for process designers.

To overcome these shortcomings we established first a correct way to capture design-time of process model variants and secondly built a process warehouse model to analyse the event logs during runtime execution of the respective variants.

Accordingly, we specifically presented a meta-model to capture process variants by introducing two new notions of *generic activities* and *generic processes* and to define specialization/generalization relationships between them. The *generic activity* is defined as a step in a process that might be realized by different activities. These step activities within our model are determined by the presence of a boolean attribute. The realized activities may be single activities or complex activities. To annotate the relationship between these activities we introduced a stereotype *«variant_specialization»*. And, a *generic process* (GP) is defined as a process that contains at least one generic activity. Whereas, a concrete process (CP) is modelled disjointly of a GP, i. e., cannot contain any GA.

In addition, the specialization between processes are derived based on the substitution/realization of the generic activities with one of its specializations.

We classified the approach in the group of *activity specialization* based on the technique proposed to capture variability from existing process models. These techniques proposed by the research communities were compared based on different criteria from literature review.

We built a reference (global) process model where generic processes and generic activities are captured and concrete process variants can be derived. We limit our development to capture variability mainly in the process control-flow and do not consider other aspects such as resources or other business objects. Even though these other aspects are recorded only in the event logs during process enactment. Other additional elements are captured from organizational perspective (i. e., resource perspective) to express how users (i. e., participants) responsible for specific tasks are positioned in a structured organization unit. Other important elements from run-time perspective are captured as well in our model, i. e., the information about instances of different process variants after executing them on top of a BPMS.

A relational database was generated from the meta-model after executing extract and load operations. We wrote algorithms to derive process specializations based on the specialization relationships between generic activities and activities.

Afterwards, we built a process warehouse with a generalization hierarchy between activities and processes. Such generalization hierarchies for processes is essential to structure the "process" dimension of process warehouses, which then can be used to analyse process metrics with the usual OLAP operations such as to roll-up and drill down the dimension hierarchy. In particular it allowed to analyse variants of the same process as individuals together, or partitioned in similar groups.

We implemented a prototype solution to establish the practical feasibility of our process warehouse design and used a business intelligence tool to perform powerful data analysis by defining custom calculated measures. We summarized these measures and their corresponding charts on interactive dashboards. Thus, different type of queries on process variants that are interesting for business users are answered and can be revealed through these dashboards. Therefore, typical OLAP operations could be performed, e. g., roll-up or drill-down and key-performance indicators (KPIs) could be computed or *conditional-if-rules* could be applied to compare between different variants at different levels of genericity. Moreover, we verified that our solution can answer as well other queries on a standalone process provided by conventional BPM systems.

The experiences suggested that the framework was feasible to manage a practical variability case study involving different variation points to distinguish between different parts of different variants using aggregated measures.

In summary, based on the consumption of process-oriented data warehouse in many business intelligence development and solutions, a framework that allows process variants to be efficiently and effectively analysed can significantly improve the state-of-art.

There are some possible extension to this study. A possible extension consists in comparing between actual and intended behaviour of a process variant. As a starting point we can find number of process instances (from a variant log exe)- *"as-is behaviour"* that *match* and/or *deviate from* process paths (instance types)- *"to-be behaviour"*.

Another direction for future work consists of investigating the automation of configuring specialized activities from a generic activity. Thus, the design efforts needed to model these generic activities and the respective specialized ones can be reduced.

A further improvement consists of moving the development of the framework to an open-source solution.

Finally, other tests need to be conducted to improve the evidence base of the evaluated study.

Bibliography

Awad, A. and Sakr, S. On efficient processing of bpmn-q queries. *Computers in Industry*, 63(9):867 – 881, 2012. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2012.06.002. URL `http://www.sciencedirect.com/science/article/pii/S0166361512001169`.

Ayora, C., Torres, V., Weber, B., Reichert, M., and Pelechano, V. Enhancing modeling and change support for process families through change patterns. In Nurcan, S., Proper, H. A., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T., and Bider, I., editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 246–260, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38484-4.

Ayora, C., Torres, V., Weber, B., Reichert, M., and Pelechano, V. Vivace: A framework for the systematic evaluation of variability support in process-aware information systems. *Information and Software Technology*, 57:248 – 276, 2015. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2014.05.009. URL `http://www.sciencedirect.com/science/article/pii/S0950584914001268`.

Ayora, C., Torres, V., de la Vara, J. L., and Pelechano, V. Variability management in process families through change patterns. *Information and Software Technology*, 74:86 – 104, 2016. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2016.01.007. URL `http://www.sciencedirect.com/science/article/pii/S0950584916000094`.

Bayer, J., Buhl, W., Giese, C., Lehner, T., Ocampo, A., Puhlmann, F., Richter, E., Schnieders, A., Weiland, J., and Weske, M. Process family engineering: Modeling variant-rich processes. Technical report, DaimlerChrysler Research and Technology, Delta Software Technology, Fraunhofer IESE, Hasso-PlattnerInstitute, 2005.

Becker, M. and Laue, R. A comparative survey of business process similarity measures. *Comput. Ind.*, 63(2):148–167, February 2012. ISSN 0166-3615. doi: 10.1016/j.compind.2011.11.003. URL `http://dx.doi.org/10.1016/j.compind.2011.11.003`.

Benker, T. A generic process data warehouse schema for bpmn workflows. In Abramowicz, W., Alt, R., and Franczyk, B., editors, *Business Information Systems: 19th International Conference, BIS 2016, Leipzig, Germany, July, 6-8, 2016, Proceedings*, pages 222–234. Springer International Publishing, Cham, 2016. ISBN 978-3-319-39426-8. doi: 10.1007/978-3-319-39426-8_18. URL `https://doi.org/10.1007/978-3-319-39426-8_18`.

Berberi, L., Eder, J., and Koncilia, C. A process warehouse model capturing process variants. *Enterprise Modelling and Information Systems Architectures International*

Journal of Conceptual Modeling, 13(1):77–85, Feb 2018. ISSN 1866-3621. doi: https://doi.org/10.18417/emisa.si.hcm.8.

Böhnlein, M. and Ulbrich-vom Ende, A. Business process oriented development of data warehouse structures. In Jung, R. and Winter, R., editors, *Data Warehousing 2000*, pages 3–21, Heidelberg, 2000. Physica-Verlag HD. ISBN 978-3-642-57681-2.

Bolt, A. and van der Aalst, W. M. P. Multidimensional process mining using process cubes. In Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., and Ma, Q., editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 102–116, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19237-6.

Bonifati, A., Casati, F., Dayal, U., and Shan, M.-C. Warehousing workflow data: Challenges and opportunities. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 649–652, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-804-4. URL `http://dl.acm.org/citation.cfm?id=645927.672212`.

Bose, R. P. J. C., van der Aalst, W. M. P., Žliobaitė, I., and Pechenizkiy, M. Handling concept drift in process mining. In Mouratidis, H. and Rolland, C., editors, *Advanced Information Systems Engineering*, pages 391–405, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21640-4.

BPMN(Spec.). About the business process model and notation specification version 2.0. Technical report, Object Management Group, https://www.omg.org/spec/BPMN/2.0/PDF, 2011. Last accessed: May 2018.

Brandt, S. C., Schlüter, M., and Jarke, M. Process data warehouse models for cooperative engineering processes. *Proceedings of the 9th IFAC Symposium on Automated Systems Based on Human Skill And Knowledge*, 39(4):219–224, 2006.

Casati, F., Castellanos, M., Dayal, U., and Salazar, N. A generic solution for warehousing business process data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 1128–1137. VLDB Endowment, 2007. ISBN 978-1-59593-649-3. URL `http://dl.acm.org/citation.cfm?id=1325851.1325978`.

Chaudhuri, S. and Dayal, U. An overview of data warehousing and olap technology. *SIGMOD Rec.*, 26(1):65–74, March 1997. ISSN 0163-5808. doi: 10.1145/248603.248616. URL `http://doi.acm.org/10.1145/248603.248616`.

Conforti, R., Dumas, M., La Rosa, M., Maaradji, A., Nguyen, H. H., Ostovar, A., and Raboczi, S. Analysis of business process variants in apromore. In *Proceedings of the Demo Track of the 13th International Conference on Business Process Management (BPM'15)*, number 1418 in CEUR Workshop Proceedings, 2015. URL `http://ceur-ws.org/Vol-1418/paper4.pdf`.

Czarnecki, K. and Antkiewicz, M. Mapping features to models: A template approach based on superimposed variants. In Glück, R. and Lowry, M., editors, *Generative Programming and Component Engineering*, pages 422–437, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31977-1.

de Medeiros, A. K. A., van der Aalst, W. M. P., and Weijters, A. J. M. M. Workflow mining: Current status and future directions. In Meersman, R., Tari, Z., and Schmidt, D. C., editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 389–406, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39964-3.

de Medeiros, A. K. A., Weijters, A. J. M. M., and van der Aalst, W. M. P. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, Apr 2007. ISSN 1573-756X. doi: 10.1007/s10618-006-0061-7. URL `https://doi.org/10.1007/s10618-006-0061-7`.

Dőhring, M. and Zimmermann, B. vbpmn: Event-aware workflow variants by weaving bpmn2 and business rules. In Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., and Bider, I., editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 332–341, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21759-3.

Dőhring, M., Reijers, H. A., and Smirnov, S. Configuration vs. adaptation for business process variant maintenance: An empirical study. *Information Systems*, 39:108 – 133, 2014. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2013.06.002. URL `http://www.sciencedirect.com/science/article/pii/S0306437913000811`.

Dijkman, R., Dumas, M., Garcia-Banuelos, L., and Kaarik, R. Aligning business process models. In *2009 IEEE International Enterprise Distributed Object Computing Conference*, pages 45–53, Sept 2009. doi: 10.1109/EDOC.2009.11.

Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., and Mendling, J. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498 – 516, 2011a. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2010.09.006. URL `http://www.sciencedirect.com/science/article/pii/S0306437910001006`. Special Issue: Semantic Integration of Data, Multimedia, and Services.

Dijkman, R., Gfeller, B., Küster, J., and Völzer, H. Identifying refactoring opportunities in process model repositories. *Information and Software Technology*, 53(9):937–948, September 2011b. ISSN 0950-5849. doi: 10.1016/j.infsof.2011.04.001. URL `http://dx.doi.org/10.1016/j.infsof.2011.04.001`.

Dijkman, R., Rosa, M. L., and Reijers, H. A. Managing large collections of business process models —current techniques and challenges. *Computers in Industry*, 63(2):91 – 97, 2012. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2011.12.003. URL `http://www.sciencedirect.com/science/article/pii/S0166361511001369`. Managing Large Collections of Business Process Models.

Dongen, V. and Boudewijn, B. Bpi challenge 2015. doi: https://data.4tu.nl/collections/BPI_Challenge_2015/5065424.

Dumas, M., García-Bañuelos, L., Rosa, M. L., and Uba, R. Fast detection of exact clones in business process model repositories. *Information Systems*, 38(4):619 – 633, 2013. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2012.07.002. URL http://www.sciencedirect.com/science/article/pii/S0306437912000993. Special section on BPM 2011 conference.

Eder, J., Olivotto, G. E., and Gruber, W. A data warehouse for workflow logs. In *Proc. Int. Conf. on Engineering and Deployment of Cooperative Information Systems*, EDCIS '02, pages 1–15, London, UK, UK, 2002. Springer. ISBN 3-540-44222-7. URL http://dl.acm.org/citation.cfm?id=646145.678736.

Ekanayake, C. C., La Rosa, M., ter Hofstede, A. H. M., and Fauvet, M.-C. Fragment-based version management for repositories of business process models. In Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D. C., White, J., Hauswirth, M., Hitzler, P., and Mohania, M., editors, *On the Move to Meaningful Internet Systems: OTM 2011*, pages 20–37, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25109-2.

Ekanayake, C. C., Dumas, M., García-Bañuelos, L., La Rosa, M., and ter Hofstede, A. H. M. Approximate clone detection in repositories of business process models. In Barros, A., Gal, A., and Kindler, E., editors, *Business Process Management*, pages 302–318, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-32885-5.

Fabio, C., Ming-Chien, S., Li-Jie, J., Umeshwar, D., Daniela, G., and Angela, B. Method of identifying and analyzing business processes from workflow audit logs. Patent, 2002.

Factor, P. 2014. URL https://www.red-gate.com/simple-talk/blogs/string-comparisons-in-sql-the-longest-common-substring/.

Fahland, D. and van der Aalst, W. M. P. Repairing process models to reflect reality. In *Proceedings of the 10th International Conference on Business Process Management*, BPM'12, pages 229–245, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-32884-8. doi: 10.1007/978-3-642-32885-5_19. URL http://dx.doi.org/10.1007/978-3-642-32885-5_19.

Fettke, P. and Loos, P. Classification of reference models: a methodology and its application. *Information Systems and e-Business Management*, 1(1):35–53, Jan 2003. ISSN 1617-9854. doi: 10.1007/BF02683509. URL https://doi.org/10.1007/BF02683509.

Giorgini, P., Rizzi, S., and Garzetti, M. Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP*, DOLAP '05, pages 47–56, New York, NY, USA, 2005. ACM. ISBN 1-59593-162-7. doi: 10.1145/1097002.1097011. URL http://doi.acm.org/10.1145/1097002.1097011.

Golfarelli, M. and Rizzi, S. *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2009. ISBN 0071610391, 9780071610391.

Gottschalk, F., van der Aalst, W. M. P., and Jansen-Vullers, M. H. Merging event-driven process chains. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems: OTM 2008*, pages 418–426, Berlin, Heidelberg, 2008a. Springer Berlin Heidelberg. ISBN 978-3-540-88871-0.

Gottschalk, F., Van Der Aalst, W. M., Jansen-Vullers, M. H., and La Rosa, M. Configurable workflow models. *International Journal of Cooperative Information Systems*, 17(02):177–221, 2008b.

Grigori, D., Casati, F., Dayal, U., and Shan, M.-C. Improving business process quality through exception understanding, prediction, and prevention. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 159–168, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-804-4. URL `http://dl.acm.org/citation.cfm?id=645927.672190`.

Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., and Shan, M.-C. Business process intelligence. *Computers in Industry*, 53(3):321 – 343, 2004. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2003.10.007. URL `http://www.sciencedirect.com/science/article/pii/S0166361503001994`. Process / Workflow Mining.

Günther, C. W. and van der Aalst, W. M. P. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In Alonso, G., Dadam, P., and Rosemann, M., editors, *Business Process Management*, pages 328–343, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-75183-0.

Gunther, C., Rinderle-Ma, S., Reichert, M., Van Der Aalst, W., and Recker, J. Using process mining to learn from process change in evolutionary. *International Journal of Business Process Integration and Management*, 3(1):61–78, 2008.

Guo, J. and Zou, Y. Detecting clones in business applications. In *2008 15th Working Conference on Reverse Engineering*, pages 91–100, Oct 2008. doi: 10.1109/WCRE.2008.12.

Hallerbach, A., Bauer, T., and Reichert, M. Managing process variants in the process lifecycle. In *Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS'08)*, pages 154–161, Barcelona, Spain, 2008. Springer.

Hallerbach, A., Bauer, T., and Reichert, M. Capturing variability in business process models: the provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546, 2010. doi: https://dl.acm.org/citation.cfm?id=1870514.

ISO/IEC-19510:2013. Technical report, International Organization for Standardization, https://www.iso.org/standard/62652.html, July . Last accessed: May 2018.

Jarke, M., List, T., and Köller, J. The challenge of process data warehousing. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 473–483, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-715-3. URL `http://dl.acm.org/citation.cfm?id=645926.671688`.

Jin, T., Wang, J., and Wen, L. Querying business process models based on semantics. In Yu, J. X., Kim, M. H., and Unland, R., editors, *Database Systems for Advanced Applications*, pages 164–178, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-20152-3.

Jin, T., Wang, J., Rosa, M. L., ter Hofstede, A., and Wen, L. Efficient querying of large process model repositories. *Computers in Industry*, 64(1):41 – 49, 2013. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2012.09.008. URL `http://www.sciencedirect.com/science/article/pii/S0166361512001455`.

Kimball, R. and Ross, M. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling.* Wiley Publishing, 3rd edition, 2013. ISBN 1118530802.

Koncilia, C., Pichler, H., and Wrembel, R. A generic data warehouse architecture for analyzing workflow logs. In *Advances in Databases and Information Systems*, pages 106–119. Springer, 2015.

Kueng, P., Wettstein, T., and List, B. A holistic process performance analysis through a performance data warehouse. In *Proceedings of the American Conference on Information Systems (AMCIS'2001)*, pages 349–356, 2001.

Kulkarni, V. and Barat, S. Business process families using model-driven techniques. In zur Muehlen, M. and Su, J., editors, *Business Process Management Workshops*, pages 314–325, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-20511-8.

Kurniawan, T. A., Ghose, A. K., Lê, L.-S., and Dam, H. K. On formalizing inter-process relationships. In Daniel, F., Barkaoui, K., and Dustdar, S., editors, *Business Process Management Workshops*, pages 75–86, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-28115-0.

La Rosa, M., Dumas, M., Uba, R., and Dijkman, R. Merging business process models. In Meersman, R., Dillon, T., and Herrero, P., editors, *On the Move to Meaningful Internet Systems: OTM 2010*, pages 96–113, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-16934-2.

La Rosa, M., Dumas, M., Uba, R., and Dijkman, R. Business process model merging: An approach to business process consolidation. *ACM Trans. Softw. Eng. Methodol.*, 22(2):11:1–11:42, March 2013. ISSN 1049-331X. doi: 10.1145/2430545.2430547. URL `http://doi.acm.org/10.1145/2430545.2430547`.

Li, C., Reichert, M., and Wombacher, A. Mining process variants: Goals and issues. In *2008 IEEE International Conference on Services Computing*, volume 2, pages 573–576. IEEE, July 2008. doi: 10.1109/SCC.2008.103.

Li, C., Reichert, M., and Wombacher, A. Mining business process variants: Challenges, scenarios, algorithms. *Data & Knowledge Engineering*, 70(5):409 – 434, 2011. ISSN 0169-023X. doi: https://doi.org/10.1016/j.datak.2011.01.005. URL `http://www.sciencedirect.com/science/article/pii/S0169023X11000127`. Business Process Management 2009.

List, B. and Korherr, B. An evaluation of conceptual business process modelling languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 1532–1539, New York, NY, USA, 2006. ACM. ISBN 1-59593-108-2. doi: 10.1145/1141277.1141633. URL `http://doi.acm.org/10.1145/1141277.1141633`.

List, B., Schiefer, J., and Tjoa, A. M. The process warehouse: A data warehouse approach for multidimensional business process analysis and improvement. In Jung, R. and Winter, R., editors, *Data Warehousing 2000*, pages 267–282, Heidelberg, 2000. Physica, Heidelberg. ISBN 978-3-642-57681-2.

List, B., Bruckner, R. M., Machaczek, K., and Schiefer, J. A comparison of data warehouse development methodologies case study of the process warehouse. In Hameurlain, A., Cicchetti, R., and Traunmüller, R., editors, *Database and Expert Systems Applications*, pages 203–215, Berlin, Heidelberg, 2002a. Springer Berlin Heidelberg. ISBN 978-3-540-46146-3.

List, B., Schiefer, J., Tjoa, A. M., and Quirchmayr, G. Multidimensional business process analysis with the process warehouse. In *Knowledge Discovery for Business Information Systems*, pages 211–227. Springer, 2002b.

Mamaliga, T. Realizing a process cube allowing for the comparison of event data. Master's thesis, TU Eindhoven, 2013.

Mansmann, S., Neumuth, T., and Scholl, M. H. Multidimensional data modeling for business process analysis. In Parent, C., Schewe, K.-D., Storey, V. C., and Thalheim, B., editors, *Conceptual Modeling - ER 2007*, pages 23–38, Berlin, Heidelberg, 2007a. Springer Berlin Heidelberg. ISBN 978-3-540-75563-0.

Mansmann, S., Neumuth, T., and Scholl, M. H. Olap technology for business process intelligence: Challenges and solutions. In Song, I. Y., Eder, J., and Nguyen, T. M., editors, *Data Warehousing and Knowledge Discovery: 9th International Conference, DaWaK 2007, Regensburg Germany, September 3-7, 2007. Proceedings*, pages 111–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007b. ISBN 978-3-540-74553-2. doi: 10.1007/978-3-540-74553-2_11. URL `https://doi.org/10.1007/978-3-540-74553-2_11`.

Marjanovic, O. The next stage of operational business intelligence: Creating new challenges for business process management. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 215c–215c. IEEE, Jan 2007.

Melchert, F., Winter, R., and Klesse, M. Aligning process automation and business intelligence to support corporate performance management. In *Proceedings of the 10th Americas Conference on Information Systems*, AMICS'04, page 4053-63. Springer, 2004.

Mendling, J. and Simon, C. Business process design by view integration. In Eder, J. and Dustdar, S., editors, *Business Process Management Workshops*, pages 55–64, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-38445-8.

Moon, M., Hong, M., and Yeom, K. Two-level variability analysis for business process with reusability and extensibility. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 263–270, July 2008. doi: 10.1109/COMPSAC.2008.129.

Mundy, J., Thornthwaite, W., and Kimball, R. *The Microsoft Data Warehouse Toolkit: With SQL Server 2008 R2 and the Microsoft Business Intelligence Toolset.* Wiley Publishing, 2nd edition, 2011. ISBN 0470640383, 9780470640388.

Neumuth, T., Mansmann, S., Scholl, M. H., and Burgert, O. Data warehousing technology for surgical workflow analysis. In *21st IEEE International Symposium on Computer-Based Medical Systems*, CBMS'08, pages 230–235. IEEE, June 2008. doi: https://doi.org/10.1109/CBMS.2008.41.

Niedrite, L., Solodovnikova, D., Treimanis, M., and Niedritis, A. Goal-driven design of a data warehouse-based business process analysis system. In *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. On Artificial Intelligence, Knowledge Engineering and Data Bases - Volume 6*, AIKED'07, pages 243–249, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-960-8457-59-1. URL `http://dl.acm.org/citation.cfm?id=1348485.1348528`.

Pascalau, E., Awad, A., Sakr, S., and Weske, M. On maintaining consistency of process model variants. In zur Muehlen, M. and Su, J., editors, *Business Process Management Workshops*, pages 289–300, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-20511-8.

Pau, K. C., Si, Y. W., and Dumas, M. Data warehouse model for audit trail analysis in workflows. In *Student Workshop of the 2007 IEEE International Conference on e-Business Engineering (ICEBE 2007)*. IEEE, 2007.

Peter, V. d. S. and Liese, B. Discovery and analysis of the dutch permitting process. 2015. doi: https://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2015:bpic2015_paper_5.pdf.

Polyvyanyy, A., Ouyang, C., Barros, A., and van der Aalst, W. M. Process querying: Enabling business intelligence through query-based process analytics. *Decision Support Systems*, 100:41 – 56, 2017. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2017.04.011. URL `http://www.sciencedirect.com/science/article/pii/S0167923617300787`. Smart Business Process Management.

Puhlmann, F., Schnieders, A., Weiland, J., and Weske, M. Variability mechanisms for process models. *PESOA-Report TR*, 17:10–61, 2005.

Reichert, M., Rinderle, S., Kreher, U., and Dadam, P. Adaptive process management with adept2. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 1113–1114, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2285-8. doi: 10.1109/ICDE.2005.17. URL `https://doi.org/10.1109/ICDE.2005.17`.

Reijers, H., Mans, R., and van der Toorn, R. Improved model management with aggregated business process models. *Data & Knowledge Engineering*, 68(2):221 – 243, 2009. ISSN 0169-023X. doi: https://doi.org/10.1016/j.datak.2008.09.004. URL `http://www.sciencedirect.com/science/article/pii/S0169023X08001328`.

Ribeiro, J. T. S. and Weijters, A. J. M. M. Event cube: Another perspective on business processes. In Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D. C., White, J., Hauswirth, M., Hitzler, P., and Mohania, M., editors, *On the Move to Meaningful Internet Systems: OTM 2011*, pages 274–283, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25109-2.

Rosa, M. L., Dumas, M., ter Hofstede, A. H., and Mendling, J. Configurable multi-perspective business process models. *Information Systems*, 36(2):313 – 340, 2011. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2010.07.001. URL `http://www.sciencedirect.com/science/article/pii/S0306437910000633`. Special Issue: Semantic Integration of Data, Multimedia, and Services.

Rosa, M. L., Aalst, W. M. P. V. D., Dumas, M., and Milani, F. P. Business process variability modeling: A survey. *ACM Computing Surveys (CSUR)*, 50(1):2:1–2:45, March 2017. ISSN 0360-0300. doi: 10.1145/3041957. URL `http://doi.acm.org/10.1145/3041957`.

Rosemann, M. and van der Aalst, W. M. P. A configurable reference modelling language. *Inf. Syst.*, 32(1):1–23, March 2007. ISSN 0306-4379. doi: 10.1016/j.is.2005.05.003. URL `http://dx.doi.org/10.1016/j.is.2005.05.003`.

Rosemann, M. Application reference models and building blocks for management and control. In Bernus, P., Nemes, L., and Schmidt, G., editors, *Handbook on Enterprise Architecture*, pages 595–615. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-24744-9. doi: 10.1007/978-3-540-24744-9_17. URL `https://doi.org/10.1007/978-3-540-24744-9_17`.

Sayal, M., Casati, F., Dayal, U., and Shan, M.-C. Chapter 79 - business process cockpit: Extended abstract. In Bernstein, P. A., , Ioannidis, Y. E., Ramakrishnan, R., and Papadias, D., editors, *{VLDB} '02: Proceedings of the 28th International Conference on Very Large Databases*, pages 880 – 883. Morgan Kaufmann, San Francisco, 2002. ISBN 978-1-55860-869-6. doi: https://doi.org/10.1016/B978-155860869-6/50086-X. URL `https://www.sciencedirect.com/science/article/pii/B978155860869650086X`.

Scheer, A.-W. and Nüttgens, M. Aris architecture and reference models for business process management. In van der Aalst, W., Desel, J., and Oberweis, A., editors, *Business Process Management: Models, Techniques, and Empirical Studies*, pages 376–389. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. ISBN 978-3-540-45594-3. doi: 10.1007/3-540-45594-9_24. URL `https://doi.org/10.1007/3-540-45594-9_24`.

Schiefer, J., List, B., and Bruckner, R. M. Process data store: A real-time data store for monitoring business processes. In *Proceedings of Database and Expert Systems Applications*, DEXA'03, pages 760–770. Springer LNCS, 2003.

Schnieders, A. and Puhlmann, F. Variability mechanisms in e-business process families. In *9th International Conference on Business Information Systems (BIS'06)(LNI)*, volume 85, pages 583–601, 2006.

Shahzad, K. and Johannesson, P. An evaluation of process warehousing approaches for business process analysis. In *Proceedings of the International Workshop on Enterprises & Organizational Modeling and Simulation*, page 10. ACM, 2009.

Shahzad, K. and Zdravkovic, J. Process warehouses in practice: a goal–driven method for business process analysis. *Journal of Software: Evolution and Process*, 24(3):321–339, 2012. doi: 10.1002/smr.555. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.555`.

Shahzad, M. K. *Improving Business Processes using Process-oriented Data Warehouse*. PhD thesis, KTH Royal Institute of Technology, 2012.

Sun, S., Kumar, A., and Yen, J. Merging workflows: A new perspective on connecting business processes. *Decision Support Systems*, 42(2):844 – 858, 2006. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2005.07.001. URL `http://www.sciencedirect.com/science/article/pii/S0167923605000990`.

Torres, V., Zugal, S., Weber, B., Manfred, R., Ayora, C., and Pelechano, V. A qualitative comparison of approaches supporting business process variability. In *3rd Int'l Workshop on Reuse in Business Process Management (rBPM 2012). BPM'12 Workshops*, number 132 in LNBIP, pages 560–572. Springer, September 2012. URL `http://dbis.eprints.uni-ulm.de/844/`.

Valença, G., Alves, C., Alves, V., and Niu, N. A systematic mapping study on business process variability. *International Journal of Computer Science & Information Technology*, 5(1):1, 2013.

van der Aalst, W. M. P., Dreiling, A., Gottschalk, F., Rosemann, M., and Jansen-Vullers, M. H. Configurable process models as a basis for reference modeling. In Bussler, C. J. and Haller, A., editors, *Business Process Management Workshops*, pages 512–518, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-32596-3.

van der Aalst, W., Adriansyah, A., de Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., van den Brand, P., Brandtjen, R., Buijs, J., Burattin, A., Carmona, J., and Castellanos, M. Process mining manifesto. In *Business Process Management Workshops*, pages 169–194, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-28108-2.

van der Aalst, W. M. P. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 3642193447, 9783642193446.

van der Aalst, W. M. P. Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia Pacific Business Process Management*, pages 1–22. Springer Verlag, 2013.

van der Aalst, W. M. P., La Rosa, M., ter Hofstede, A., and Wynn, M. Liquid business process model collections. In Gianni, D., D'Ambrogio, A., and Tolk, A., editors, *Modeling and Simulation-Based Systems Engineering Handbook*, pages 401–424. CRC Press, 2014.

Van Der Aalst, W. M. Business process management: A comprehensive survey. *ISRN Software Engineering*, pages 1–37, 2013. doi: http://dx.doi.org/10.1155/2013/507984.

van der Aalst, W. M. and van Hee Kees. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-01189-1.

van der Aalst, W. M., Dreiling, A., Gottschalk, F., Rosemann, M., and Jansen-Vullers, M. H. Configurable process models as a basis for reference modeling. In *International Conference on Business Process Management*, pages 512–518. Springer, 2005.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros, A. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, Jul 2003. ISSN 1573-7578. doi: 10.1023/A:1022883727209. URL `https://doi.org/10.1023/A:1022883727209`.

van Dongen, B. F., Alves de Medeiros, A. K., and Wen, L. Process mining: Overview and outlook of petri net discovery algorithms. In Jensen, K. and van der Aalst, W. M. P., editors, *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, pages 225–242. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00899-3. doi: 10.1007/978-3-642-00899-3_13. URL `https://doi.org/10.1007/978-3-642-00899-3_13`.

Vogelgesang, T. and Appelrath, H.-J. A relational data warehouse for multidimensional process mining. In *International Symposium on Data-Driven Process Discovery and Analysis*, pages 155–184. Springer, 2015.

Vogelgesang, T., Kaes, G., Rinderle-Ma, S., and Appelrath, H.-J. Multidimensional process mining: questions, requirements, and limitations. In *Proceedings of the CAiSE 2016 Forum*, pages 169—-176, 2016.

Weber, B., Reichert, M., Mendling, J., and Reijers, H. A. Refactoring large process model repositories. *Computers in Industry*, 62(5):467 – 486, 2011. ISSN 0166-3615. doi: https://doi.org/10.1016/j.compind.2010.12.012. URL `http://www.sciencedirect.com/science/article/pii/S0166361510001843`.

Weidlich, M., Mendling, J., and Weske, M. A foundational approach for managing process variability. In Mouratidis, H. and Rolland, C., editors, *Advanced Information Systems Engineering*, pages 267–282, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21640-4.

Weijters, A. J. M. M. and van der Aalst, W. M. P. Rediscovering workflow models from event-based data using little thumb. *Integr. Comput.-Aided Eng.*, 10(2):151–162, April 2003. ISSN 1069-2509. URL `http://dl.acm.org/citation.cfm?id=1273320.1273325`.

Weske, M. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag Berlin Heidelberg, The Netherlands, 2012. ISBN 978-3-642-28616-2. doi: 10.1007/978-3-642-28616-2.

Wyner, G. M. and Lee, J. Defining specialization for process models. In Malone, T. W., Crowston, K., and Herman, G. A., editors, *Organizing Business Knowledge - The MIT Process Handbook.*, pages 131–174. MIT Press, 2003.

Yan, Z., Dijkman, R., and Grefen, P. Fast business process similarity search with feature-based similarity estimation. In Meersman, R., Dillon, T., and Herrero, P., editors, *On the Move to Meaningful Internet Systems: OTM 2010*, pages 60–77, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-16934-2.

Appendices

APPENDIX A

Process meta-model

Figure A.1 consists of the entity-relationship diagram that captures only design-time process elements; it's modeled using Visual Paradigm v.14.1 (database modeling component). Whereas Figure A.2 consists of the entity-relationship diagram that captures all design-time and runtime process elements. From this physical data model we generate our relational database.

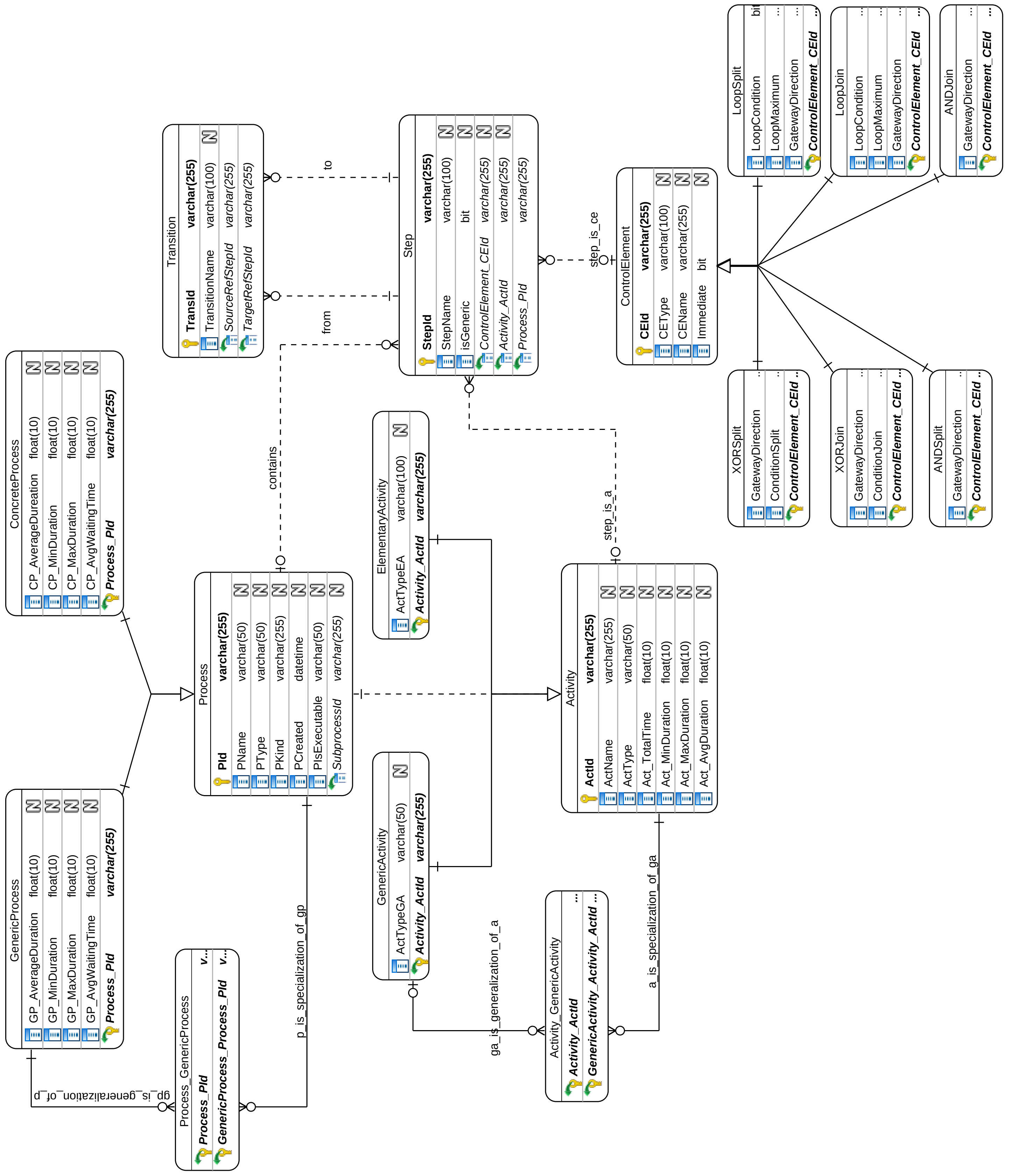


Figure A.1: An excerpt of process ER physical diagram

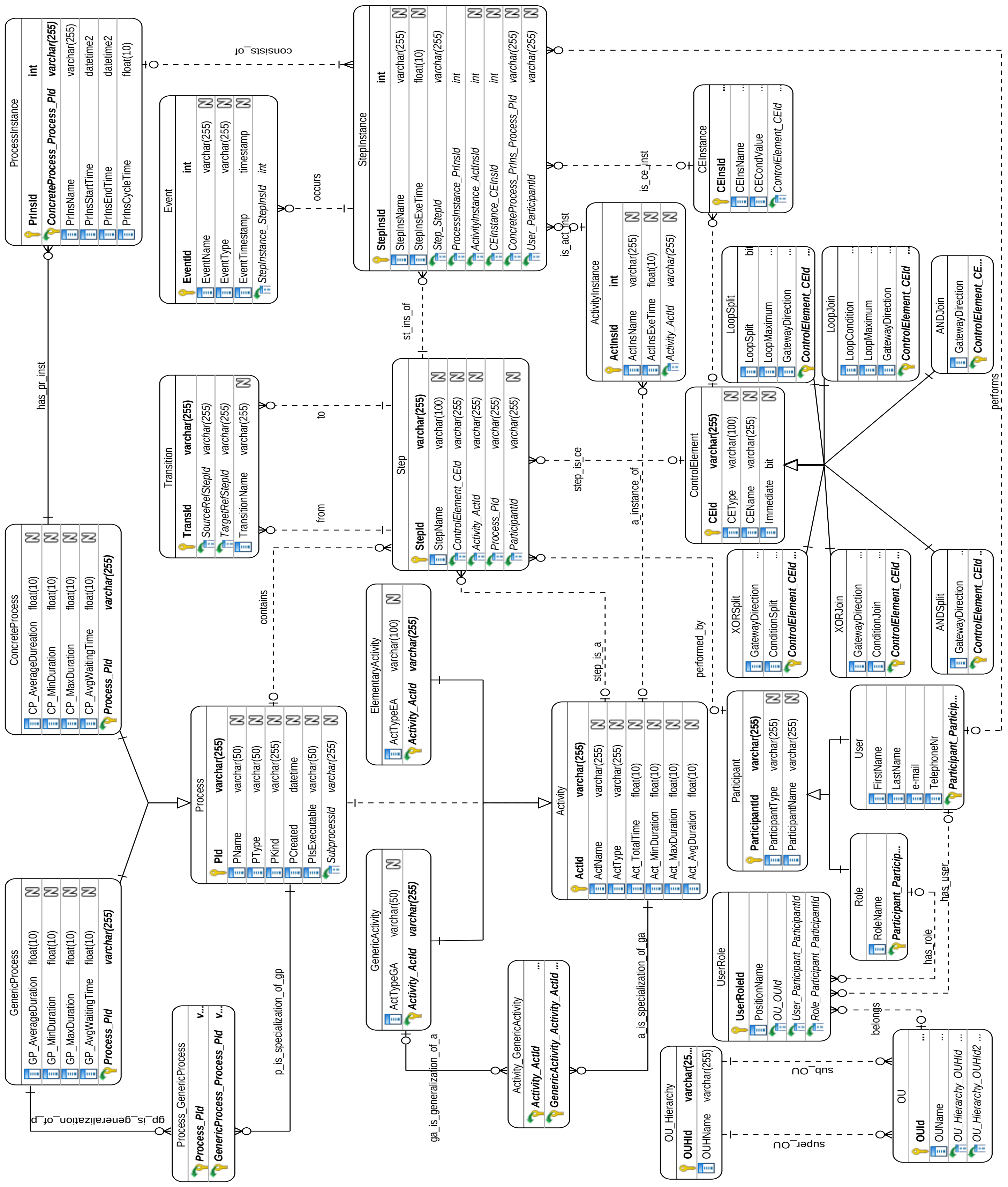


Figure A.2: A full ER physical diagram

APPENDIX B

Building permit application details

In the following we give a list of activity codes and names(both English and Dutch) of building permit application process.

Table B.1: List of activity codes and their En + Dutch labels

taskCode	taskEngName	taskNLname
01_HOOFD_490_1	creating environmental permit decision	aanmaken besluit omgevingsvergunning
01_HOOFD_375	phase advice known	fase advies bekend
09_AH_I_010	article 34 WABO applies	artikel 34 WABO van toepassing
01_HOOFD_380	grounds for refusal	weigeringsgrond
01_HOOFD_200	send letter in progress	brief in behandeling versturen
01_HOOFD_010	register submission date request	registratie datum binnenkomst aanvraag
01_HOOFD_020	send confirmation receipt	versturen ontvangstbevestiging
01_HOOFD_330	procedure change finalize	procedure verandering
01_HOOFD_015	phase application received	fase aanvraag ontvangen
01_HOOFD_030_1	send confirmation receipt finalize	versturen ontvangstbevestiging
01_BB_540	objection lodged against decision	bezwaar tegen beschikking ingediend
01_HOOFD_030_2	enter senddate acknowledgement	invoeren verzenddatum ontvangstbevestiging
01_HOOFD_495	phase decision taken	fase besluit genomen
01_HOOFD_065_1	send procedure confirmation	procedurebevestiging versturen
01_HOOFD_195	phase application receptive	fase aanvraag ontvankelijk
01_HOOFD_490_2	decision date prior to decision	besluitdatum voorafgaand aan beschikking
01_HOOFD_370	assessment of content completed	inhoudelijke beoordeling afgerond
01_BB_770	set phase: phase permitting irrevocable	instellen besluitfase: oorspronkelijk besluit
05_EIND_010	terminate on request	beeindigen op verzoek
01_HOOFD_510_1	transcript decision environmental permit to stakeholders	afschrift beschikking omgevingsvergunning aan belanghebbenden
01_HOOFD_480	by law	van rechtswege
01_HOOFD_430	ask stakeholders views	belanghebbenden zienswijzen vragen
01_HOOFD_510_2	enter senddate decision environmental permit	invoeren verzenddatum beschikking omgevingsvergunning
01_HOOFD_810	register deadline	registratie einddatum
01_HOOFD_065_2	enter senddate procedure confirmation	invoeren verzenddatum procedurebevestiging
01_HOOFD_180	procedure change	procedure verandering
03_GBH_005	applicant is stakeholder	aanvrager is belanghebbende

An example of process variant 1 with names and corresponding task codes.

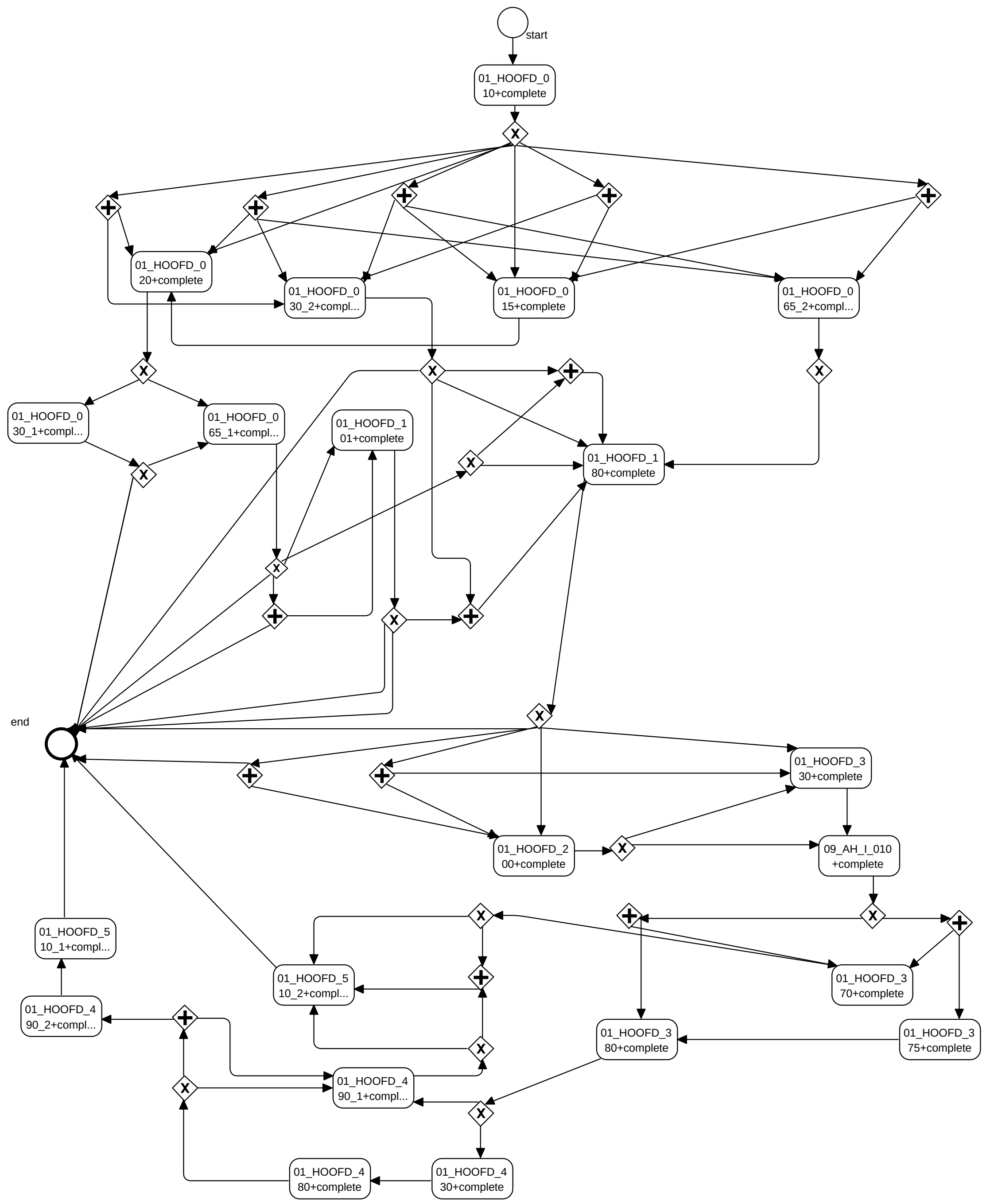


Figure B.1: The full process model of building permit application of Municipality 1
01_HOOFD_030_1 = send confirmation receipt; 01_HOOFD_375 = phase advice known 01_HOOFD_180 = procedure change; 01_HOOFD_020 = send confirmation receipt finalize; 01_HOOFD_380 = grounds for refusal;09_AH_I_010 = article 34 WABO applies; 01_HOOFD_490_1 = creating environmental permit decision 01_HOOFD_200 = send letter in progress; 01_HOOFD_490_2 = decision date prior to decision 01_HOOFD_370 = assessment of content completed; 01_HOOFD_065_1 = send procedure confirmation 01_HOOFD_010 = register submission date request; 01_HOOFD_015 = phase application received 01_HOOFD_510_1 = transcript decision environmental permit to stakeholders; 01_HOOFD_101 registration date publication; 01_HOOFD_330 = procedure change finalize; 01_HOOFD_030_2 = enter senddate acknowledgement 01_HOOFD_510_2 = enter senddate decision environmental permit; 01_HOOFD_430 = ask stakeholders views 01_HOOFD_065_2 = enter senddate procedure confirmation; 01_HOOFD_480 = by law

Whereas in the following pages we show all the five process model variants we imported to our dB. Each of the process model variant is displayed with typical task codes as follow:

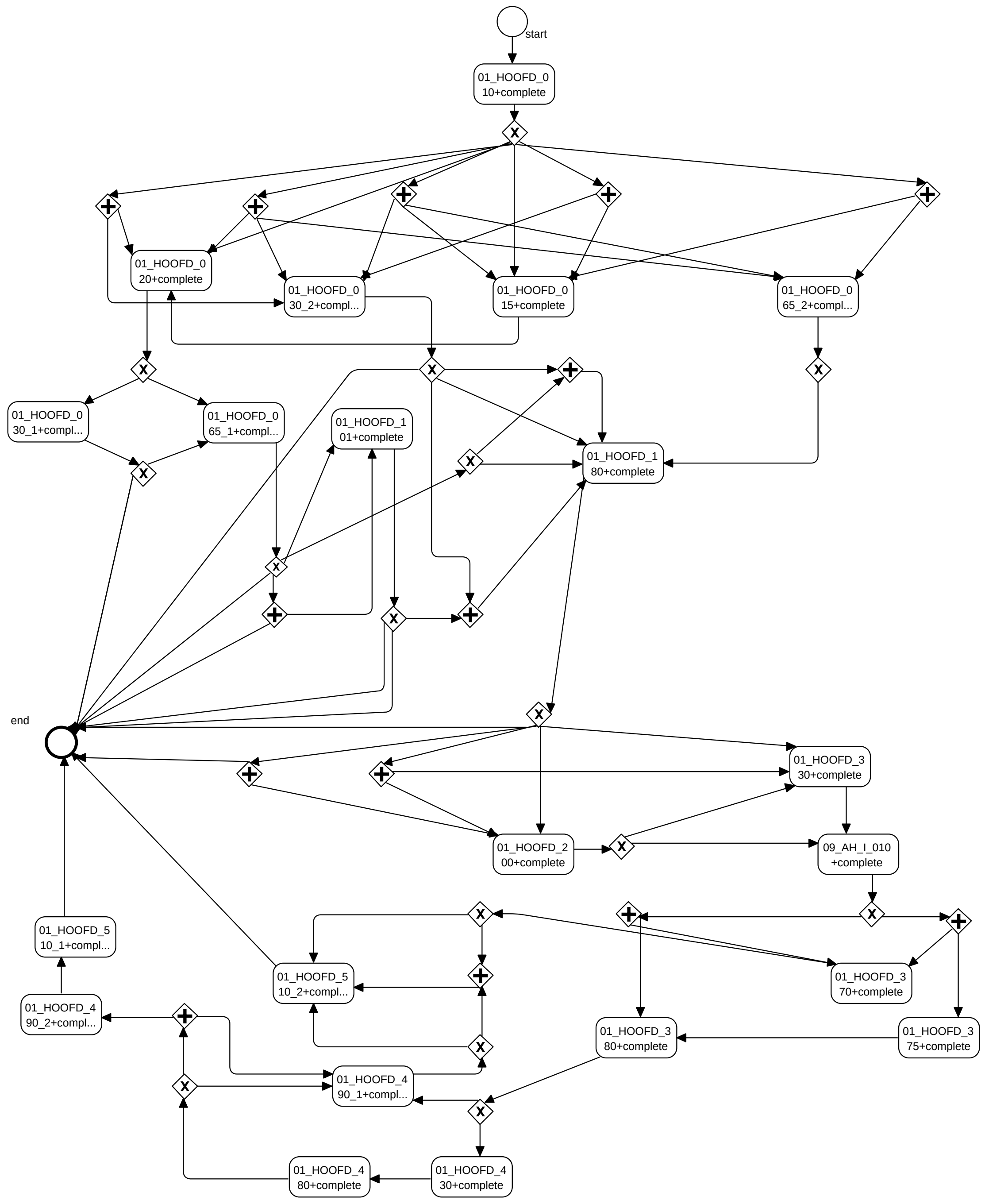


Figure B.2: The full process model of building permit application of Municipality 1

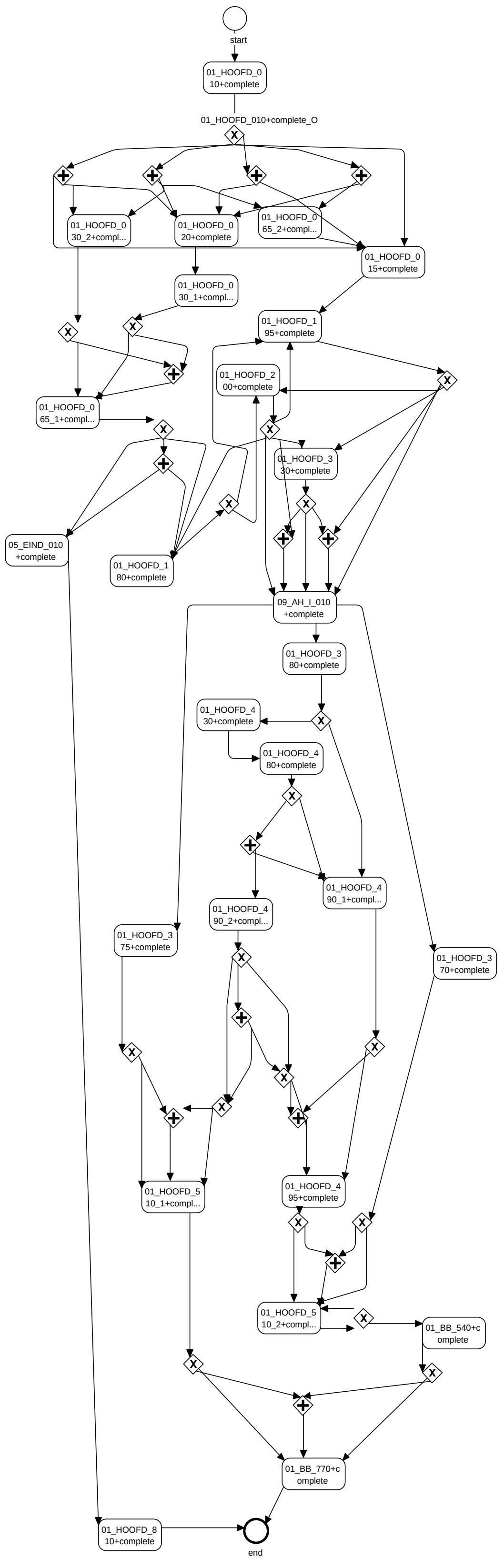


Figure B.3: Process model of building permit application of Municipality 2

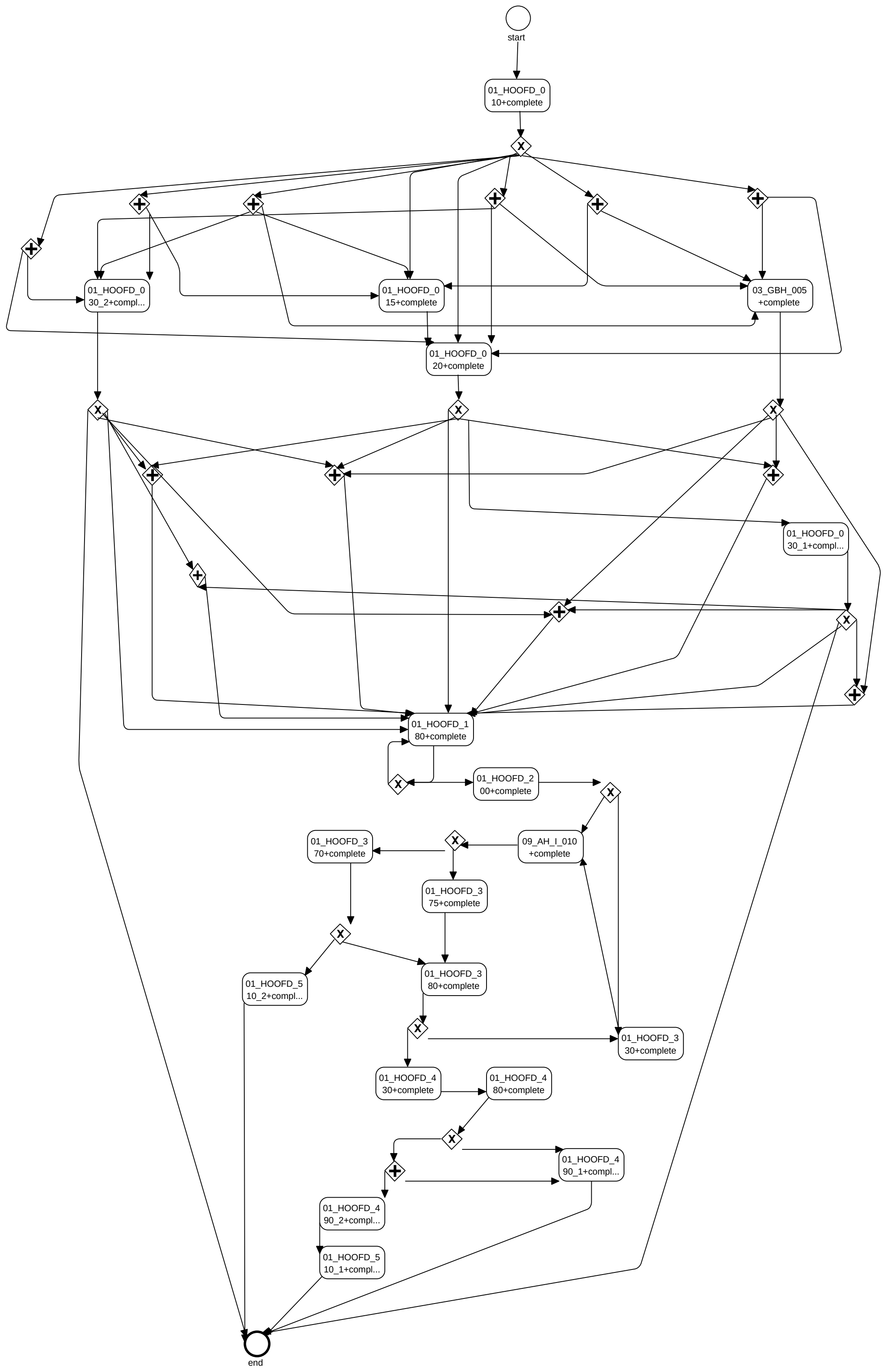


Figure B.4: Process model of building permit application of Municipality 3

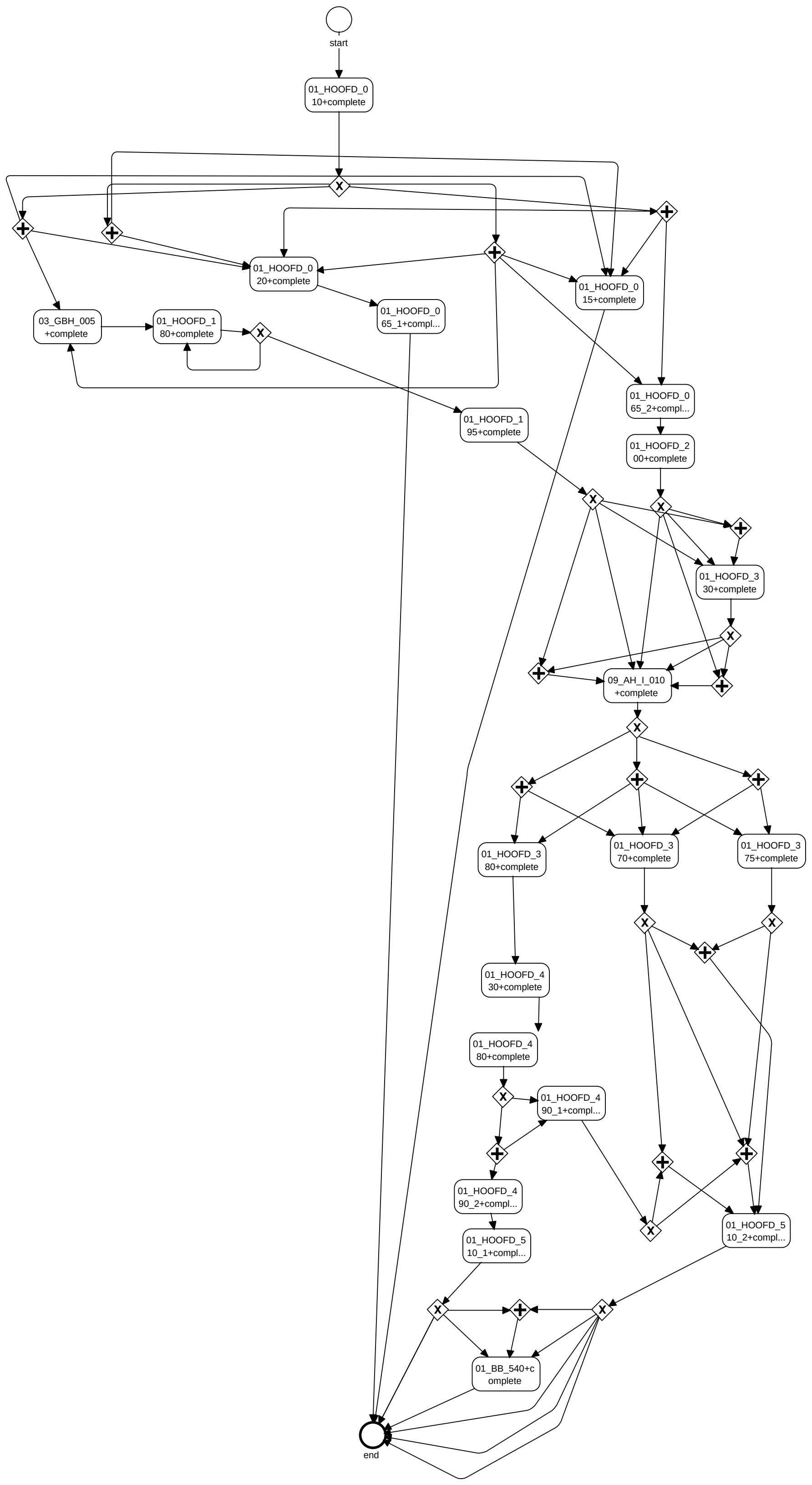


Figure B.5: Process model of building permit application of Municipality 4

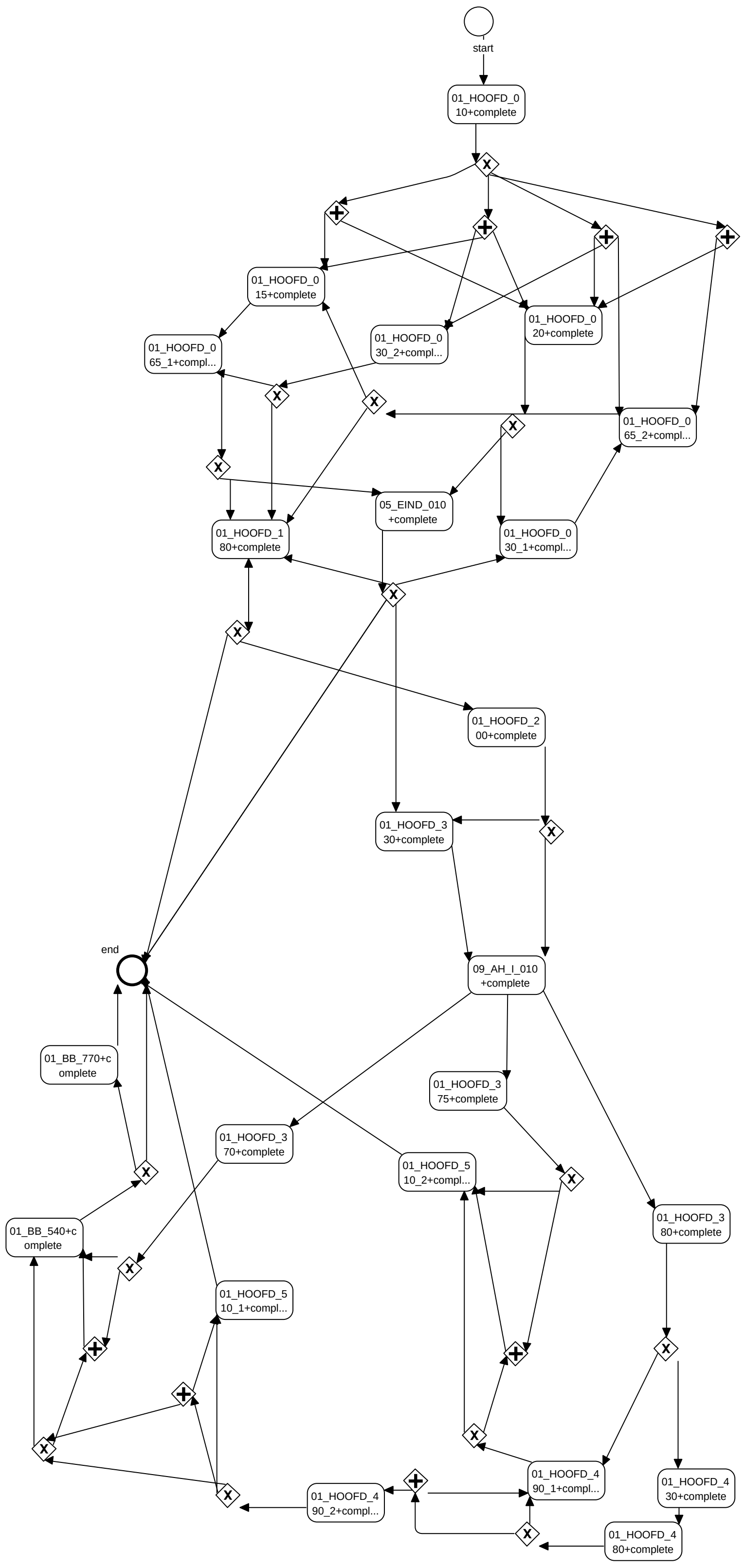


Figure B.6: Process model of building permit application of Municipality 5

Scripts to generate data for Date and Time dimension of the process warehouse
In the following section we write the SQL scripts we used to generate data for our two dimensions.

Script for generating Time *Dim*

```
SET DATEFORMAT ymd;

DECLARE     @startdate   datetime2(7)
DECLARE     @enddate      datetime2(7)
DECLARE     @date         datetime2(7)
DECLARE     @time_id int


SET         @startdate   = (SELECT MIN(EventTimestamp)
                             FROM BuildingPermit_dB2.dbo.Event
WHERE        EventName IN  (SELECT StepInsName
                             FROM BuildingPermit_dB2.dbo.StepInstance ))

SET         @enddate   = (SELECT MAX(EventTimestamp)
                             FROM BuildingPermit_dB2.dbo.Event
WHERE         EventName IN  (SELECT StepInsName
                             FROM BuildingPermit_dB2.dbo.StepInstance ))

SET         @startdate   = CONVERT(datetime2(7), @startdate)
SET         @enddate     = CONVERT(datetime2(7), @enddate )

SET @date=@startdate

SET @time_id = 0;

WHILE     @date <= @enddate
BEGIN
SET @time_id = @time_id + 1
    INSERT INTO dbo.Time(TimeId,FullDateTime, Hour, FullDate)
    VALUES (
     @time_id
    ,    CONVERT(datetime,@date)                          --TheDate
  --   ,  DATEPART(MI, @date)                           --Minute, this is altered
    in this case study and uncomment in the first case study and added to the
     insert statement
    ,   DATEPART(HH, @date)                              --Hour
    ,  CONVERT(date, @date,103)                          --only date format
    )


    SET   @date   =    DATEADD(HH, 1, @date) -- ** add 1 h
--  SET   @date   =    DATEADD(MI, 1, @date) -- ** add 1 m --this statement is
     used instead of the previous one in the first case study

END
```

Script for generating Date *Dim*

```

SET ANSI_NULLS ON
```

```
GO

SET QUOTED_IDENTIFIER ON
GO



CREATE PROCEDURE [dbo].[usp_GenerateDateDimension]
(
   @StartDate date
   ,@NumberOfYears int
)

AS

SET NOCOUNT ON

SET DATEFIRST  1; -- Monday
SET DATEFORMAT ymd;
--SET LANGUAGE   US_ENGLISH;

DECLARE @CutoffDate DATE = DATEADD(YEAR, @NumberOfYears, @StartDate);

-- temp holding table
IF OBJECT_ID ('tempdb..#dim') IS NOT NULL
DROP TABLE #dim;

CREATE TABLE #dim
(
   [DateId]        date PRIMARY KEY,
   [Day]          AS DATEPART(DAY,       [DateId]),
   [Month]        AS DATEPART(MONTH,     [DateId]),
   [MonthName]    AS DATENAME(MONTH,     [DateId]),
   [Week]         AS DATEPART(WEEK,      [DateId]),
   [ISOweek]      AS DATEPART(ISO_WEEK,  [DateId]),
   [DayOfWeek]    AS DATEPART(WEEKDAY,   [DateId]),
   [Quarter]      AS DATEPART(QUARTER,   [DateId]),
   [Year]         AS DATEPART(YEAR,      [DateId]),
);


BEGIN TRY
   BEGIN TRANSACTION
      INSERT #dim([DateId])
      SELECT d
      FROM
      (
         SELECT d = DATEADD(DAY, rn - 1, @StartDate)
         FROM
         (
            SELECT
               -- Get TOP X rows, the number of dates to generate
               TOP (DATEDIFF(DAY, @StartDate, @CutoffDate))
               rn = ROW_NUMBER() OVER (ORDER BY s1.[object_id])
            FROM sys.all_objects AS s1
            CROSS JOIN sys.all_objects AS s2
```

```
          ORDER BY s1.[object_id]
        ) AS x
      ) AS y;

    IF OBJECT_ID(N'dbo.[Date]', N'U') IS NOT NULL
    DROP TABLE dbo.[Date]
  CREATE TABLE dbo.[Date]
  (
    [DateId]            date         NOT NULL PRIMARY KEY,
    [Day]               tinyint      NOT NULL,
    [Weekday]           tinyint      NOT NULL,
    [WeekdayName]       nvarchar(10) NOT NULL,
    [IsWeekend]         bit          NOT NULL,
    [DayOfYear]         smallint     NOT NULL,
    [WeekOfMonth]       tinyint      NOT NULL,
    [WeekOfYear]        tinyint      NOT NULL,
    [ISOWeekOfYear]     tinyint      NOT NULL,
    [Month]             tinyint      NOT NULL,
    [MonthName]         nvarchar(10) NOT NULL,
    [Quarter]           tinyint      NOT NULL,
    [QuarterName]       nvarchar(6)  NOT NULL,
    [Year]              int          NOT NULL,
    [YYYY-QQ]           nvarchar(7)  NOT NULL,
    [YYYY-MM]           nvarchar(7)  NOT NULL,
  );

  INSERT INTO dbo.[Date]
  SELECT
     [DateId]           = [DateId]
    ,[Day]              = CONVERT(tinyint, [Day])
    ,[Weekday]          = CONVERT(tinyint, [DayOfWeek])
    ,[WeekDayName]      = CONVERT(nvarchar(10), DATENAME(WEEKDAY, [DateId]))
    ,[IsWeekend]        = CONVERT(bit,
                          CASE WHEN [DayOfWeek] IN (6, 7) THEN 1 ELSE 0 END)
    ,[DayOfYear]        = CONVERT(smallint, DATEPART(DAYOFYEAR, [DateId]))
    ,[WeekOfMonth]      = CONVERT(tinyint,
                          DENSE_RANK() OVER (
                            PARTITION BY [year], [month] ORDER BY [week] ))
    ,[WeekOfYear]       = CONVERT(tinyint, [week])
    ,[ISOWeekOfYear]    = CONVERT(tinyint, ISOWeek)
    ,[Month]            = CONVERT(tinyint, [month])
    ,[MonthName]        = CONVERT(nvarchar(10), [MonthName])
    ,[Quarter]          = CONVERT(tinyint, [quarter])
    ,[QuarterName]      = CONVERT(nvarchar(6),
                          CASE [quarter]
                            WHEN 1 THEN N'First'
                            WHEN 2 THEN N'Second'
                            WHEN 3 THEN N'Third'
                            WHEN 4 THEN N'Fourth'
                          END)
    ,[Year]             = [year]
    ,[YYYY-QQ]          = CAST([year] AS nvarchar(4))
                          + '-' + RIGHT( '00' +
                          CAST( CONVERT(tinyint, [quarter]) AS nvarchar(1) )
                          , (2) )
    ,[YYYY-MM]          = CAST([year] AS nvarchar(4))
```

```
                             + '-' + RIGHT( '00' +
                             CAST( CONVERT(tinyint , [month]) AS nvarchar(2) )
                             , (2) )
    FROM #dim

  COMMIT TRANSACTION -- End of transaction

END TRY -- End of try

BEGIN CATCH
  IF @@TRANCOUNT > 0
    ROLLBACK TRANSACTION

END CATCH
GO
```

Some process variant statistics for a specific year

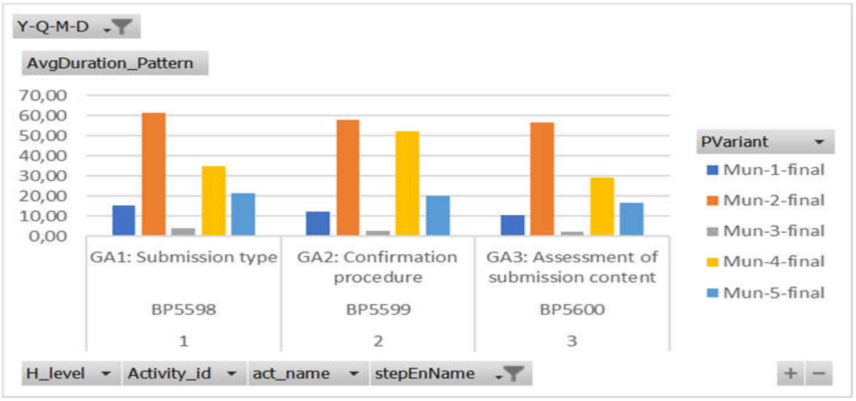
We show average duration pattern results in Figure B.7 across multiple variants for two specific years as well, e. g., 2011 and 2012.

AvgDuration_Pattern	Column Labels				
Row Labels	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**

Average Pattern Duration KPIs across different municipalities for Y: 2011

Y-Q-M-D	2011

AvgDuration_Pattern	Column Labels				
Row Labels	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**
⊟**1**					
⊟**BP5598**					
⊞GA1: Submission type	🟢15,40	🔴61,34	🟢4,07	🟡34,88	🟢21,49
⊟**2**					
⊟**BP5599**					
⊞GA2: Confirmation procedure	🟢12,40	🔴58,11	🟢2,87	🔴52,17	🟢20,28
⊟**3**					
⊟**BP5600**					
⊞GA3: Assessment of submission content	🟢10,67	🔴56,52	🟢2,37	🟡29,10	🟢16,56



Y-Q-M-D	2012

AvgDuration_Pattern	Column Labels				
Row Labels	**Mun-1-final**	**Mun-2-final**	**Mun-3-final**	**Mun-4-final**	**Mun-5-final**
⊟**1**					
⊟**BP5598**					
⊞GA1: Submission type	🟡7,46	🔴12,46	🟢4,42	🟡8,54	🟢4,64
⊟**2**					
⊟**BP5599**					
⊞GA2: Confirmation procedure	🟢4,55	🟡7,86	🟢2,49	🔴14,02	🟢4,07
⊟**3**					
⊟**BP5600**					
⊞GA3: Assessment of submission content	🟢1,94	🟢4,74	🟢1,73	🟢5,35	🟢1,40

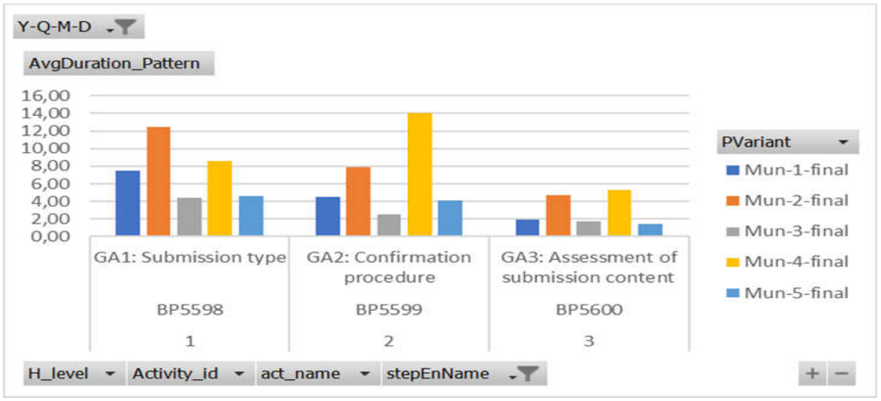


Figure B.7: Average duration pattern across five variant for two specific years

APPENDIX C

Some operations

Some operations on our relational database

Derive process variants hierarchy from generating all steps of concrete process variants after each substitution of a GA

```

IF EXISTS (SELECT * FROM INFORMATION_SCHEMA.tables
            WHERE TABLE_NAME = 'PV_Hierarchy_Level'
            AND   TABLE_SCHEMA='dbo')
DROP TABLE PV_Hierarchy_Level;

CREATE TABLE [dbo].PV_Hierarchy_Level(
    [step_id] [varchar](255) NULL,
    [activity_id] [varchar](255) NULL,
    [pr_id] [varchar](255) NULL,
    [lvl_order] [int] NULL,
    [type] [varchar](50) NULL
) ON [PRIMARY]


GO

DECLARE @step_ID VARCHAR(255)
        , @isGeneric bit
        , @process_ID VARCHAR(255)
        , @activity_ID VARCHAR(255)
        , @lvl INT;


DECLARE steps_cursor CURSOR FOR
WITH REC AS (
    --take all steps which do not have a step before them (aka FIRST steps)
    SELECT *, 1 as lvl from  [dbo].[udf_getFirstStep_all]()

    UNION ALL

    --take all next steps in order
    SELECT s.*, lvl + 1 --increase the level by one
      FROM step s
      JOIN Transition t on s.StepId = t.TargetRefStepId
      JOIN REC r ON r.StepId = t.SourceRefStepId
)
-- if a step is twice, we are interested only with the instance with the
    last lvl
SELECT Process_PId, StepId, Activity_ActId, isGeneric, MAX(lvl) lvl
  FROM REC
 GROUP BY Process_PId, StepId, Activity_ActId, isGeneric
 ORDER BY Process_PId, lvl


OPEN steps_cursor

FETCH NEXT FROM steps_cursor
INTO @process_ID, @step_ID, @activity_ID, @isGeneric, @lvl

WHILE @@FETCH_STATUS = 0
BEGIN

    --skip non activity steps like control element steps
    IF @activity_ID IS NOT NULL
    BEGIN

        IF @isGeneric = 1 --get only Generic activity steps
        BEGIN

            INSERT INTO PV_Hierarchy_Level(step_id, activity_id, pr_id,
                                           [lvl_order])
            SELECT ba.actid, @step_ID,@process_ID,@lvl
             FROM Activity_GenericActivity aga
             CROSS APPLY dbo.udf_boundActivity(aga.Activity_ActId) ba
             WHERE GenericActivity_Activity_ActId = @step_ID


            INSERT INTO PV_Hierarchy_Level(step_id, activity_id, pr_id,
                                          [lvl_order])
            SELECT bfs.Id, @step_ID,@process_ID,@lvl
             FROM Activity_GenericActivity aga
             CROSS APPLY dbo.udf_boundActivity(aga.Activity_ActId) ba
             CROSS APPLY dbo.udf_Breadth_First_Search(ba.actid, null) bfs
             WHERE GenericActivity_Activity_ActId = @step_ID



            INSERT INTO Process_GenericProcess(Process_PId,
    GenericProcess_Process_PId)
            SELECT DISTINCT ba.Process_PId, @process_ID
             FROM Activity_GenericActivity aga
             CROSS APPLY dbo.udf_boundActivity(aga.Activity_ActId) ba
             WHERE GenericActivity_Activity_ActId = @step_ID
             AND NOT EXISTS (SELECT * FROM Process_GenericProcess pgp
                             WHERE pgp.Process_PId = ba.Process_PId
                             AND pgp.GenericProcess_Process_PId =
    @process_ID)


        END
        ELSE
    END

    FETCH NEXT FROM steps_cursor
    INTO @process_ID, @step_ID, @activity_ID, @isGeneric, @lvl
END
CLOSE steps_cursor;
DEALLOCATE steps_cursor;
GO
```

A table-valued user-defined function that gives first steps of all processes

```
CREATE FUNCTION [dbo].[udf_getFirstStep_all] ()
returns table
as return

--get first steps of all processes, except specialized activities

SELECT * FROM step s
     WHERE NOT EXISTS (
             SELECT *
               FROM Transition t
              WHERE s.StepId = t.TargetRefStepId
    )
```

TABLE RESULTS

Table C.1: Results from ordering steps of Process Variant 1

StepId	StepName	StepRenamed	Order	ProcessName
BP4745_BP2335_BP2332	start order	start order-V.1	1	Receive Invoice
BP4745_BP2335_BP2407	Place order	A1	2	Receive Invoice
BP4745_BP2349_BP2407	Receive order	B1	3	Receive Invoice
BP4745_BP2349_BP2409	Request payment by credit-card	E1	4	Receive Invoice
BP4745_BP2335_BP2279	Receive e-Invoice	G1	5	Receive Invoice
BP4745_BP2335_BP2281	Perform Tasks	Perform Tasks	6	Receive Invoice
BP4745_BP2335_BP2283	Manage account	I1	7	Receive Invoice
BP4745_BP2335_BP2335	Update profile	J1	7	Receive Invoice
BP4745_BP2335_BP2292	Tasks performed	Tasks performed	8	Receive Invoice
BP4745_BP2335_BP2294	Ready to pay?	Ready to pay?	9	Receive Invoice
BP4745_BP2335_BP2416	Manage payment	M1	10	Receive Invoice
BP4745_BP2335_BP4801	Make billing inquiry	N1	10	Receive Invoice
BP4746_BP2335_BP2976	start pay	start pay-V.1	11	Pay Invoice
BP4746_BP2335_BP2279	Identify or verify credit-card info	R1	12	Pay Invoice
BP4746_BP2335_BP1539	Enough credit?	Enough credit?	13	Pay Invoice
BP4746_BP2335_BP1541	Notify client	T1	14	Pay Invoice
BP4746_BP2335_BP1536	Charge credit	U1	14	Pay Invoice
BP4746_BP2335_BP2954	Credit charged or declined	Credit charged or declined	15	Pay Invoice
BP4746_BP2335_BP2953	Update customer balance	V1	16	Pay Invoice
BP4746_BP2883_BP2448	Verify successful payment	X1	17	Pay Invoice
BP4746_BP2883_BP2954	Successful payment?	Successful payment?	18	Pay Invoice
BP4746_BP2883_BP2970	Cancel invoice	cancel invoice-V.2	19	Pay Invoice
BP4746_BP2883_BP2406	Invoice paid	invoice paid-v.2	19	Pay Invoice

A table-valued user-defined function that gets all outgoing steps from a specific step using a Breadth first search routine

```sql

CREATE FUNCTION [dbo].[udf_Breadth_First_Search](@StartStep varchar(255),
                                                  @EndStep varchar(255) = NULL )
RETURNS @rtnTable TABLE (
    Id varchar(255)
) AS
BEGIN
    DECLARE @Discovered TABLE (
        Id varchar(255) NOT NULL, -- The Step Id
        Predecessor varchar(255) NULL,    -- The step we came from to get to this step.
        OrderDiscovered int -- The order in which the steps were discovered.
    )

    -- Initially, only the start step is discovered.
    INSERT INTO @Discovered (Id, Predecessor, OrderDiscovered)
    VALUES (@StartStep, NULL, 0)


    DECLARE @lvl INT = -1
    -- Add all steps that we can get to from the current set of steps,
    -- that are not already discovered. Run until no more steps are discovered.
    WHILE @@ROWCOUNT > 0
    BEGIN
        SET @lvl += 1;

         INSERT INTO @Discovered (Id, Predecessor, OrderDiscovered)
         SELECT e.TargetRefStepId, e.SourceRefStepId, d.OrderDiscovered + 1
           FROM @Discovered d
           JOIN dbo.Transition e ON d.Id = e.SourceRefStepId
          WHERE e.TargetRefStepId NOT IN (SELECT Id From @Discovered)
            AND d.OrderDiscovered = @lvl
            AND NOT EXISTS (SELECT 1 FROM @Discovered WHERE Id = @EndStep);
    END;

    INSERT INTO @rtnTable(id)
    SELECT DISTINCT Id
      FROM @Discovered
     WHERE Id NOT IN
            (SELECT CEId
                FROM ControlElement) --exclude all control element steps like xorjoin/split etc.

    RETURN
END

GO
```

A scalar user-defined function that outputs first step of each process

```

--Create a udf that gets the first step for each process
CREATE FUNCTION [dbo].[udf_getFirstStep_of_Process] (@prid varchar(255))
RETURNS varchar(255)
AS
BEGIN
DECLARE @res varchar(255)

--get the first step for a specific process

SELECT @res=StepId  FROM step s
     WHERE NOT EXISTS (
             SELECT *
               FROM Transition t
              WHERE s.StepId = t.TargetRefStepId)
              AND Process_PId=@prid

    RETURN @res
END


GO
```