A Process Warehouse for Process Variants Analysis

Lisana Berberi[1][0000−0002−7632−2466]

Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Karlsruhe, Germany
lisana.berberi@kit.edu
https://www.scc.kit.edu

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 related to data warehouse solutions 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. This paper addresses the above shortcoming by proposing a framework to analyse process variants.
The framework consists of two original contributions: (i) a novel metamodel 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 online analytical processing operations on different variation parts thus providing support to decision-making through KPIs; The framework concepts were defined and validated using a real-life case study. Moreover, a prototype is implemented to support the validation of the framework and performance dashboards are provided with detailed statistics at different levels of abstraction.

Keywords: Process variant · process warehouse · business process analysis.

1 Introduction

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. 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 [17]. For more than a decade process-oriented data warehouse have been introduced as a solution on analysing effectively process activities of an organization. Process Warehouses [2, 9, 16] 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. A way to manage these variants is expressing all the variants 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.

To address these shortcomings we propose 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 metamodel 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.

In this context, the main research question this paper addresses 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. 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, a process performance analysis is crucial. Moreover it helps in estimating process improvement efforts.

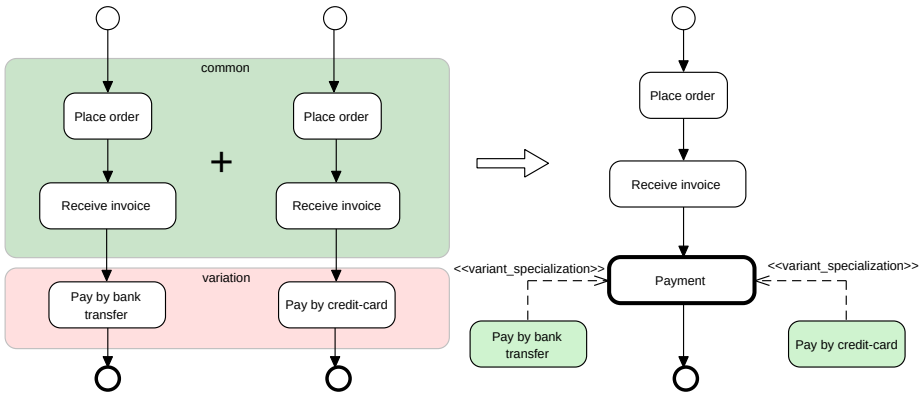


Fig. 1: An example of reference(global) process model abstracting multiple process variants

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 shows two variants of the order-to-pay process represented Business Process Modelling Notation (BPMN)[6]. 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 $\ll variant\_specialization \gg$ assigned to the generic connector between the generic activity and the specialized activities. The remaining of this paper is as follows: chapter 2 presents the meta-model for capturing process variants, chapter 3 gives the Generic PW model to analyse these variants, chapter 4 gives some evalua-

tion results through dashboards, chapter 5 reviews the literature and chapter 6 finally draws some conclusions.

2 A meta-model to capture process variants

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. This meta-model is an extension and alteration of this work [4] we published years ago. The meta-model presented in that paper limited to capture only concepts/elements concerning to a process model and not elements related to the process instances after the enactment of a process. We depict the full meta-model using UML class diagram as shown in Figure 2 that captures process model elements and their variants from a design and run-time perspective. A detail description of all its elements is as follows. [9] 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 [6] 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*. 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. A 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. 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*. 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 *GA2: Confirmation procedure* as depicted with a thick border and a yellow color in Figure 4 might be realized by one of the specialized elementary activities e. g., *F: send confirmation receipt finalize* or *G: send procedure confirmation*. In addition, *GA1: Submission type* might be realized by one of the specialized *customized subprocess* e. g., *SP1: A,B,(C,D,E)* or *SP2: B,C,(D,E)* etc. depending on the activities performed in the different variants. In the meta-model a generic activity and its specializations are related by *a_is_specialization_of_ga* relationship as shown in the figure. We annotate these specialization relationships by using a stereotype ≪*variant_specialization*≫.
A *generic process* (GP) is defined as a process that contains at least one generic activity, e. g., *GP_Permit* contains at least one generic activity, e. g., *GA3: Assessment of submission content* as shown in Figure 4. CP and GP are modelled 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 concrete process *PermitApp* is a specialization of the generic process *GP_Permit*. Other typical elements in the meta-model capture information from organizational and run-time perspectives.

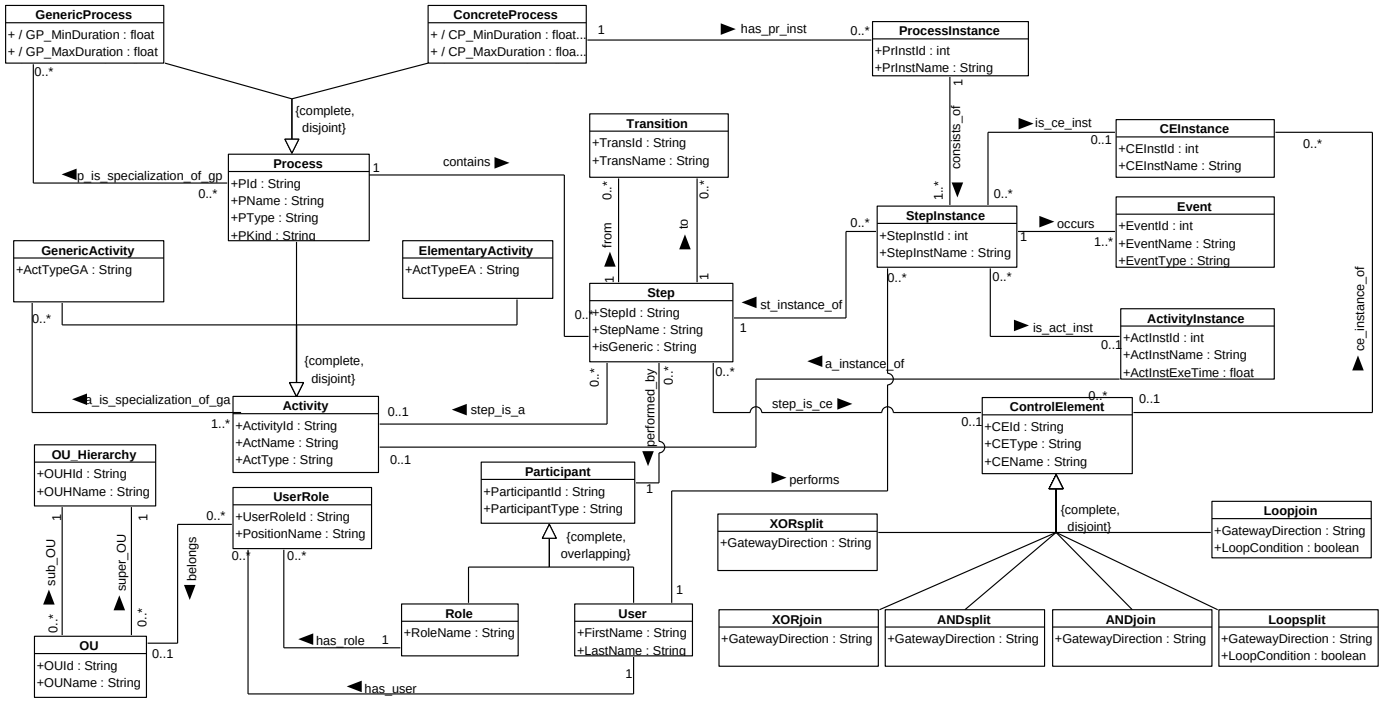


Fig. 2: A full process meta-model capturing process variants and their instances

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.

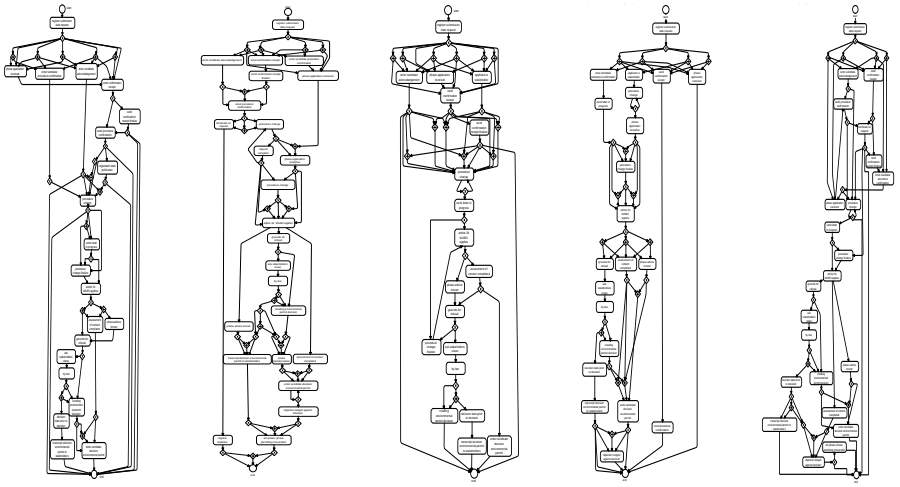
Additional elements captured from organizational perspective (i. e., resource perspective) are: Participant, User, Role, OU, OU_hierarchy. *Participant* consists of a resource that performs many *steps*, i. e., *activities*. Participant might be *User* (individual/registered users of the system) or *Role* (logical description of a position in an OU). Role-based assignments and direct user assignments are possible to express. Other important elements from run-time perspective store information about instances of different process variants such as *ProcessInstance*, *StepInstance*, *ActivityInstance*, *CEInstance*, as instance representations of *processes*, *steps*, *activities*, and their *control elements* during runtime. 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 it belongs.

2.1 Generic process model of building permit application

As a scenario we consider real-life event logs from five Dutch municipalities of 4TU-Center for Research Data repository. [1]
This data contains all building permit applications over a period of approximately four years, from 2010 to 2013 [8]. To obtain readable process models we filtered the event logs using heuristic miner discovery algorithms using *PROM* tool [2] as depicted in Figure 3. Due to page limitations we cannot show explicitly the discovered process models but they are available to view and download in our repository solution. 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* and *extensive procedure*. This process starts with activity *register submission date request* from a stakeholder. Based on the submitted documents the application may be cancelled or followed by activity that publish the registered date of the permit application. If there any incorrect or missing information in it then a change in procedure is needed and followed by two other parallel activities to notify stakeholders for their submitted application and final changes are stored.

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

Afterwards, *article 34 WABO applies* activity in all the five variants follows, where WABO (Wet Algemene Bepalingen Omgevingsrecht) procedure allows that multiple permits related to one project to be 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. 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, if the assessment of the content results complete and positive then a decision environmental permit date is registered and a respective transcript is sent to stakeholders. Otherwise, the application is refused an an extra advisory board should be asked to get their opinions.



(a) Municipality 1 (b) Municipality 2 (c) Municipality 3 (d) Municipality 4 (e) Municipality 5

Fig. 3: Process models mined from 5 event logs

In this section, we build the generic process called "GP_Permit" with three generic activities that marks the variation points among the variants. The approach identifies variation based on structural differences, i. e. only the control flow dependencies are captured. They are designed with a thick border and filled with a yellow color as shown in Figure 4. For each of them the respective specialized activities are designed as well. Instead, the similar activities that are shared among the five process variants are highlighted with a light gray color. In this subsection we illustrate how we model the reference process model as a generic process to capture all the variants by defining three GAs.
In this reference process model we assign each activity a letter preceding the

activity label used as abbreviation. 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 (GAs): *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)* in order to capture all different control flows by using AND, XOR split/joins gateways among the activities.

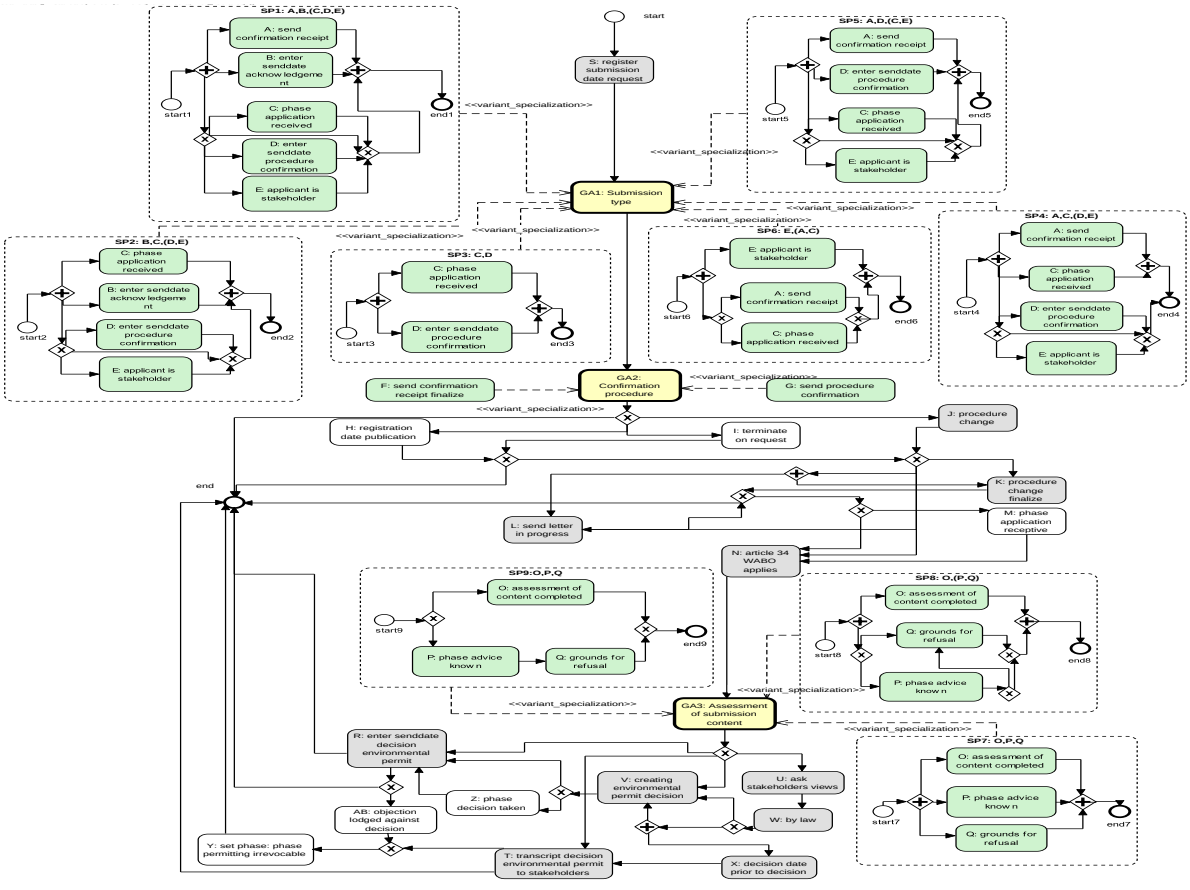


Fig. 4: Generic process model of building permit process

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 briefly describe how the specialization between activities and processes is defined. Specifically, how to apply the substitution/transformation sets and further represent them in a consolidation hierarchy. Accordingly, we derive a process variants hierarchy consolidating step activities of concrete process in different GAs positions.

2.2 Substitution operations and transformation sets

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: -Firstly, filter (procedure FILTER_STEPS() is left out due to space limitation) only some specific occurrences after applying the sequential order of steps from concrete and generic processes; -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 1 . For each direct specialization of generic activities we obtain respective activities from concrete processes as bounded activities. 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. -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 levels of the hierarchy.

Algorithm 1 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[3] each row iteratively (line *3* of this algorithm) until 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 neighbour 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* (e.g. *SP1: A,B,(C,D,E)* from Figure 4) or an *elementary activity* (e.g. *F: send confirmation receipt finalize* from Figure 4).

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

Algorithm 1 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: $bounded\_step\_a \leftarrow \pi_{S\_Bound.act\_id,}$**step_ID,process_ID**$_{,lvl}(\rho_{Sub\_S}(substituted\_step) \times \rho_{S\_Bound}(\text{GETBOUNDEDSTEP\_OF\_A}(Sub\_S.ActivityId)))$

▷ 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: $indirect\_step\_a \leftarrow \pi_{BFS.act\_id,}$**step_ID,process_ID**$_{,lvl}(\rho_{S\_Bound}(bounded\_step\_A) \times \rho_{BFS}(\text{BREADTH\_FIRST\_SEARCH}(S\_Bound.act\_id)))$

▷ 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**

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). 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). Detailed information the interested reader may find in [3] thesis.

3 A generic process warehouse for variants analysis

We based our process warehouse model on information supply derived from the process meta-model presented in chapter 2. This multi-dimensional model is designed as a fact constellation schema with two fact tables, i. e. *ActivityFact* and *ProcessFact* sharing respective dimension tables. The main reason is the possibility for business users to perform analysis based either on different process instances or on different activity instances. It consists of the following six dimensions: *Process*, *Activity*, *Process_Variant*, *GeoLocation*, *Participant*, *Time*, *Date*.

Some ad-hoc queries that are answered based on this model are for example: Display the average process duration of permit applications from a specific pattern: *select avg_duration from* $P_x|P_y|P_z$, $*$ " where $P_x = X, *, ,$ where $X = A|B|C|D|E$, i. e.-comprise all the activities starting from activity A or B or C or D o E in any order and followed by any other activities; whereas $P_y = F|G, *, ,$ i. e.-comprise all the activities starting from activity F or G in any order and followed by any other activities; and $P_z = Z, *, ,$ where $Z = O|P|Q|R$, i. e.-comprise all the activities starting from activity F or G in any order and followed by any other activities. Patterns P_x, P_y, P_z correspond with the concrete activities of the respective generic activities: *GA1: Submission type*, *GA2: Confirmation procedure*,*GA3: Assessment of submission content*. Process warehouses typically feature the dimensions process/activity, organization (department hierarchy), actor/participant, geolocation and time. An extra dimension is used to consolidate occurrences to different levels of genericity named *Process_Variant*. 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 concrete activities.

This approach allows to drill-down to the instances starting either from generic activity to activity or from generic activity to concrete process. We give an excerpt of instances of the process warehouse schema due to page limitation instead of showing the PW schema itself. The *ActivityFact* and *ProcessFact* tables contain all activity instances and process instances respectively after being parsed and imported from the event logs.

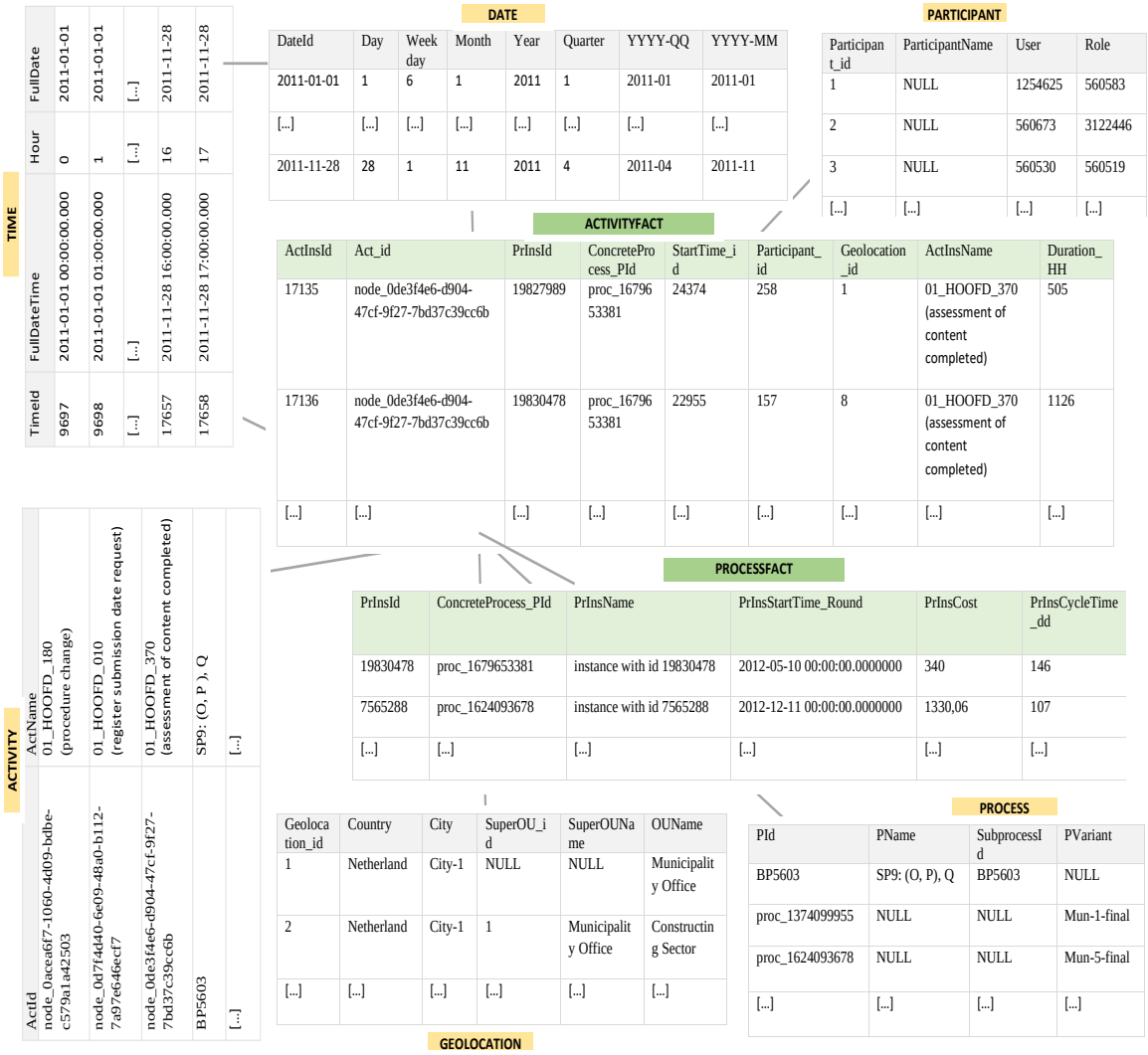


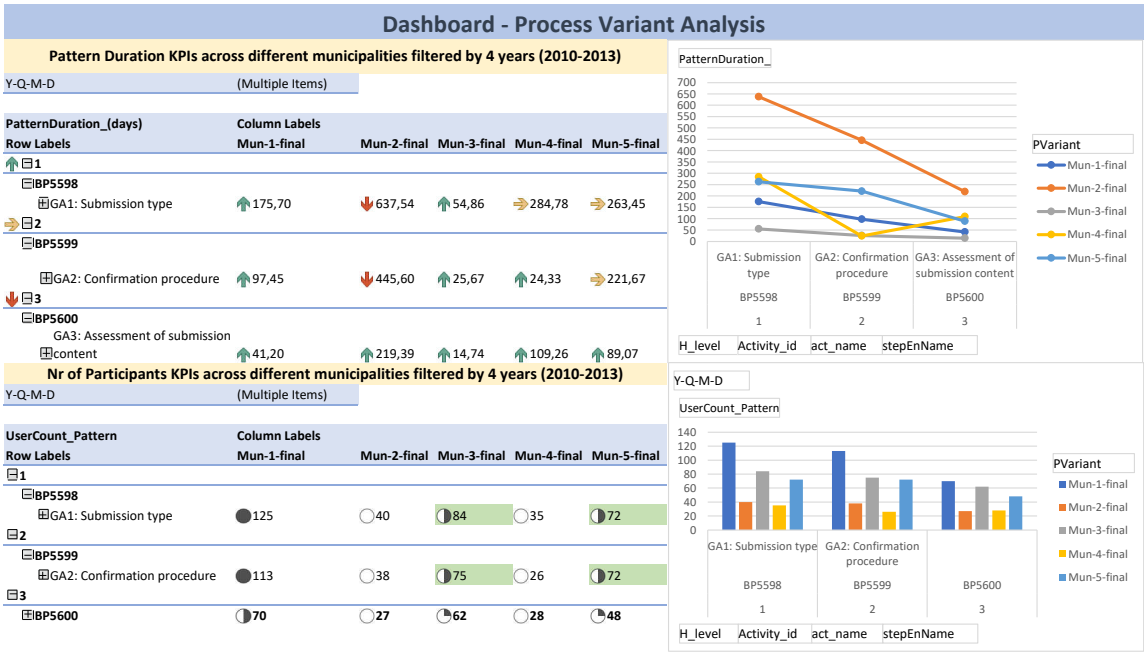
Fig. 5: Instances of the process warehouse for the building permit application

4 Evaluation results from real-life case study

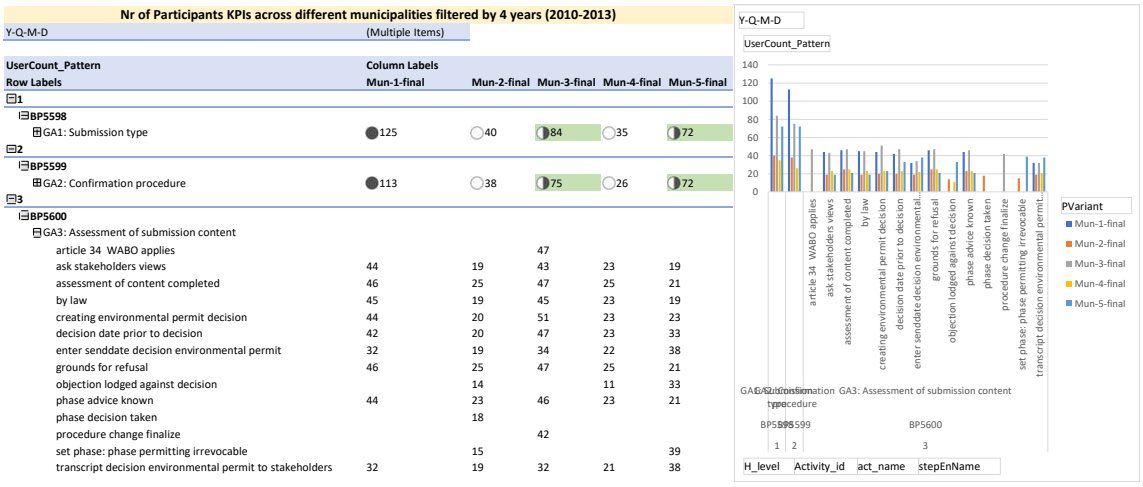
By using this solution different type of queries that are interesting for business users can be answered. First, we address the achievement of core objective we mention at the introduction chapter and then verify that our solution can answer as well other queries provided by conventional BPM systems. We give a

comprehensive analysis of our approach divided in the following two sections: *Process pattern analysis* and *Process analysis* (not presented here but available to read in [3]). 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/activity. *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 and offer the possibility to *compare* different patterns from different variants and select one as best chosen example. In the dashboard are summarized different aggregated measures (average pattern duration and nr of participants in a pattern) with corresponding lines charts as shown in Figure 6. We can filter the results to only a specific year/quarter/month or date. In part (b) of the following figure, an expanded view of the GA3 pattern is given to show the values for each activity that are comprised in a pattern among the five different variants. The interactive dashboard results are available in this github link.[4] We can conclude that *Mun-1* report the highest value for number of participants working on different activities whereas *Mun-3* the above average values, followed by *Mun-5*, *Mun-2* and *Mun-2* for the overall period of 4 years. Instead, *Mun-3* reports the best performance based on time aggregated measures whereas *Mun-2* the worst one for four years period.

[4] `https://github.com/lisanab/pwh/`



(a) Dashboard of process variant analysis (collapsed view)



(b) Dashboard of process variant analysis (expanded view)

Fig. 6: Dashboard of process variant analysis

5 Related Work

This section provides an overview of current process-oriented data warehouses approaches to analyse business processes through the most important analysis parameters which are elicited from the generic meta-model of business processes. [2] proposes to derive a generic data warehouse structures from the meta model of the BPMN, whereas [11] proposes a Sequence Warehouse (SeWA) architecture and OLAP tools to analyse data stemming from workflow logs but a conceptual model for DW is missing. Approaches based on goal-oriented methodology for requirement analysis in order to design a data warehouse were proposed by [10, 15, 19]. A multidimensional data modelling for business process analysis was proposed by [7, 13, 14, 16] During our research work we have found a number of relational and multidimensional data warehouse design used for process mining analysis as well. According to approach in [1], 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 data cubes and associated operations such as slice, dice, roll-up, and drill-down. Authors in [18] introduced an event cube as a basis for process discovery and analysis. A framework is further developed to realize a process cube allowing for the comparison of event data in [12]. A hierarchy level was defined only in the time dimension. An adaptation is needed to accommodate event data through multidimensional process mining which is far from trivial given the nature of event data which cannot be easily summarized or aggregated, conflicting with classical OLAP assumptions. For instance, multidimensional process mining can be used to analyse the different versions of a sales process, where each version can be defined according to different dimensions such as location or time, and then the different results can be compared as proposed in [5]. Furthermore, authors in [20] partition event logs into groups of cases called sublogs with homogeneous features in a dynamic and flexible way, in order to manage comparisons between models.

6 Conclusions

This paper proposed a process warehouse approach to efficiently and effectively analyse a family of process variants. 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. 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. As a summary, based on the consumption of PW in many business intelligence development and solutions, a framework that allows process variants to be efficiently analysed can significantly improve the state-of-art.

References

[1] van der Aalst, W.M.P.: Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. Springer Verlag (2013)

[2] Benker, T.: A generic process data warehouse schema for bpmn workflows. Springer International Publishing (2016)

[3] Berberi, L.: Analysis of Process Variants with a Process Warehouse Approach. Ph.D. thesis (2021)

[4] Berberi, L., Eder, J., Koncilia, C.: A process warehouse model capturing process variants. EMISA Journal (2018)

[5] Bolt, A., van der Aalst, W.M.P.: Multidimensional process mining using process cubes. Springer International Publishing (2015)

[6] BPMN(Spec.): About the business process model and notation specification version 2.0. Tech. rep., OMG, https://www.omg.org/spec/BPMN/2.0/PDF

[7] Casati, F., Castellanos, M., Dayal, U., Salazar, N.: A generic solution for warehousing business process data. VLDB Endowment (2007)

[8] Dongen, V., Boudewijn, B.: Bpi challenge 2015 (2015)

[9] Eder, J., Olivotto, G.E., Gruber, W.: A data warehouse for workflow logs. Springer (2002)

[10] Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented requirement analysis for data warehouse design. ACM (2005)

[11] Koncilia, C., Pichler, H., Wrembel, R.: A generic data warehouse architecture for analyzing workflow logs. Springer (2015)

[12] Mamaliga, T.: Realizing a Process Cube Allowing for the Comparison of Event Data. Master's thesis, TU Eindhoven (2013)

[13] Mansmann, S., Neumuth, T., Scholl, M.H.: Olap technology for business process intelligence: Challenges and solutions. Springer Berlin Heidelberg (2007)

[14] Neumuth, T., Mansmann, S., Scholl, M.H., Burgert, O.: Data warehousing technology for surgical workflow analysis. IEEE (2008)

[15] Niedrite, L., Solodovnikova, D., Treimanis, M., Niedritis, A.: Goal-driven design of a data warehouse-based business process analysis system. AIKED'07, World Scientific and Engineering Academy and Society (WSEAS) (2007)

[16] Pau, K.C., Si, Y.W., Dumas, M.: Data warehouse model for audit trail analysis in workflows. IEEE (2007)

[17] Polyvyanyy, A., Ouyang, C., Barros, A., van der Aalst, W.M.: Process querying: Enabling business intelligence through query-based process analytics. Decision Support Systems (2017). https://doi.org//10.1016/j.dss.2017.04.011

[18] Ribeiro, J.T.S., Weijters, A.J.M.M.: Event cube: Another perspective on business processes. Springer Berlin Heidelberg (2011)

[19] Shahzad, K., Zdravkovic, J.: Process warehouses in practice: a goal–driven method for business process analysis. Journal of Software: Evolution and Process (2012)

[20] Vogelgesang, T., Appelrath, H.J.: A relational data warehouse for multidimensional process mining. Springer (2015)