IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 25, 2021, accepted July 15, 2021, date of publication July 30, 2021, date of current version August 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101575

Modeling Variability in the Performance Perspective of Business Processes

BEDILIA ESTRADA-TORRES[1], ADELA DEL-RÍO-ORTEGA[2], MANUEL RESINAS[2], AND ANTONIO RUIZ-CORTÉS[2], (Member, IEEE)

[1]Department of Computer Languages and Systems, Universidad de Sevilla, 41012 Seville, Spain
[2]Smart Computer Systems Research and Engineering Laboratory (SCORE), Research Institute of Informatics Engineering (I3US), Universidad de Sevilla, 41012 Seville, Spain

Corresponding author: Bedilia Estrada-Torres (iestrada@us.es)

This work was supported in part by the FEDER/Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación under Project HORATIO/OPHELIA (RTI2018101204-B-C21/RTI2018101204-B-C22), and in part by the Andalusian Administration through the Project APOLO (US-1264651), and the Project EKIPMENT-PLUS (P18-FR-2895).

ABSTRACT The modeling and management of business processes often leads to the definition of several variants of the same process. This variability can be reflected in different process perspectives such as control-flow, data, resources or performance. The management of process variants can be a laborious, time-consuming and error-prone task since they require a high coordination in the management of each variant and in most cases this management is done manually. For this reason, many proposals have been developed to deal with the variability of business processes. However, none of them covers in detail the variability in the performance perspective, which is concerned with the definition of performance requirements usually specified as a set of Process Performance Indicators (PPIs). This variability can be reflected in the form of repetitive and redundant PPI definitions, and can lead to errors and inconsistencies in PPI definitions. To address this problem, in this article we propose a detailed PPI variability classification and a formalization of how PPIs can be modeled together with the variability of other process perspectives. To this end, we considered variability management approaches, called by restriction and by extension, and we illustrated our proposal by integrating it with existing variability modeling languages. An evaluation conducted in two scenarios shows the feasibility and usefulness of our proposal.

INDEX TERMS Business process, performance indicators, performance management, PPIs, process performance, variability management and modeling.

I. INTRODUCTION

The definition and modeling of business processes may vary slightly from one context to another within an organization, for example, to adapt to new requirements [1] and business strategies [2], by regulations in different countries or regions [3] and/or to reflect new resource allocations and responsibilities [2].

In general, variability has been defined as the ability to change or customize a system to make a set of changes easier [4] and as the ability of software and artifacts ''to be extended, changed, customized or configured in a specific context'' [5]. Variability modeling techniques help understand and define the commonalities and variations between software product lines [6]. In the context of business processes, each variation of a given process is called a business process variant, hereinafter, *variant* for short. Process models can exist as collections of different *variants* [7]–[9] that share a common base structure and some strategic and business goals. This variability has an impact on the time and cost spent for the modeling and maintenance of business processes, promotes the reuse of portion of process models, and it also has an impact on process management quality, because it helps manage redundancies and inconsistencies [10]. The set of *variants* of the same process is known as a business process family, hereinafter *process family* [1], [11].

Often variability is not modeled and managed in a specific way, but each *variant* is modeled either as a separate process, or as a set of conditional branches [12]. Both modeling alternatives result in process models that are more complex

The associate editor coordinating the review of this manuscript and approving it for publication was Wen-Sheng Zhao.

to manage, require more resources for their management, and are more prone to errors. Those process models sometimes contain a high level of redundancy, inconsistency or outdated information [12], [13]. To address this problem, several techniques to model business process variability have been proposed in the last decade [9]. They are aimed at avoiding duplication and redundancy of information through the reuse of some parts of the process model, by identifying its common parts and modeling a business process block only once, thus reducing design time and model maintenance time [14]. The idea, then, is to define a single model, from which each *variant* can be derived using certain transformations. This can be carried out in two ways: *by extension* or *by restriction*, depending on whether the behavior of a unified process model is extended or restricted to suit the requirements of each particular *variant* [9]. These proposals cover different perspectives of business process variability [9]. One of the most studied perspectives is the one related to control-flow, where a number of works exist that address how to model process families that present variability in this perspective [12], [15]–[19]. Other approaches related to variability in resources and data (objects) perspectives have also been proposed [9], [19]–[21].

The performance perspective of business processes, however, has been disregarded up to date and, as far as we know, there are not solid proposals that address variability in it. This perspective is concerned with the definition of performance requirements addressing different performance dimensions such as time, cost and quality [22], and it is usually expressed as a set of Process Performance Indicators (PPIs). PPIs are quantifiable metrics that allow the evaluation of efficiency and effectiveness of business processes and can be measured using data generated within the process [23]. They also allow the analysis of whether the objectives of an organization are met and help the decision making within the organization. The modeling of PPIs seeks to provide structures and notations that allow the user to define, in a complete, understandable, verifiable and unambiguous way, how the performance of the processes will be measured [24]. Like other perspectives of business processes, the performance perspective is also susceptible to variability [25]. The lack of a mechanism to manage the variability of the performance perspective means that, for instance, if a PPI is defined on all or some *variants*, the PPI has to be modeled and managed individually for each *variant* in which it is defined, although for all those variants the PPI might share their attributes or characteristics. Therefore, as it happens with the other perspectives, the management of such PPIs become a repetitive, laborious and error-prone task.

The goal of this article is to provide mechanisms to manage the variability of PPIs together with the other perspectives of business processes. To help guide our research, we formulated the following three research questions:

RQ1 *What types of variability should be taken into account in the performance measurement of a business process?*

RQ2 *How can the variability of the business process performance perspective be modeled?*

RQ3 *How can the existing techniques for modeling PPIs be integrated with current alternatives for variability management?*

To answer these research questions, we followed the principles of the design science research methodology described in [26]. Specific actions taken are described in Section IV.

This article extends a previous work [25]. In it, we introduced (i) the classification of the different types of variability identified in the performance perspective, which we called *dimensions of change*. We (ii) provided a preliminary extension of a metamodel used for PPIs modeling, called PPINOT [23]. Based on this extension, we proposed the first approach to the definition of variability from a performance perspective. However, that extension was limited to variability by restriction and could not be used together with any existing variability management approach. In our current proposal, we extended this previous approach in the following ways:

i We redefined and expanded a formalization to reflect in more detail how PPI variability can be modeled taking into account the relationship between PPI variability and other process perspectives such as control-flow.

ii We reformulated the variability modeling using the two existing approaches to variability management, variability by restriction and by extension. This is performed by means of the extension and integration of the PPINOT metamodel together with two well-known proposals for each of the two aforementioned approaches for the variability management: C-iEPC [21] (by restriction) and PROVOP [3] (by extension). Furthermore, for each of the approaches, we formalized how variants are derived from the variability model and define the conditions that the variability model must meet to obtain syntactically correct variants.

iii We expanded and modified the evaluation of the proposal. This article reports on an evaluation that involves two scenarios: a set of measures and PPIs defined in the SCOR model and the IT incident management process of a public organization in Spain. In both cases, we analyze the feasibility and applicability of the dimensions of change and show how our approach can be successfully used to model the variability that is present in the PPIs defined in those scenarios.

The remainder of this paper is structured as follows. Section II describes the main concepts related to variability in business processes and PPIs, and the related work in those areas. Section III introduces our motivating scenario. Section IV presents the research method followed. Section V describes the kinds of variability identified and the dimensions of change proposed. Section VI formally defines PPINOT as the basis for variability extensions. Section VII describes the PPINOT extension considering two types of variability management: variability by restriction and by extension. The evaluation of the proposal based on

two variability scenarios is described in Section VIII. A brief discussion is presented in Section IX. Finally, Section X outlines conclusions and discusses future work.

II. BACKGROUND AND RELATED WORK

This section introduces the main concepts and related work regarding variability in business processes and PPIs.

A. VARIABILITY IN BUSINESS PROCESSES

The cause of variability can be linked to organizational questions such as [7]: *How* is each *variant* implemented? (*Operational variation*); *When* do organizations produce/offer several products if its execution depends on the product to be provided? (*Product/service variation*); *Where* is the process applied? (*Market variation*); *Who* is the company dealing with? (*Customer and stakeholders variability*); and *When* is the process executed and if is it influenced by external factors? (*Time variation*). Regardless of the causes of variability, when it is not explicitly managed, each variation in the process is represented as an independent process model; thus ensuring the representation of all information, but generating a large amount of models, redundant information and making future adaptations difficult. When repositories have hundreds or thousands of process models, similarity measures, such as those discussed and proposed in [27], [28], can be useful to measure conformance between reference and actual models, to identify similar models in a repository or to locate those models that conform to a given specification.

The lack of control over multiple *variants* usually causes an increase of the time required to design, configure and modify each *variant*, and may introduce errors from their definitions to the evaluation of its performance [12], [13]. To deal with these issues, many approaches have been proposed to manage variability in business processes. Most of them focus on the *design and analysis* phase of the business process lifecycle [29]. They are aimed at avoiding duplication and redundancy of information through the reuse of some parts of the process control flow, by identifying the common parts of the flow and modeling them only once, thus reducing design time and model maintenance time [14].

There are two ways of approaching the modeling of the variability of business processes [9]. In the first one, each *variant* is modeled separately from another, which can lead to redundancies and inconsistencies due to the information shared by the *variants*. The second way is the definition of a single model, from which each *variant* can be derived using certain transformations; this generates fewer models, but they are often more complex and more difficult to understand. Within the second option, variability management can be carried out in two ways: *by extension* or *by restriction*, depending on whether the behavior of a unified process model is extended or restricted to meet the requirements of each particular *variant* [9].

For managing variability by extension, a *customizable* process model represents the most common behavior or the behavior that is shared by most *variants* and then its behavior is expanded to meet the requirements of each *variant*. Examples of modeling languages to manage variability by extension are PROVOP [3], [15] and Business Process Family Model (BPFM) [8], [16]; the former can be used with any business process modeling language, while the latter is specific for UML Activity Diagrams. Those proposals, like most of the approaches in this category, also allow restrictions on the behavior of the model. For managing variability by restriction, a *customizable* process model is called *configurable* process model and contains the behavior of all *variants*. The customization and derivation of a *variant* is made by restricting the behavior of the customizable process model. C-EPC [18], C-iEPC [9], [21], PESOA [19] or Feature Model Composition [20] are examples of business process modeling languages for managing variability by restriction. Many of these variability management approaches can be used in conjunction with different business process modeling languages. For example, PESOA can be applied over UML ADs or with BPMN; ADOM [30] can be used with UML ADs, EPCs or with BPMN [31]; and BPMN* [32] and BPMNext [33], which are considered extensions of BPMN for modeling variability. Along the same lines, the proposal of Common Variability Language (CVL) [34] allows expressing variability in a language independently of the base modeling language. Although CVL was not intended exclusively for process variability, approaches such as those described in [35] and [36] combine BPMN with CVL.

Despite the wide variety of approaches to manage process variability from a control-flow [12], [15]–[19], data [9], [20], [21] or resource perspective [9], [19]–[21], the performance perspective remains disregarded. However, we consider that the benefits derived from these approaches, such as reuse in the definition of process models and coordinated variability management, could also be applied to the performance perspective.

B. PROCESS PERFORMANCE INDICATORS AND THEIR VARIABILITY MANAGEMENT

Although a business process may change from one context of implementation to another, the way in which its performance is measured can be standardized [37]. Several frameworks and notations have addressed the definition and management of PPIs and performance measures [23], [24], [38]–[43].

PPIs are defined by means of a set of attributes that specify relevant information to establish what and how to measure the process performance [23], [41]. The most relevant and recurrent attributes that represent a PPI are the generic attributes required to identify the PPI, a set of attributes to relate the PPI to the process and its objectives, and other attributes focused on defining how the PPI is calculated. There is a wide variety of approaches, techniques and tools for measuring process performance, but no solid proposal has been found to address variability from the performance perspective.

The proposal presented in [44] describes an industrial case study on the variability of the calculation of performance

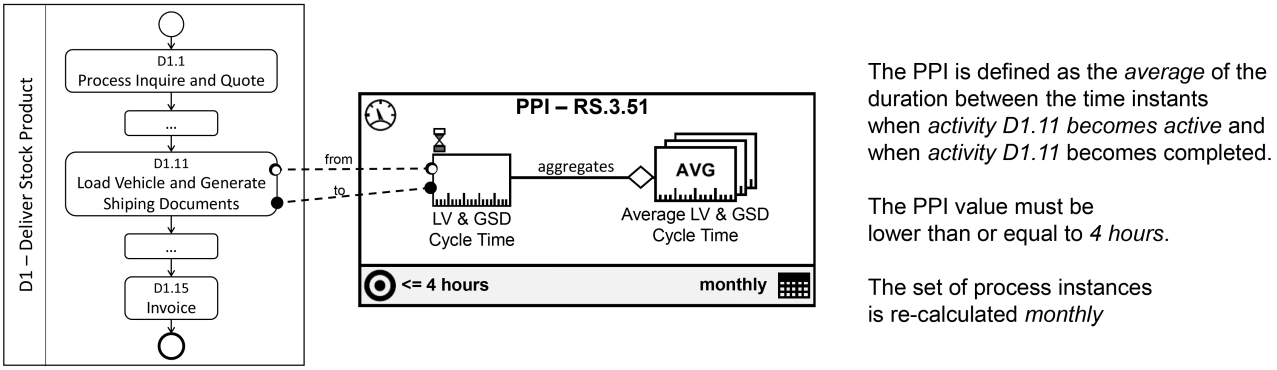


FIGURE 1. Example of a PPI definition modeled using Visual PPINOT.

indicators in the business process families of a public water distribution agency. The authors propose to extend the BPFM notation [8] to model and support variations in KPIs. That proposal was extended in [45], where a tree model of PPIs variability is proposed as an extension of the BPFM model. The tree is defined using a PPI reference model and defining a series of variants related associated with certain constraints. In this approach, ''a family of PPIs refers to the calculation variability of a set of PPIs sharing a certain number of operators'', however, it does not specify what is the relationship between values of the indicator attributes and the PPI variability. Our proposal, besides taking into account the variability of PPIs and reflecting its traceability with the process or process family where it is defined, it also provides a detailed structure that allows the variability modeling considering the different PPI attributes. Another advantage of our proposal is that it is not tied to a single modeling approach, but rather we provide a formal structure that can be used and integrated along with a wide variety of existing process variability modeling approaches and allows for variability management by both restriction and extension.

Two other proposals, [46] and [47], also discuss concepts related to variability and indicators, however, in those scenarios variability is not addressed from the point of view of business processes. In both proposals the indicators are defined taking into account some PPI attributes such as the goals or the measure to calculate the PPI. However, both proposals lack a solid and detailed structure on which to build the indicators in a clear and unambiguous way.

C. INTRODUCTION TO PPINOT

PPINOT is a metamodel that allows the definition of performance models composed of a set of *PPIs* [23]. In PPINOT, a PPI is defined by means of a set of attributes namely, an *identifier*, a descriptive *name*, a *process* in which the PPI is defined, a set of *goals* indicating the relevance of the PPI, a measure definition that specifies how to calculate the PPI, a *target* value to be reached indicating the fulfillment of the previously defined goals, the *scope* that is used to define the subset of instances to be considered to calculate the PPI value, a human resource *responsible* for the PPI, and human resources to be *informed* about the PPI.

Figure 1 shows a PPI definition example using the graphical notation of Visual PPINOT [42]. The PPI *RS.3.51* is defined over the SCOR *process* Deliver Stock Product. This PPI is calculated as the average of the cycle time between the start and the end of the execution of the task *D1.11 - Load Vehicle and Generate Shipping Documents*. The expected value for the PPI must be lower than or equals to 4 hours, calculated monthly over all instances (scope).

PPINOT provides a wide variety of measures, namely: (i) Base Measures represent a single-instance measure that can be instantiated as one of four types: *Time Measures* calculate the duration of time between two time instants; *Count Measures* indicate the number of times something happens; *State Condition Measures* evaluate the fulfillment of certain condition in a process instance; and *Data Measures* measure the value of a certain attribute of a data object. (ii) Aggregated Measures is the second type, which are defined by aggregating one of the base measures applied to several process instances; and (iii) Derived Measures represent either a single-instance or a multi-instance measure whose value is obtained by calculating a mathematical function over other measures. The traceability between measures and business process elements (i.e., tasks, events, data objects) is kept by means of *Conditions* that indicate how and when to take values from the process, and *Data Content Selections* to obtain an attribute of a data object. More specifically, a Time measure requires two conditions to indicate the start (*from*) and end (*to*) points of the measurement; a Count measure needs a *when* condition that indicates the point when something happens and should be measured; a State condition measure uses a *meets* condition to indicate the condition whose fulfillment is being measured; and a Data measure uses a *measuresData* condition to select the attribute of the data object that is being measured.

Figure 2 shows the PPINOT metamodel [23], which represents the concepts described above. In this article, we decided to use PPINOT for illustrating and modeling variability in

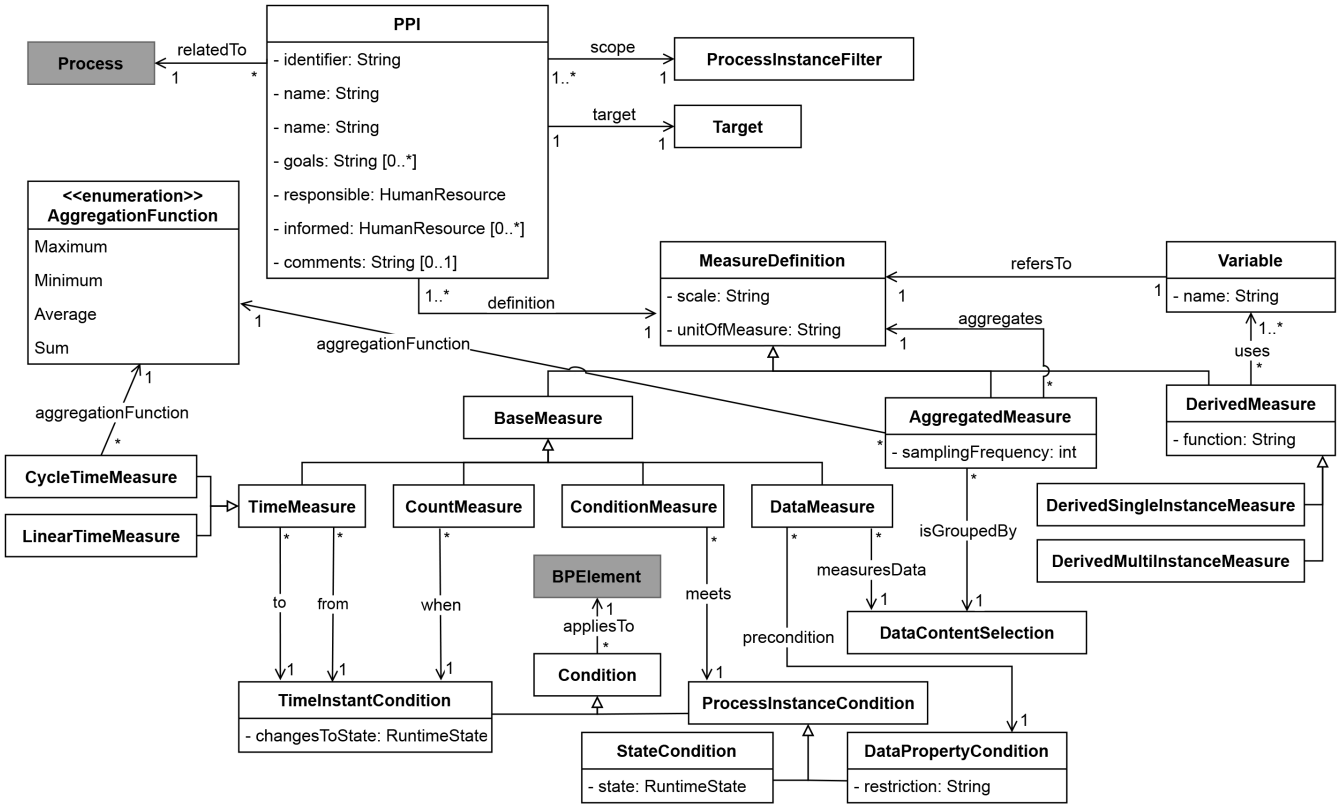


FIGURE 2. The PPINOT metamodel. Image based on [23].

the modeling of PPIs because, as described in the analysis presented in [23], PPINOT is highly expressive, allows PPI definitions in an unambiguous and complete way and facilitates traceability between business process elements and PPIs. In addition, PPINOT provides two notations for modeling PPIs. One is a template based notation and linguistic patterns [48] that provide pre-written sentences and placeholders to be filled in, to facilitate the definition of PPIs. The second notation is Visual PPINOT [42], a graphical notation that provides icons for the definition of PPIs and its measures, as well as its connections with the process model. In addition to the above, PPINOT also facilitates the automatic calculation of PPIs and their subsequent management. This proposal is not limited to the use of PPINOT, but can be applied to other performance indicator modeling proposals.

III. MOTIVATING SCENARIO

The definition of business processes is usually derived from the strategic lines and goals of each company. However, some organizations have taken initiatives to propose general recommendations that can be used or adapted to different scenarios.

The *Supply Chain Operation Reference* model (SCOR) [49] is a reference model for evaluating and comparing supply chain activities and performance that enables users to address, improve, and communicate supply chain management practices within and between all company stakeholders [50]. We focused on two elements of its structure: *processes* and *measure definitions* (called *metrics,* in *SCOR*). *SCOR processes* identify a set of unique activities within a supply chain. These activities are described at a high level of abstraction since the implementation of processes requires internal and specific definitions of the activities of each organization, which are out of the scope of SCOR. *SCOR measure definitions* are defined as a standard for measuring the process performance. These measures are classified into 5 categories: Reliability (RL), Responsiveness (RS), Agility (AG), Costs (CO) and Asset Management (AM). In this article, we provide examples of RS, but the concepts presented could be applied to any type of SCOR measure.

Due to its structure and the definition of its components, *SCOR processes* have intrinsic variability. *Deliver process (D)*, for instance, is defined as the processes associated with performing customer-facing order management and order fulfillment activities. It can be implemented in four different ways depending on the selected strategy: *D1-Stocked Product (variant PV-1), D2-Make to Order Product (variant PV-2), D3-Engineering to Order Product (variant PV-3)* and *D4-Retail Product (variant PV-4)*. Although they pursue the same goal, the way to achieve it differs from strategy to strategy. PV-2 varies in 13% with respect to activities defined for PV-1; PV-3 differs in 33% and PV-4 differs in its entirety

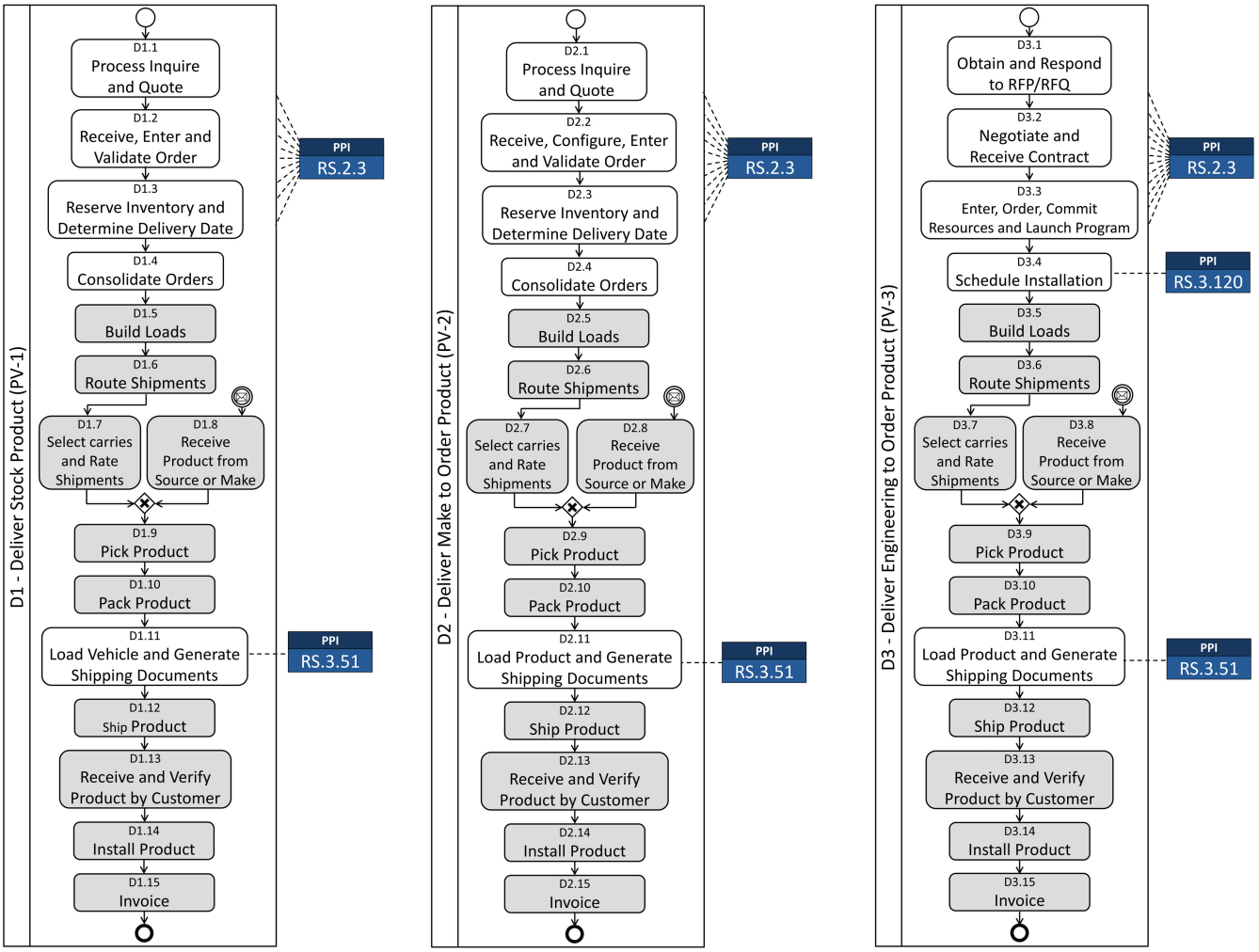


FIGURE 3. Three *variants* of the *Deliver* SCOR process. The shaded activities are common to all *variants*. The dashed lines indicate the number of measures (process values) required to calculate the PPIs.

from PV-1. For simplicity, we only focus on the three first *variants*, which are shown in Fig. 3, because *D-4* is totally different from the other *variants*.

Variability is also reflected in SCOR through its *measure definitions*, (i) due to their dependence on the process control-flow in which they are defined or (ii) by specific requirements of the measures defined for each variant. Measures like *RS.3.120 Schedule Installation Cycle Time*[1] reflect the first case. The measure is defined only in one *variant*, because it is connected to the task *D3.4 - Schedule Installation* that only appears in PV-3. The second case is shown in measures that vary regarding the required components to calculate its value. For example, measure *RS.2.3*, named *Deliver Cycle Time*, is calculated differently in each *variant* where it is defined. The measure requires 9 time values in PV-1, 10 time values in PV-2 and 12 time values in PV-3.

[1]In this article we use the same measures indicators used in the SCOR model

Managing this variability could have a significant impact in the effort of modeling and maintaining the PPIs of a SCOR process. In SCOR, for example, process Deliver defines 100 measures for PV-1, 96 measures for PV-2, and 96 measures for PV-3. Almost half of them are repeated for all or several *variants*. To model them, it would be necessary to independently model the PPIs of each *variant*, making it a endeavouring task. Furthermore, if a PPI definition changes, it would require the modification of each and any *variant* involved, which threatens the PPI integrity through all *variants*, because some PPIs might be disregarded and not consequently updated, becoming obsolete or inconsistent. If these errors go undetected, they may be carried over the entire process lifecycle. This can lead to new trouble spots, such as poor monitoring of PPIs and inaccurate information collection that could eventually lead to a misguided decision making.

In summary, modeling variability of PPIs could bring similar advantages than modeling variability in the other business process perspectives. Consequently, new techniques and tools

for the definition of PPIs that allow the management of variability in the performance perspective need to be devised. Furthermore, since the variability of PPIs is closely related to the variability of the activities and control-flow of the process, it is convenient that the techniques developed for variability management of PPIs can be integrated to those designed to manage the variability of the other perspectives of the business process.

IV. METHOD

Given the scenario described above and the potential pitfalls identified, and with the aim of answering the research questions introduced Section I, we followed the principles of the design science research methodology described in [26]. In our research, the different phases consisted of:

- *Problem identification and motivation phase*. This phase consists of two parts. First, in conducting the literature review, we found that there is a wide variety of approaches that highlight the need to manage business process variability. They mostly address the variability management of control-flow, resources, or data, but we identified that the performance perspective has been disregarded to date. Based on the literature analyzed, it was identified that there is variability in the performance perspective but also a lack of techniques to facilitate its management. In addition to the literature review, we also identified actual scenarios in which variability is present in both control-flow and performance perspective. In these cases, variability is not treated in a special way, so it may suffer from problems of redundancy and inconsistencies mentioned above.
- The *objective of the solution* is to provide a mechanism that allows the modeling of several cases of variability related to PPIs together with the variability of the other perspectives of a business process.
- *The design and development phase* includes the design and development of an artifact that contains the concepts and restrictions required to model PPIs, taking into account the different types of variability, the notations that facilitate its modeling, and techniques to integrate it to other proposals. This is an iterative phase in which the resulting artifact is a metamodel for the definition of PPIs and the variability associated with them. The metamodel is not built from scratch, but is based on an existing PPI modeling approach, PPINOT. A formal definition reflecting the elements and constraints described by the metamodel is defined and provided. Furthermore, we detail how *variants* are derived from a PPI variability model and define the conditions that the variability model must meet to obtain syntactically correct variants.
- In *the demonstration phase*, the proposal for modeling PPIs was integrated with two business process modeling languages capable of managing business process control-flow variability: one that manages variability by extension and the other one by restriction. To select the modeling languages to be integrated, we relied on the study presented in [9], which describes a large set of variability management approaches. We selected the two approaches most cited in that study, one for the management by restriction, C-iEPC, and the other for the management by extension, PROVOP. To demonstrate the applicability of our proposal, we extended the graphical notation of PPINOT with the variability concepts identified and we used it in conjunction with the extended graphical notations of PROVOP and C-iEPC in examples of modeling variability in the control-flow and the performance perspective of business processes. In this way, it is shown the flexibility of the proposal to be adapted and integrated with other modeling proposals and its independence from the modeling language used.
- Finally, in the *evaluation phase*, an exploratory case study was conducted. Two scenarios were analyzed addressing the modeling of SCOR processes and measures, and PPIs of the IT incident management process in a Spanish public organization. The case study sought to assess the feasibility and usefulness of the proposal to model PPIs on process families.

V. DEFINING VARIABILITY IN PPIs

Despite the studies related to variability in PPIs, we did not find a definition of when one PPI is a *variant* of another and when it is considered a new PPI. To clarify this concept, we based on the literature related to variability modeling in business process in general, and in PPI definitions in particular, to identify and define the different characteristics that can make a PPI vary. As a result of this analysis, in our approach, given a *process family* and a set of PPIs that have the same business goals, we consider a PPI is a *variant* of another PPI when:

i. the PPI is defined in the same way for some of the *variants* of the process family, that is, the values of its attributes are the same for all *variants* to which it applies, but the PPI is not defined for all *variants* **(DimC-1)**.
ii. at least one of the values of its attributes changes from one *variant* to another regardless of whether the PPI is defined for all or just several *variants* **(DimC-2)**.

We call these two characteristics *dimensions of change* in PPI definitions. As shown in Fig. 4, four subdimensions are derived from DimC-2, one associated with each attribute that can vary in a PPI definition. In addition, other two subdimensions are derived from *DimC-2.M (Measure)* (see Fig. 4). We provide further details on this in the following.

Let us suppose a *process family* that consists of two or more *variants*. If a PPI is defined for all those *variants* and all its attributes are set with the same values for all *variants*, there is no variability. Instead, if a PPI is defined in some of the *variants* of the business process, but not for all of them, we are representing the variability expressed by DimC-1, regardless of whether their attributes change or not. In the example shown in Fig. 3, if we assume a *process family* consisting of PV-1, PV-2, and PV-3, the PPI *RS.3.51* has no variability

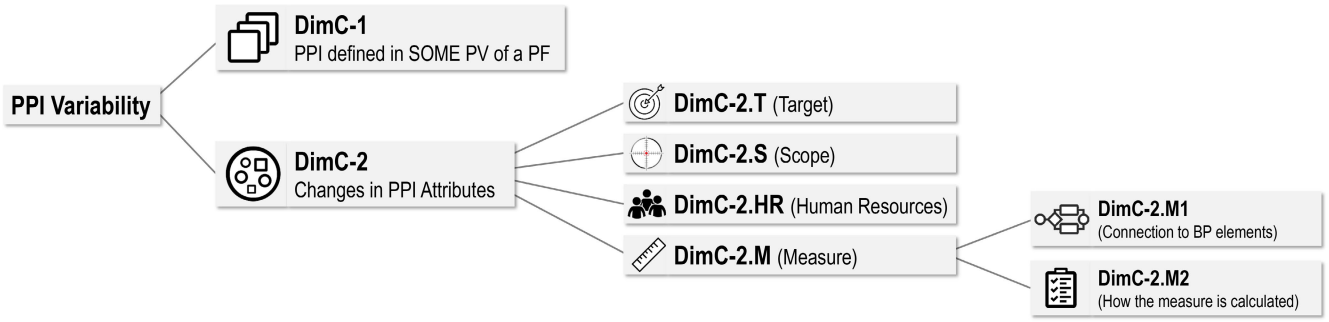


FIGURE 4. Dimensions of change related to PPI definitions.

according to DimC-1 because it appears in all *variants* of the *process family*. The PPI *RS.3.120*, however, has variability according to DimC-1.

In addition, a PPI may also vary depending on the changes applied to the value of one or more of its attributes, which is related to the definition of the DimC-2. In Section II-B, we mentioned the attributes of a PPI. Here we list the cases where PPI variability can happen, considering that a PPI varies if at least one of the following attributes changes:

- **Target *(T)*** changes when the target value to be reached changes. PPI *RS.3.51* of our scenario calculates the cycle time of the task where it is defined. The target value for PPI *RS.3.51* could be defined as <= 3,5 hours for PV-1, <= 4, 2 hours for PV-2 and <= 4, 5 hours for PV-3.
- **Scope *(S)*** changes when the set of instances to be evaluated changes. For example, if we have one *variant* that applies during weekdays and another one that applies in weekends (e.g., due to limited availability of resources during weekends), we might define two variants of the same PPI, one that evaluates instances that occur on weekdays, and another one that evaluates those that occur on weekends.
- **Human resources *(HR)*** may change by two attributes: *responsible* and *informed*. For example, depending on the *variant* where the PPI is defined, the person responsible for the PPI or the persons informed about it might change.
- **Measure definition *(M)*** is a complex attribute through which a PPI is calculated. In this case, there are two subdimensions of change, one related to the measure definition itself and another one related to the relationship with the business process.
- **DimC-2.M1:** A measure definition is calculated in the same way, but its definition may vary depending only on the business process element to which it is connected. For example, those that provide the information required to compute the PPI value.
- **DimC-2.M2:** A measure definition changes the way in which it is calculated.

DimC-2.M1 occurs when a PPI is connected to a business process element such as a task, that is not available for all *variants*, or when for certain *variants*, the PPI has to be defined over a different task for some reason. An example of DimC-2.M1 is the PPI defined over the SCOR measure *RS.3.51 - Load Product & Generate Shipping Documentation Cycle Time*, which is defined in the Deliver process over task 11. In PV-1, this PPI is computed over task *D1.11 - Load Vehicle & Generate Shipping Documents*, but in PV-2 and PV-3 this task is not available (See Fig. 3). For this reason the same PPI is defined over an equivalent task, *D2.11, D3.11 - Load Product & Generate Shipping Documents*.

An example of DimC-2.M2 is the *RS.2.3 - Deliver Cycle Time* measure definition (see Fig. 3) that is defined in PV-1, PV-2 and PV-3, but in each *variant* that measure requires different information: 9, 10 and 12 time values calculated over different activities in each *variant*, respectively.

VI. FORMAL DEFINITION OF PPINOT

The PPINOT metamodel [23] allows the definition of a *performance model* as a set of PPIs. In this section, we formally describe the PPINOT components that will later be adapted to take into account the dimensions of change related to variability. In this way, we also seek to avoid possible incomplete, vague or confusing definitions of PPIs, and to be able to adapt this proposal to different contexts and scenarios.

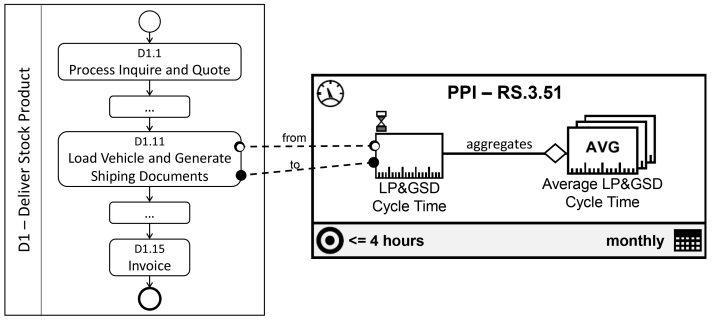
To formally define a PPINOT performance model, we first formalize the concept of *Condition*, which is the link between the performance model and the other elements of a business process.

Definition 1 (Condition): Let bp be a *business process*, $\mathcal{A}$ be a not empty set of *activities* for bp, $\mathcal{S}_\mathcal{A}$ be a set of *activity states* of $\mathcal{A}$, $\mathcal{D}$ be a finite set of *data objects* for bp, $\mathcal{S}_\mathcal{D}$ be a finite set of *data object states* of $\mathcal{D}$, $\mathcal{A}_\mathcal{D}$ be a non-empty set of *data object attributes* of $\mathcal{D}$, $\mathcal{E}$ be a non-empty set of *events* for bp, $\mathcal{S}_\mathcal{E}$ be a set of *event states* of $\mathcal{E}$. $\mathcal{C}_{bp} = \mathcal{A} \times \mathcal{S}_\mathcal{A} \cup D \times \mathcal{S}_\mathcal{D} \cup \mathcal{E} \times \mathcal{S}_\mathcal{E}$ is the set of all possible *Conditions* that can be defined over bp.

For example, a condition $\mathcal{C} = (ActivityA, active)$ represents the moment when activity *ActivityA* becomes *active* in a given running instance.

Now, a PPINOT performance model can be defined as follows.

Definition 2 (PPINOT Performance Model): Let bp be a business process, $\mathcal{C}_{bp}$ be the set of all possible conditions



$PM_1 = (P_1, M_1, L_{P1}, L_{M1})$
$P_1 = \{RS.3.51\}$
$M_1 = \{LP\&GSD\ Cycle\ Time,\ Average\ LP\&GSD\ Cycle\ Time\}$
- $LP\&GSD\ Cycle\ Time \subseteq TimeM \subseteq BM$
- $Average\ LP\&GSD\ Cycle\ Time \subseteq AggM$

$L_{P1} = \{scope, target\}$
- $scope = \{RS.2.51, \text{`<= 4 hours'}\} \subseteq sco$
- $target = \{RS.2.51, \text{`monthly'}\} \subseteq tar$

$L_{M1} = \{from, to, aggregates\}$
- $from \subseteq M_{time} \times C = \{LP\&GSD\ Cycle\ Time, \{D1.11, Active\}\}$
- $to \subseteq M_{time} \times C = \{LP\&GSD\ Cycle\ Time, \{D1.11, Completed\}\}$
- $aggregates \subseteq agg \subseteq \{M_{agg}=\text{`Average LP\&GSD Cycle Time'}, M_{time}=\text{`LP\&GSD Cycle Time'}, F_{agg}=\text{`AVG'}\}$

FIGURE 5. Example of PPINOT and its relationship with the formal definition.

defined over bp, $\mathcal{S}$ be the set of *scopes* that can be defined for a PPI, $\mathcal{T}$ be the set of targets that can be defined for a PPI, $\mathcal{HR}$ be the set of human resources that can be related to the PPI, $\mathcal{M} = \{time, count, state, data, agg, der\}$ be the set of types of measure definitions that can be defined, $\mathcal{B}$ be the business schedule used to compute time measures (e.g., from 9 to 5, or 24 hours), $\mathcal{F}_{agg} = \{MIN, MAX, AVG, SUM, \ldots\}$ be a set of *aggregation functions*. A performance model PM over $\mathcal{S}$, $\mathcal{T}$, $\mathcal{HR}$, $\mathcal{C}_{bp}$, $\mathcal{M}$ and $\mathcal{F}_{agg}$ is a tuple $PM = (P, M, L_P, L_M)$, where:

- P is the set of *process performance indicators* of a bp;
- M is a set of *measure definitions*;
- $L_P = sco \cup tar \cup res \cup inf \cup mes$ is the set of links between a PPI $p \in P$ and its attributes, where:
  - $sco \subseteq P \times \mathcal{S}$ is the set of *scope links* assigned to each PPI;
  - $tar \subseteq P \times \mathcal{T}$ is the set of *target links* assigned to each PPI;
  - $res \subseteq P \times \mathcal{HR}$ is the set of human resource links to indicate the person *responsible* of the PPI;
  - $inf \subseteq P \times \mathcal{P}(\mathcal{HR})$ is the set of human resource links to indicate the people *informed* about the PPI, where $\mathcal{P}(\mathcal{HR})$ is the power set of $\mathcal{HR}$;
  - $mes \subseteq P \times M$ is the set of links with the *measure* that defines each PPI;
- $L_M = \tau \cup cond \cup data \cup agg \cup cyclic \cup cal \cup uses$ is the set of links between *measure definitions* and its attributes, where:
  - $\tau \subseteq M \times \mathcal{M}$ is a set that maps each measure definition $m \in M$ with its type $\mathcal{M}$. We use M_x, where $x \in \mathcal{M}$ to refer to the subset of measures with type x: $M_x = \{m \in M | (m, x) \in \tau\}$.
  - $cond = from \cup to \cup when \cup meets$ is a set of links among measures and conditions, where:
    - $from \subseteq M_{time} \times \mathcal{C}$: set of links to *time conditions* of *from* type;
    - $to \subseteq M_{time} \times \mathcal{C}$: set of links to *time conditions* of *to* type;
    - $when \subseteq M_{count} \times \mathcal{C}$: set of links to *time conditions* of *when* type;
    - $meets \subseteq M_{state} \times \mathcal{C}$: set of links to *state conditions* of *meets* type;
  - $data \subseteq M_{data} \times \mathcal{D} \times \mathcal{S}_{\mathcal{D}} \times \mathcal{A}_{\mathcal{D}}$ is the set of links to *data* conditions;
  - $cyclic \subseteq M_{time} \times (\{linear\} \cup \mathcal{F}_{agg})$ is the specification of whether the time measure is linear or cyclic, in which case the aggregation function is specified;
  - $cal \subseteq M_{time} \times \mathcal{B}$ is the specification of the schedule of the business hours used to compute time measures. For instance, a schedule of 9 to 5 means that only the time between these two hours is computed in the time measure (e.g., each day computes only as 8 hours instead of 24 hours.)
  - $agg \subseteq M_{agg} \times (M_{time} \cup M_{count} \cup M_{state} \cup M_{data} \cup M_{der}) \times \mathcal{F}_{agg}$ is a set that maps each aggregated measure with the measure that it aggregates and the aggregation function used;
  - $uses \subseteq M_{der} \times (\mathbb{N} \to M) \times F$, where $(\mathbb{N} \to M)$ is a sequence $\langle md_n \rangle$ of the elements of M involved in the definition of the derived measure, and F is the set of all possible functions that could be applied on any sequence of measures in M.

PPI *RS.3.51* shown in Fig. 3 has variability according to *DimC-2.M1* since the element of the process where it is defined (task) changes depending on the variant. Taking as an example this PPI, and assuming that it is calculated as the average of the instances ($\mathcal{F}_{agg}$) occurring during one month (scope), with a target value of "less than or equal to 4 hours", Fig. 5 shows the relationship between the modeling of the PPI using Visual PPINOT and the proposed formal definition.

To make the definition of properties over a performance model easier, we need to define several auxiliary functions. Given a connector link $lm \in L_M$, $\Pi_M(lm)$ represents the measure involved in lm and $type_M(lm) \in T_M$, where $T_M \in \{\tau, from, to, when, meets, cyclic, data, agg, uses\}$ represents the type of the link. For instance, let $lm = (m_1, c_1) \in from$, $\Pi_M(lm) = m_1$ and $type_M(lm) = from$. Similarly, given a connector link $lp \in L_P$, $\Pi_P(lp)$ represents the PPI where the attribute has been assigned and $type_P(lp) \in T_P \in \{sco, tar, res, inf, mes\}$ represents the type of the link.

We also define $p \xrightarrow{t}$ as the subset of L_P whose PPI is p and whose type is t, i.e., $p \xrightarrow{t} = \{lp \in L_P \,|\, \Pi_P(lp) = p \wedge type_P(lp) = t\}$. Likewise, $m \xrightarrow{t}$ is the subset of L_M whose

measure definition is m and type is t, i.e., $m \xrightarrow{t} = \{lm \in L_M \mid \Pi_M(lm) = m \wedge type_M(lm) = t\}$.

Finally, $m*$ is the set of all measure definitions used in the definition of m. If m is a base measure, then $m* = m$, but if m is an aggregated measure or derived measure, $m*$ includes m and all measure definitions m aggregates or combines. It can be defined as follows:

$$m* = \begin{cases} m & \text{if } m \in M_{time} \cup M_{count} \cup M_{state} \cup M_{data} \\ m \cup ma* & \text{if } m \in M_{agg} \wedge (m, ma, x) \in agg \\ m \cup \bigcup_{md \in \langle md_n \rangle} md* & \text{if } m \in M_{der} \wedge (m, \langle md_n \rangle, f) \in uses \end{cases}$$

Similarly, $p*$ is the set of all measure definitions used to calculate p, i.e. $p* = m*$ with $(p, m) \in mes$

Now it is possible to define a syntactically correct PPINOT performance model PM. This is based on the metamodel specification introduced in [23]. We mainly specify restrictions about relationships of measuring elements and define link constraints between PPIs and its attributes and between measures and its connectors.

Definition 3 Syntactically Correct PPINOT Performance Model: Let $PM = (P, M, L_P, L_M)$ be a performance model, PM is syntactically correct if it fulfills the following requirements:

1) There is at least one PPI p in the performance model $|P| > 0$.
2) Each PPI attribute can only have exactly one single value linked to the PPI: $\forall p \in P, t \in T_P(|p \xrightarrow{t}| = 1)$
3) The links between measure definitions and its attributes have a specific cardinality depending on the type of the measures:
   - $\forall m \in M(|m \xrightarrow{\tau}| = 1)$
   - $\forall tm \in M_{time}(|tm \xrightarrow{from}| = 1 \wedge |tm \xrightarrow{to}| = 1 \wedge |tm \xrightarrow{cyclic}| = 1 \wedge |tm \xrightarrow{cal}| = 1)$
   - $\forall cm \in M_{count}(|cm \xrightarrow{when}| = 1)$
   - $\forall sm \in M_{state}(|sm \xrightarrow{meets}| = 1)$
   - $\forall dm \in M_{data}(|dm \xrightarrow{data}| = 1)$
   - $\forall am \in M_{agg}(|am \xrightarrow{agg}| = 1)$
   - $\forall dm \in M_{der}(|dm \xrightarrow{uses}| = 1)$
4) A derived measure cannot be used to compute itself: $\forall(m, \langle md_n \rangle, f) \in uses[m \notin \bigcup_{md \in \langle md_n \rangle} md*]$
5) For all $(m, \langle md_n \rangle, f) \in uses$, f must be a function defined over the Cartesian product of the set of all possible values of the sequence of measure definitions $\langle md_n \rangle$ linked to m.

Restrictions 1 to 3 are reflected in the cardinality of the relationships between the different classes represented in the PPINOT metamodel (Fig. 2).

VII. PPINOT VARIABILITY EXTENSION

In Section II-A, we discussed that there are two different approaches to manage variability, namely by restriction and by extension. In this section, we show how to extend the definition of PPINOT to manage the variability of PPIs identified in Section V using both approaches. To this end, we illustrate how these extensions can be used in conjunction with existing proposals for variability management in business processes. We selected the two most cited approaches out of the 23 proposals surveyed by La Rosa *et al.* [9], namely: C-iEPC (restriction) with 1313 citations and PROVOP (extension) with 577 citations. PROVOP also supports variability by restriction. In the following subsections we focus on these two approaches. The definitions provided in them follow a philosophy common to other variability management approaches by restriction and extension, so their adaptation to any other approach would be straightforward. However, these definitions should not be used as such with other approaches, because each one has particularities in its specification that must be taken into account.

Throughout this section, we use the example of the motivating scenario (Section III) to illustrate the definitions introduced. Due to space constraints, the example related to the IT Incident Management process of a public organization in Spain is included as part of the supplementary material provided for this article (Appendix C).

A. PPINOT-VR: EXTENSION OF PPINOT - VARIABILITY BY RESTRICTION

Variability by restriction starts with a customizable process model that contains the whole behavior of all process variants. Customization is achieved by restricting the behavior of the customizable process model. Next, we detail how the PPINOT performance model can be extended to support this variability and how it can be integrated with C-iEPC.

Like all variability by restriction approaches, C-iEPC captures multiple *variants* in a consolidated model, called *configurable integrated process model*, hereinafter *configurable process* for short. Like a C-EPC [18], a C-iEPC is defined by means of nodes (functions/activities, events, and gateways) and a specification of which of these nodes are configurable, but one can also include resources and objects assigned to activities, which can also be defined as configurable, to define a complete C-iEPC (aka $i\Gamma$). Each configurable node can be assigned a set of customization options. For instance, the customization options for a configurable activity are either enabled (`ON`) or disabled (`OFF`). Customization involves restricting the configurable elements of the configurable model by assigning one customization option to each configurable node. This assignment of values is called a configuration ($\mathcal{C}_{i\Gamma}$).

Figure 6 represents a C-iEPC model that combines three Deliver *variants* (See Fig. 3). A configuration of the model would involve assigning a customization option to each of the configurable activities and configurable gateways of the model. For instance, *PV-1* is obtained with the following configuration:$\{(C_1, SEQ_1), (C_2, SEQ_3), (C_5, SEQ_5), (n_1, ON), (n_3, ON), (n_6, ON), (n_8, ON), (n_{10}, ON)\}$.

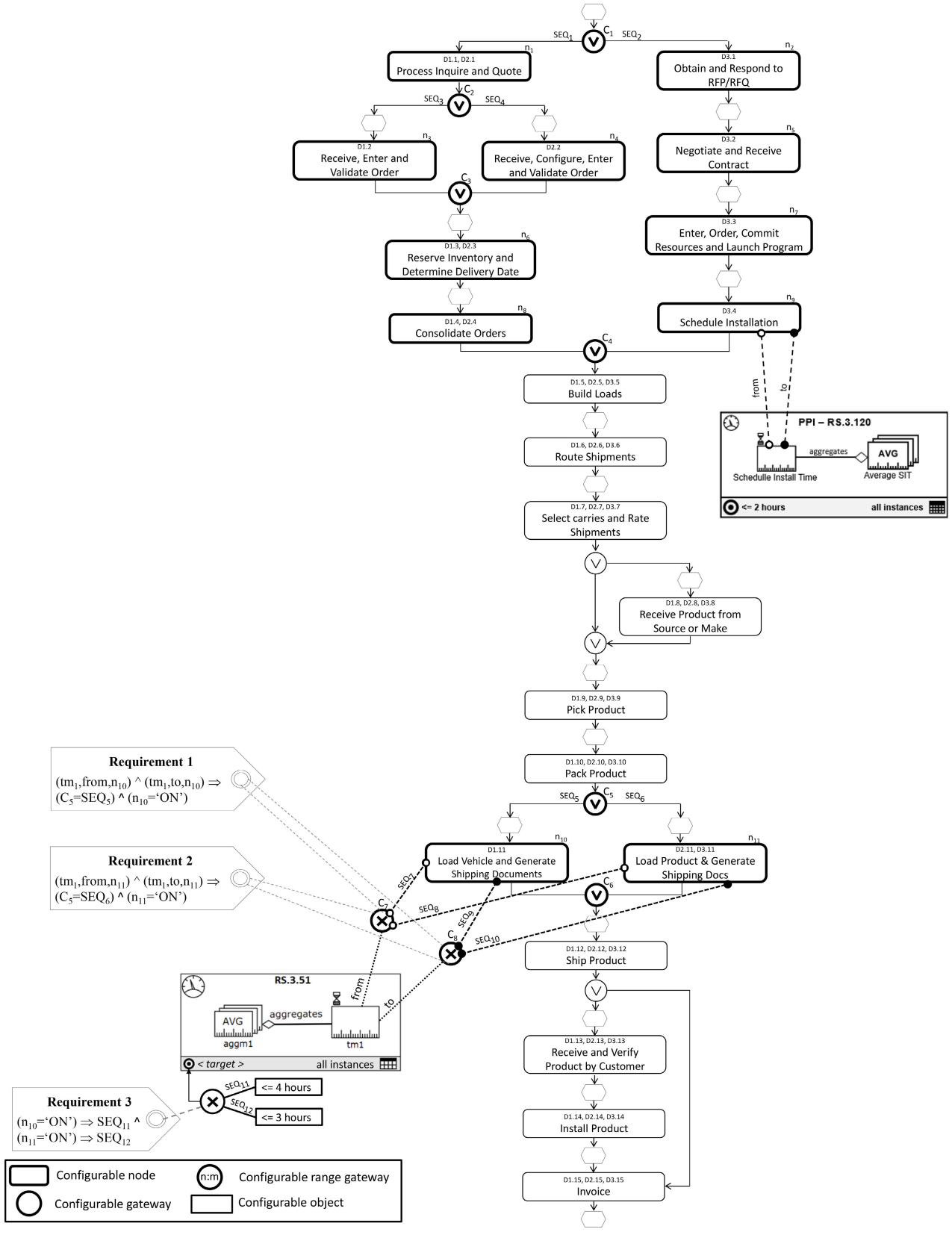


FIGURE 6. A configurable integrated process model (C-iEPC) that represents three process variants of the Deliver SCOR process with two customized PPIs.

Based on these ideas, a configurable PPINOT Performance Model can be defined as follows:

Definition 4 (Configurable PPINOT Performance Model): Let $i\Gamma$ be a C-iEPC. A configurable PPINOT performance model (*cPM*) for $i\Gamma$ is a tuple $(P, M, L_P, L_M, P^C, R^C)$, where:

- P, M, L_P and L_M are the same as in a PPINOT Performance Model *PM* applied to the process model defined by $i\Gamma$.
- $P^C \subseteq P$ is the set of PPIs that can be removed from the configurable performance model.
- R^C is a set of configuration requirements, i.e., logical expressions where the atomic statements bind the configurable elements to concrete values. The definition of configurable elements and the values they can take are described in the following paragraphs.

The configurable elements of a *cPM* are defined both explicitly and implicitly depending on the type of element. The set of configurable PPIs i.e. those that can be removed from the performance model, are explicitly specified in P^C and their customization options are either `ON` (included) or `OFF` (removed) like configurable activities. The set of configurable PPI links are implicitly specified from those PPI links for which several values are defined. Since only one value for each attribute is allowed, a *cPM* that includes more than one possible value for each attribute is implicitly defining that attribute as a configurable element, whose customization options are each of the values defined in the *cPM*.

For instance, a *cPM* could define two possible values for attribute *target* of PPI *RS.3.51*, namely "$\leq$ 4 hours" for PV-1 and "$\leq$ 3 hours" for PV-2 and PV-3. In such a model, the target of PPI *RS.3.51* is a configurable element, and its customization options would be "$\leq$ 4 hours" and "$\leq$ 3 hours." The same approach is followed by the set of configurable measure links. For instance, let us say that a time measure $tm1$ measures the time from the beginning to the end of task *'D1.11 - Load Vehicle and Generate Shipping Documentation* (n_{10})*'* or from the beginning to the end of task *'D2.11, D3.11 - Load Product and Generate Shipping Docs* (n_{11})*.*' This is modeled in the *cPM* by including two *from* and *to* attributes that represent these two options, namely $\{(tm1, (n_{10}, starts)), (tm1, (n_{11}, starts))\} \in from$ and $\{(tm1, (n_{10}, ends)), (tm1, (n_{11}, ends))\} \in to$. Figure 6 depicts these alternatives visually by means of a `configurable gateway` that links the *from* and *to* connectors with both activities. To facilitate the readability of the model and the description of our proposal, Fig. 6 includes the model to represent PPI *RS.3.51* (with variability) and PPI *RS.3.120* (without variability). The model of PPI *RS.2.3* is included in Appendix A of the supplementary material.

Therefore, in summary, the configurable elements of a *cPM* are:

- The set of PPIs that can be removed from the performance model as specified at P^C.
- The set of PPI attributes for which several values are defined: $A_P^C = \{(\pi_P(lp), type_P(lp)) | \exists lp, lp' \in L_P [lp \neq lp' \wedge \pi_P(lp) = \pi_P(lp') \wedge type_P(lp) = type_P(lp')]\}$.
- The set of measure attributes for which several values are defined: $A_M^C = \{(\pi_M(lm), type_M(lm)) | \exists lm, lm' \in L_M [lm \neq lm' \wedge \pi_M(lm) = \pi_M(lm') \wedge type_M(lm) = type_M(lm')]\}$

These configurable elements enable modeling the variability dimensions described in Section V as follows:

- P^C allows expressing *Dim-1* by providing a mechanism to specify which are the *variants* to which a PPI applies.
- A_P^C allows expressing *Dim-2* by providing a mechanism to specify which are the alternative attributes for a PPI. This includes target, scope, human resources, and measure definition, which are the attributes whose links are included in L_P
- A_M^C allows expressing *Dim-2.M1* and *Dim-2.M2* by providing a mechanism to specify which are the alternative options for the links between measure definitions and process elements (*Dim-2.M1*) or to which a certain structure of a measure definition (*Dim-2.M2*) apply. The former includes *cond* and *data* links, whereas the latter includes *cyclic*, *cal*, *agg*, and *uses* links.

A configuration of a *cPM* gives specific values to all these configurable elements by choosing one of the available options. In the case of PPIs (P^C), the options are either to include them or not in the variant. In the case of configurable attributes, their options can be defined as follows.

Definition 5 (Options of Configurable Attributes): Let $cPM = (P, M, L_P, L_M, P^C, R^C)$ be a configurable performance model, and let L_P^C (resp. L_M^C) be the set of PPI attributes (resp. measure attributes) for which several values are defined. The options $O_{(p,t)}$ of an attribute $(p, t) \in A_P^C$ (resp. $(m, t) \in A_M^C$) is the set of links defined in the *cPM* for this attribute: $O_{(p,t)} = \{l \in L_P | \pi_P(l) = p \wedge type_P(l) = t\}$ (resp. $O_{(m,t)} = \{l \in L_M | \pi_M(l) = m \wedge type_M(l) = t\}$).

Once we have these options of the configurable elements of a *cPM*, a configuration of *cPM* can be formalized as follows.

Definition 6 (Configuration): Let $cPM = (P, M, L_P, L_M, P^C, R^C)$ be a configurable performance model, a configuration is a tuple $C_{cPM} = (C_P, C_{L_P}, C_{L_M})$ such that:

- $C_P \in P^C \rightarrow \{ON, OFF\}$, which indicates which PPIs are chosen (ON) and which ones are removed (OFF).
- $C_{L_P} \in A_P^C \rightarrow L_P$, where $\forall a \in A_P^C [C_{L_P}(a) \in O_a]$, is a function that picks one PPI link for each configurable PPI attribute amongst its options.
- $C_{L_M} \in A_M^C \rightarrow L_M$, where $\forall a \in A_M^C [C_{L_M}(a) \in O_a]$, is a function that picks one measure link for each configurable measure definition attribute amongst its options.

However, not all configurations are valid. Like in *C-EPCs*, we use R^C to specify a set of requirements defined as logical expressions that binds the configurable elements to one of the concrete values defined in the configurable performance model. A typical use case for such requirements is to ensure that two PPIs ($p_1, p_2 \in P$) are always enabled together: "$p_1 = ON \Leftrightarrow p_2 = ON$". Therefore, with such a restriction,

a configuration that enables $p_1 = ON$ and disables $p2 = OFF$ is not a valid configuration.

These requirements can also be used to ensure that the links chosen in a configuration refer to process elements that are enabled. For instance, we could have the restriction "$(tm1, from) = (tm1, from, n_{10}) \implies n_{10} =' ON' \wedge C_5 = SEQ_5$" to make sure that attribute *from* of time measure $tm1$ refers to an activity (n_{10}) that is enabled in the configuration. This is visually depicted in Fig. 6 by means of Requirement 1.

Definition 7 (Valid Configuration): Let $\mathcal{C}_{i\Gamma}$ be a configuration of a C-iEPC $i\Gamma$. Let also $cPM = (P, M, L_P, L_M, P^C, R^C)$ be a configurable performance model for $i\Gamma$ and C_{cPM} a configuration of cPM. $(\mathcal{C}_{i\Gamma}, C_{cPM})$ is a valid configuration if:

- It satisfies all logical expressions in R^C.
- All chosen measure links refer to process elements that are enabled in $\mathcal{C}_{i\Gamma}$.
- At least one PPI defined in cPM is enabled, i.e., $|P \setminus P^C| + |\{p \in P^C | C_P(p) = ON\}| \geq 1$

Therefore, the notion of valid configuration includes both being consistent with the requirements and ensuring that the configuration of the performance model is consistent with the configuration of the process model.

Once we have a valid configuration of a configurable performance model, it is possible to obtain the performance model PM that is the result of applying the configuration to the configurable performance model. This can be done as follows.

Definition 8 (Performance Model of a Configuration): Let $cPM = (P, M, L_P, L_M, P^C, R^C)$ be a configurable performance model, and $C_{cPM} = (C_P, C_{L_P}, C_{L_M})$ be a valid configuration of cPM. The performance model of the configuration is a tuple $PM = (P', M', L'_P, L'_M)$, where:

- $P' = \{p \in P^C | C_P(p) = ON\} \cup P \setminus P^C$ (all chosen PPIs and all those PPIs that were not configurable)
- $M' = \{p * | p \in P'\}$ (all measures of the chosen PPIs)
- $L'_P = \{lp \in L_P | \pi_P(lp) \in P' \wedge (\exists a \in A^C_P [lp = C_{L_P}(a)] \vee (\pi_P(lp), type_P(lp)) \notin A^C_P)\}$ (all chosen PPI links and all those links that were not configurable).
- $L'_M = \{lm \in L_M | \pi_M(lm) \in M' \wedge (\exists a \in A^C_M [lm = C_{L_M}(a)] \vee (\pi_M(lm), type_M(lm)) \notin A^C_M)\}$ (all chosen measure links and all those links that were not configurable).

Finally, an interesting property to check about configurable performance models is to ensure that for any valid configuration, it obtains a syntactically correct performance model. We call this a syntactically correct configurable performance model and is defined based on Definition 3 about syntactically correct performance models as follows.

Definition 9 (Syntactically Correct Configurable Performance Model): A configurable performance model $cPM = (P, M, L_P, L_M, P^C, R^C)$ is syntactically correct if after applying any valid configuration, the resulting performance model is also syntactically correct. This is fulfilled by meeting the following conditions:

1) *(P, M, L_P, L_M) fulfills conditions* 4 *and* 5 *of a syntactically correct performance model (Definition 3) related to attribute uses of derived measures.*
2) *It provides at least one value for each PPI attribute* $\forall p \in P, t \in T_P[|p \xrightarrow{t}| \geq 1]$ *and for each measure attribute, depending on the type of the measures*:
   - $\forall m \in M(|m \xrightarrow{\tau}| = 1)$
   - $\forall tm \in M_{time}(|tm \xrightarrow{from}| \geq 1 \wedge |tm \xrightarrow{to}| \geq 1 \wedge |tm \xrightarrow{cyclic}| \geq 1 \wedge |tm \xrightarrow{cal}| \geq 1)$
   - $\forall cm \in M_{count}(|cm \xrightarrow{when}| \geq 1)$
   - $\forall sm \in M_{state}(|sm \xrightarrow{meets}| \geq 1)$
   - $\forall dm \in M_{data}(|dm \xrightarrow{data}| \geq 1)$
   - $\forall am \in M_{agg}(|am \xrightarrow{agg}| \geq 1)$
   - $\forall dm \in M_{der}(|dm \xrightarrow{uses}| \geq 1)$

It is easy to see that any valid configuration applied to a cPM with these characteristics will result in a syntactically valid performance model. On the one hand, a valid configuration guarantees that at least one PPI is selected in the configuration and that at most one value is selected for each PPI or measure definition attribute. Therefore, it meets condition 1 of Definition 3 and, partially, conditions 2 and 3 of a syntactically valid performance model because it ensures that all PPI and measure definition attributes will have at most one value. On the other hand, by definition, a cPM meets conditions 4 and 5 of a syntactically valid performance model, and it ensures the other part of conditions 2 and 3 because it ensures that all PPI and measure definition attributes will have at least one value.

B. PPINOT-VE: EXTENSION OF PPINOT - VARIABILITY BY EXTENSION

Variability by extension also starts with a model (usually called *base model*) but, in this case it represents the most common behavior or the behavior that is shared by most process variants. Customization involves extending the behavior of the process to serve a particular situation.

To illustrate how variability by extension can be used for modeling the variability of the performance model we use PROVOP [3]. In PROVOP, a `base model` (S) is built with the parts of the control-flow most common to all or most of the *variants* involved. The `base model` is marked with a set of `adjustment points`. Each `adjustment point` indicates where the base model should be modified to derive a specific `variant` (S'). Thus, customization is made by applying a `sequence of changes` ($\sigma =< op_1, op_2, \ldots, op_n >$) to the base model, which is expressed as $S[\sigma\rangle S'$. The sequence of changes σ that has to be applied is determined by a set of `options`. Each `sequence of changes` op_i, where $i = \{1, 2, \ldots, n\}$, is composed of a set of parameterized change `operations` ($op_i =< \Delta_1, \Delta_2, \ldots, \Delta_n >$). Each operation Δ can be one of these four types: *INSERT, DELETE, MOVE* and *MODIFY*. Finally, there are rules that determine which options can be applied based on the values of some context variables.

Figure 7 (a) represents a *base model* resulting from the most common elements of the three Deliver SCOR main *variants* presented in Fig. 3. PROVOP is not restricted to a specific business process modeling language. In this example, the *base model* is modeled using BPMN.[2] Figure 7 (b) depicts the `option` and `operations` that can be used to configure the *base model* to derive PV-3. In particular, `Option 1` adds 4 tasks between adjustment points a and b, adds 1 task between adjustment points d and e, and deletes the task between adjustment points c and d.

The variability in PPIs can be formalized using PROVOP by means of two elements: (1) extending the *base model* to include a base performance model and (2) defining a set of `operations` $\triangle$ for performance models so that they can be applied in a *sequence of changes* σ. The extension of the base process model to include a base performance model is straightforward because the base process model is a regular process model and hence, the PPINOT performance model described in Section VI can be directly used for this task. Like in the case of the base process model, the performance model must be syntactically correct as detailed in Definition 3. Regarding the `operations` for performance models, they can be defined based on the variability identified in Section V as follows:

- *Insert PPI*: $\triangle_{iP} = InsertPPI(PM, p, m, l_p, l_m) \mapsto PM'$. This operation adds a syntactically correct PPI p with its measure definition m, and their links l_p and l_m to the set of PPIs defined in the performance model $PM = (P, M, L_P, L_M)$. The result of applying $\triangle_{iP}$ is a new performance model that includes the new PPI: $PM' = (P \cup \{p\}, M \cup m, L_P \cup l_p, L_M \cup l_m)$.
- *Delete PPI*: $\triangle_{dP} = DeletePPI(PM, p) \mapsto PM'$. This operation removes a PPI (p) from the set of PPIs in the performance model $PM = (P, M, L_P, L_M)$. This operation also eliminates all references related to the p given, including links between the measure definition of the PPI and the business process elements. Therefore, the result of applying $\triangle_{dP}$ is a new performance model $PM' = (P \setminus p, M \setminus p*, L_P \setminus p \rightarrow, L_M \setminus m* \rightarrow)$, where $p \rightarrow = \{lp \in L_P | \Pi_P(lp) \in p\}$ and $m \rightarrow = \{lm \in L_M | \Pi_M(lm) \in m\}$ are the set of links used by p and m, respectively.
- *Modify PPI*: $\triangle_{mlP} = ModifyPPI(PM, lp) \mapsto PM'$. This operation modifies an attribute of a PPI $p = \Pi_P(lp)$ in the performance model $PM = (P, M, L_P, L_M)$ with the value provided in lp given that the attribute is not its measure definition, i.e. $type_P(lp) \neq mes$. The result of applying $\triangle_{mlP}$ is a new performance model $PM' = (P, M, (L_P \setminus p \xrightarrow{type_P(lp)}) \cup lp, L_M)$.
- *Modify PPI Measure*: $\triangle_{mlPm} = ModifyPPIMeasure(PM, lp, m, L_m) \mapsto PM'$. This operation modifies the measure that defines a PPI $p = \Pi_P(lp)$ in the performance model $PM = (P, M, L_P, L_M)$. Therefore, a precondition for the operation is that $type_P(lp) = mes$. The parameters include a syntactically correct measure and all the measures that are necessary to compute it m and all of their links Lm. The result of applying $\triangle_{mlPm}$ is a new performance model PM' in which we first remove the old elements and then add the new ones: $PM' = (P, (M \setminus p*) \cup m, (L_P \setminus p \xrightarrow{mes}) \cup lp, (L_M \setminus p* \rightarrow) \cup L_m)$
- *Move Measure*: $\triangle_{mvm} = MoveMesure(PM, lm) \mapsto PM'$. This operation moves the link between measures and process elements from one element of the process to another. It also can be used to modify the configuration of time measures. Its parameters are the performance model $PM = (P, M, L_P, L_M)$ and the new link lm such that $type_M(lm) \in \{from, to, when, meets, data, cyclic, cal\}$. The result is a new performance model $PM' = (P, M, L_P, (L_M \setminus m \xrightarrow{type_M(lm)}) \cup \{lm\})$, where $m = \Pi_M(lm)$.
- *Modify Measure*: $\triangle_{mm} = ModifyMeasure(PM, m, M_m, L_{M_m}) \mapsto PM'$. This operation modifies the measure definition of aggregated measures or derived measures. Its parameters are the performance model $PM = (P, M, L_P, L_M)$, the aggregated or derived measure that is being modified $m \in M$, the new set of measures M_m that will be used by m, and the set of links L_{M_m} of the measures in M_m and between m and M_m. The result of the operation is a new performance model PM' in which we first remove the old elements and then add the new ones: $PM' = (P, M \setminus m*, L_P, (L_M \setminus m* \rightarrow \setminus m \rightarrow) \cup L_{M_m}$. Therefore, this operation changes the whole definition of derived measures and not only a part of them.

An example of the application of these operations is depicted in Fig. 7. In `operation 1`, shown in Fig. 7 (b), where the process block between a and b is inserted, the `INSERT operation` is implicitly included for PPI *RS.3.120*, although for ease of notation, the PPI is directly connected to the block of tasks to be inserted. The *base model* includes the PPI *RS.3.51* (Fig. 7 (a)). The PPI appears in all three variants but in different tasks for each variant: in task *D1.11* for PV-1 and in task *D2.11, D3.11* for PV-2 and PV-3. For this reason, in addition to inserting the process model blocks, `option 1` reconnects PPI *RS.3.51* using `MOVE operation` to indicate the change of the *from* and *to* connectors from the task *D1.11* to *D2.11,D3.11*, changes the value of the target by means of the `MODIFY operation`, and deletes the unnecessary activity (*D1.11*) using `DELETE operation`. PV-3 also includes the PPI *RS.2.3* associated to 12 tasks of the process, so `option 1` also includes an `INSERT operation` to add this PPI definition indicating that it must be connected to the existing tasks, which are depicted with dashed borders.

- Operations *Insert PPI* and *Delete PPI*, express the *DimC-1* since they allow to identify the *variants* in which the PPI is defined.
- Operations *Modify PPI* and *Modify PPI Measure* address *DimC-2*, because they allow to specify the PPI attributes that can vary from one *variant* to another.

[2]https://www.omg.org/spec/BPMN/2.0/

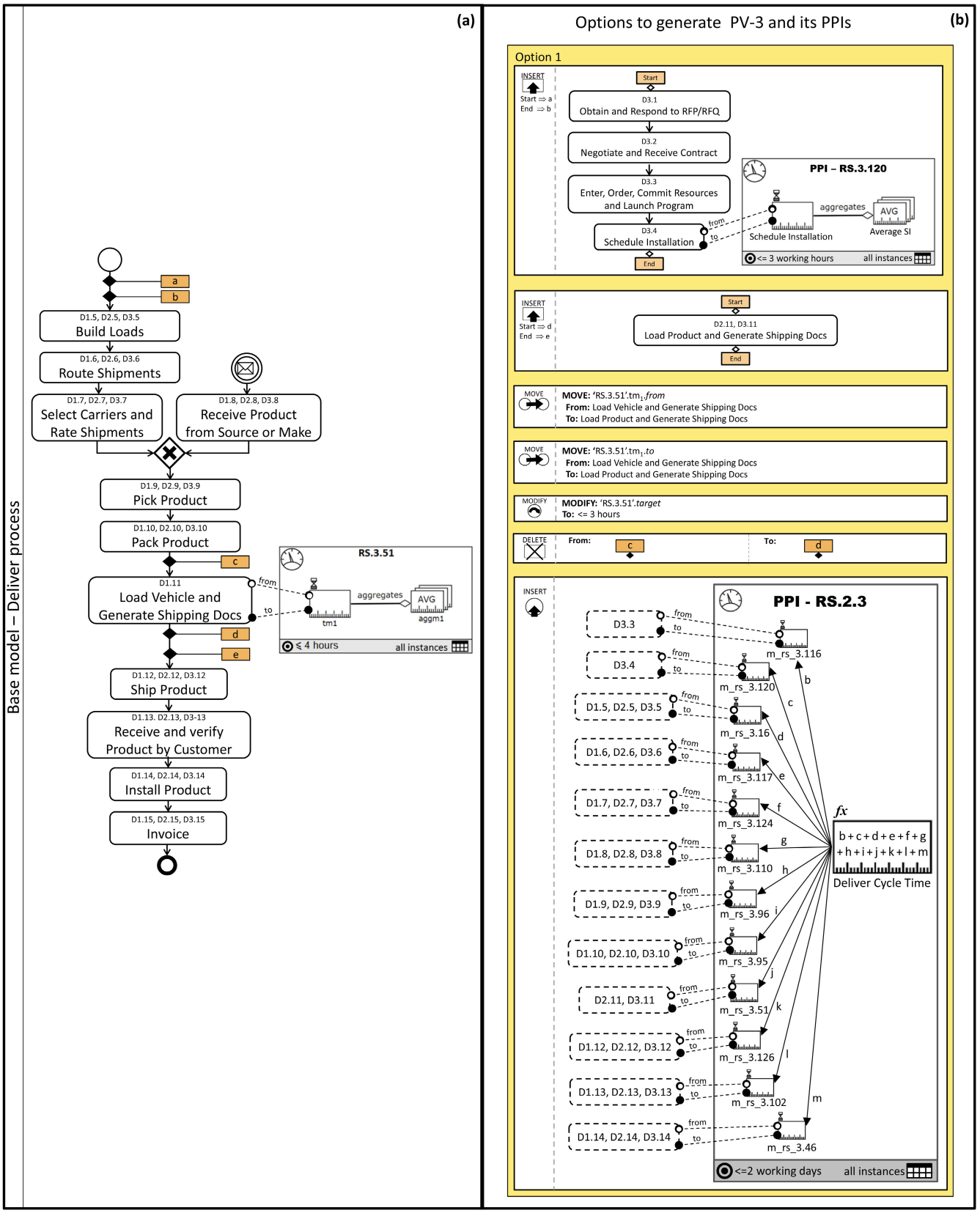


FIGURE 7. (a) Base model in PROVOP for the Deliver SCOR *variants*; (b) PROVOP *options* to derive PV-3 and its PPIs based on the performance model.

- Operation *Move Measure* allows to determine *DimC-2.M1*, since this operation specifies the variations in the process elements where the PPI is defined. In addition, operation *Modify Measure* addresses *DimC-2.M2* since it allows to modify the way in which the measure is calculated from one *variant* to another.

Due to space constraints and for the sake of readability of the manuscript, in this section we do not include the complete models for deriving the three variants with all the PPIs introduced in Fig. 3. Those models can be found in Appendix B of the supplementary material.

It is easy to see that if the base performance model is syntactically correct, then the application of these operations keep the performance model syntactically correct according to Definition 3:

- *InsertPPI* and *DeletePPI*: Adding or removing a syntactically correct PPI does not have an influence on the other PPIs, which are still syntactically correct. Therefore, the whole model remains syntactically correct.
- *ModifyPPI*: The modification of the attributes of the PPI replaces one existing value with exactly a new one and hence, does not affect the cardinality of the attribute, which means that the resulting model will still be syntactically correct.
- *ModifyPPIMeasure*: The change of measure definition of a PPI involves removing the existing one and adding a new one that is syntactically correct. This means that the PPI still has one single measure defined linked to it and, since that measure is syntactically correct, the resulting model is still syntactically correct.
- *MoveMeasure*: The change of the links with the business process replaces an existing link with a new one. Therefore, it does not affect the cardinality of the attributes and hence, the resulting model is syntactically correct.
- *ModifyMeasure*: The change of measure definitions used in aggregated a derived measures always involves removing the existing ones and adding new ones that is syntactically correct. This means that it does not affect the cardinality of the links between measures and hence, the result is still syntactically correct provided that in the case of derived measures, the conditions 4-6 of Definition 3 holds in the new links.

VIII. EVALUATION

The objective of the evaluation presented in this section was to analyze the feasibility and applicability of the dimensions of change that describe the different types of variability.

We applied our proposal to two different scenarios. On the one hand, the processes provided by the SCOR reference model and its associated measures. On the other hand, the IT Incident Management process of a Spanish public organization[3] together with its associated indicators. These two scenarios were selected, first, for being scenarios that contain variability in the performance perspective of the business process and/or in the business process control-flow perspective. Second, for belonging to different domains: the supply chain management in the first case, and IT Incident Management in the second one. And third, because of the availability of information to generate business process and PPI models.

Four phases were carried out to evaluate both scenarios.

1) *Preparation of the data to be collected.* It involved the identification and selection of the processes, measures and PPIs to be modeled and analyzed. Although the SCOR model provides six different *process families*, we selected the two *process families* with the highest similarity in the control-flow of their *variants* (Deliver and Make processes) and the set of measures associated with each one. The focus of the second scenario was the IT Incident Management process of a Spanish organization and the PPIs related to it.
2) *Collection of evidences*. To model the *variants* and the *performance models* of each of them it was necessary to take as a basis the existing documentation. On the one hand, the SCOR reference model which describes the processes and measures textually. On the other hand, the process definition documents and the spreadsheets provided by the Spanish organization in which the PPIs are described in a structured natural language format have also been used.
3) *Modeling*. Both *variants* and PPIs were modeled identifying the dimensions of change (*c.f.* Section V) and following the approaches described in Section VII. More details about this step are provided later in this section.
4) *Analysis of the data collected*. A qualitative and quantitative analysis of the data was carried out to answer the research questions presented in Section I. The analysis was done individually for each scenario.

A. SPECIFICATIONS OF SCENARIOS

Table 1 summarizes the information involved in the analysis of each scenario in terms of process families, control flow *variants*, and measures and PPIs modeled. The SCOR scenario consisted of two process different processes (Deliver and Make). A process family with three control flow *variants* was identified for each process. A total of 120 measures were analyzed within the two process families. In the scenario based on the IT Incident Management processes only one control-flow *variant* is involved. It was selected first, with the aim of identifying and analyzing variability in PPIs that does not depend on the variability in the process control-flow and because it represented a different domain from the previous scenario. A total of 18 PPIs were analyzed.

B. MODELING

As there was a relationship between the control-flow and the performance perspectives in SCOR's case, all processes and their *variants* were first modeled independently of each other

[3]No further details are revealed due to confidentiality restrictions, however, a summarized version of the process and some examples of the PPIs used for this analysis are presented in Appendix C of the supplementary material.

TABLE 1. Summary of the characteristics of the two scenarios analyzed.

Scenario	# of process families	# of control-flow variants	# of Measures /PPIs	Variability Control-flow Perspective
SCOR	2	6=3+3	120	Yes
IT I. M.	1	1	18	No
Total	3	7	138	-

using the modeling language *BPMN*. The selected process families were modeled using C-iEPC and PROVOP. The next step was to model the PPIs related to each *variant*, and then model them according to the extensions set out in Section VII following the guidelines established by the two variability modeling languages. A similar process was followed for the IT Incident Management scenario. However, unlike the previous case, in the IT Incident Management process, the variability identified lies in the performance perspective, not the control-flow. Because of this, the domain experts decided to initially model the variability of the PPIs using a template based notation. Each PPI was defined using a template in the form of a table, and within it, attributes are indicated in the rows. Moreover, for cases where required, in front of each attribute value, the condition to be applied is specified. For example, the row [target | low priority | < 10%] indicates that for that PPI, the target value should be less than 10% when the incident priority is low.

In both scenarios, several iterations were required to generate the final models.

C. ANALYSIS AND RESULTS

After modeling, all measures were classified according to the dimensions of change. Tables 2 and 3 show the classification of all measures, specifying whether they relate to DimC-1 and/or DimC-2. Of all possible combinations of the two main dimensions, six combinations were identified for the SCOR scenario and three combinations for the IT scenario. Each combination represents a row in the table, according to the following:

- DimC-1 = NO means that *measures were defined in "all" variants*.
- DimC-1 = YES means that *measures were defined only in "some" variants*.
- DimC-2 = NO means that *measures were defined "in the same way" in all the variants in which they were defined*.
- DimC-2 = YES means that *PPI attributes change from one variant to another.* DimC-2 = YES requires the specification of the changing attribute in the PPI, specifying a sub-dimension.
- DimC-2.M1 indicates the PPI measure definitions are connected to different business process elements.
- DimC-2.M2 indicates the PPI measures are calculated differently depending on the variant.
- DimC-2.T indicates that PPI target values are different from one variant to another.

In the SCOR scenario, of the 120 measures, 32 were related to *DimC-1* and 27 were related to *DimC-2*. All measures related to *DimC-2* only represented variability in *DimC-2.M* (*M1, M2*). The reason is that SCOR is a reference model and hence, it does not include information related to other attributes required to define PPIs like target value or responsible resource. This classification shows that 42.5% of measures (51 measures) have some kind of variability and 57.5% (69 measures) do not have any. When all PPIs/measures were defined over all variants (DimC-1 = NO), we found variability in the elements to which PPI measures were connected (DimC-2.M1) for 8 measures, 4 measures were calculated using different values depending on the variant (DimC-2.M2), and by 7 measures both sub-dimensions were identified *DimC-2*. When PPI/measures were defined over only some variants (DimC-1 = YES) we found two combinations: (24 measures definitions changed in structure from one variant to another (DimC-2) and 8 measures were connected to different business process elements (DimC-2.M1), but there were not measures calculated using different values depending on the variant (DimC-2.M2) or with both of them (DimC-2.M1 and DimC-2.M2). In the IT scenario, we only found variability related to changes in PPI attributes (DimC-2). Specifically, those PPIs changed in target values *(T)* (DimC-2.T). From the tables and its subsequent analysis we extracted that:

- The results reflect the high percentages of variability identified in each scenario. This justifies the need to use appropriate techniques and tools to manage variability since it would allow reducing the modeling of PPIs by 42.5% in the SCOR scenario and by 50% for the IT scenario, as well as facilitating the management and future PPI modifications.
- The metamodel is feasible to be used in different scenarios with different types of variability and types of PPIs.
- The metamodel provides enough flexibility so that, if needed, it can still be extended. That is, the PPINOT metamodel can be modified so that the structure used to define a PPI can be extended with new attributes or points of variability without making drastic changes to the version we are currently proposing.

IX. DISCUSSION

Throughout this article, we tried to answer the research questions posed in Section I related to variability in the performance perspective of business processes.

Different types of variability were identified in Section V addressing RQ1, *What types of variability should be taken into account in the performance measurement of a business processes?* They were called *dimensions of change*. After the analysis of the two scenarios, we can conclude that all of these dimensions of change can be found in real cases, which supports the idea that the performance perspective of business process is subject to variability like other perspectives. This variability sometimes is a consequence of the variability in control-flow and sometimes due to changes in the

TABLE 2. Classification of SCOR measures according to dimensions of change.

Reflects DimC-1	No. Measures DimC-1	Reflects DimC-2	No. Measures DimC-2	Sub-dimension DimC-2	Total measures involved
NO	0	NO	0	-	69
NO	0	YES	8	DimC-2.M1	8
NO	0	YES	4	DimC-2.M2	4
NO	0	YES	7	DimC-2.M1,DimC-2.M2	7
YES	8	YES	8	DimC-2.M1	8
YES	24	NO	0	-	24
Total DimC-1	32	Total DimC-2	27	**Total of all measures**	**120**

TABLE 3. Classification of measures related to the IT incident management process.

Reflects DimC-1	No. Measures DimC-1	Reflects DimC-2	No. Measures DimC-2	Sub-dimension DimC-2	Total measures involved
NO	0	NO	0	-	9
NO	0	YES	9	DimC-2.T	9
Total DimC-1	0	Total DimC-2	9	**Total of all measures**	**18**

requirements of the PPI themselves. Two main dimensions and four subdimensions were identified: Target, Scope, Human resources and Measures. Furthermore, the analysis showed that the variability of PPIs is quite common in the scenarios analyzed, affecting almost half of the PPIs defined in each scenario.

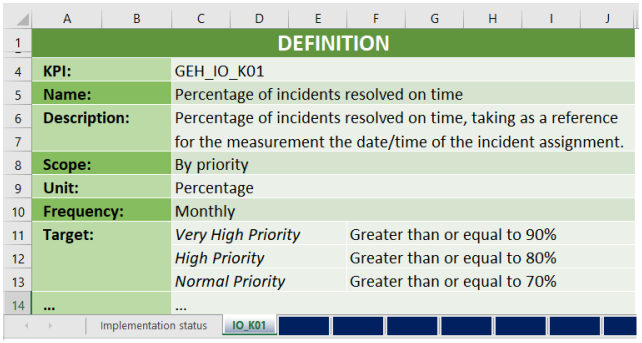
Regarding RQ2, *How can the variability of the business process performance perspective be modeled?*, we showed how to extend the PPINOT metamodel to include variability concepts and to be able to manage the dimensions of change identified in RQ1 using a two-pronged approach for the management of variability, by restriction and by extension.

To address the two variability management approaches, we proposed two alternative formalizations, one for variability by restriction (PPINOT-VR) and another one for variability by extension (PPINOT-VE). These extensions, presented in Section VII, help us specify restrictions between PPINOT elements and the requirements of a syntactically correct variable performance model that ensure that each *variant* has a syntactically correct performance model.

In our scenarios, we modeled each variant both independently of the others using PPINOT and together as a process family using PPINOT-VR and PPINOT-VE. By using this second alternative, redundant information derived from information common to all *variants* is reduced, the reuse of information is promoted, and if any change needs to be applied to a set of *variants*, the changes are applied directly to a set of *variants* and not to individual ones, thus reducing possible errors resulting from poorly updated information. This is reflected in the data presented in Table 2 and 3. In the case of SCOR, for example, out of the 120 PPIs identified, 88 do not reflect variability according to DimC-1, and of these, 19 reflect variations with respect to DimC-2. This indicates that the variations between the *variants* are few, so the information in the models is repeated and must be properly managed, for example using the proposals using PPINOT-VR and PPINOT-VE.

Finally, we addressed RQ3, *How can the existing techniques for modeling PPIs be integrated with current alternatives for variability management?*, by providing integrated solutions with two well-known notations for process variability management, namely C-iEPC and PROVOP. The solutions include the aforementioned formalizations (PPINOT-VR and PPINOT-VE) together with an adaptation of the Visual PPINOT graphical notation to define the variability based on the notations used by C-iEPC and PROVOP. In addition, the formalization can be used with any other process variability modeling language that follows either variability by restriction or by extension. This is done through straightforward modifications to the examples provided, to include the concepts specific to each language. These are slight changes, since all approaches follow the same philosophy of management by restriction or extension, as appropriate. This is possible because the concepts, operations provided and verification conditions to describe whether a model is syntactically correct are based on the same core elements, which are the PPINOT elements. Moreover, in scenarios where the variability is present only in the performance perspective, like in the IT Incident Management scenario, it is possible to use other alternatives to represent variability using, for instance, tables or templates. This enables a simpler representation of the variability for these scenarios, although it is insufficient when the variability of PPIs depend on the control-flow. Figure 8 shows a real template extract, translated into English, used by the IT Incident Management process domain experts to define their PPIs and their variability. Note that this is just an alternative notation to represent the PPINOT-VR model. The template provides several links for the same PPI attribute (e.g., target lower than or equal to 90%, 80% or 70%) and the condition requirements (R^C) that binds the configurable elements to concrete values (e.g., very high priority, high priority, and normal priority).

One reasonable question at this point is if it is really necessary to have two alternative approaches for variability by restriction and by extension while there are proposals like PROVOP that supports both of them at the same time. If we focus only on the control-flow perspective, PROVOP does fit the needs of both approaches (by restriction and by



DEFINITION			
KPI:	GEH_IO_K01		
Name:	Percentage of incidents resolved on time		
Description:	Percentage of incidents resolved on time, taking as a reference for the measurement the date/time of the incident assignment.		
Scope:	By priority		
Unit:	Percentage		
Frequency:	Monthly		
Target:	*Very High Priority*	Greater than or equal to 90%	
	High Priority	Greater than or equal to 80%	
	Normal Priority	Greater than or equal to 70%	
...	...		

Implementation status | IO_K01

FIGURE 8. Example of a template (excerpt) used to define the PPIs of the IT Incident Management scenario.

extension). However, when we talk about the performance perspective, this is no longer true if we want to keep the syntactic correctness properties. The reason for this is the different characteristics of the base model of both perspectives when variability by restriction is used. In variability by restriction, all variants are modeled in a single model and the derivation of the variants is obtained by restricting (or eliminating) certain options. In a control-flow model, this single model is syntactically correct (see [21] for example). However, in a performance model, this single model, which includes all the variants of the PPIs, is not necessarily syntactically correct. For instance, it could include a PPI with two or more ''to'' or ''from'' conditions or with more than one measure definition. In fact, it only meets conditions 4 and 5, as described in Section VII-A.

Therefore, unlike with the control-flow perspective, if we want an approach that allows both variability by restriction and by extension, we would need to allow the use of base models that are not syntactically correct (otherwise we cannot apply variability by restriction with PPIs). However, if we do so, then we will not be able to guarantee that the resulting model after applying variability by extension operations is syntactically correct.

There is a solution for this. It involves performing a multi-step configuration: first we build a syntactically correct model by restricting certain options (i.e., we apply variability by restriction), and then we apply the extension operations. However, this is the same as if we first apply PPINOT-VR and then we apply PPINOT-VE. In other words, we could create a unified approach, but it would be just the union of the two approaches described in the paper.

As a result of this work, we have also identified some limitations. We have extended the PPINOT metamodel to model the variability of PPIs together with two variability business process modeling languages, one for each alternative for managing variability (extension and restriction). It would be interesting to adapt these extensions to other variability business process modeling languages. To do so, PPINOT-VR and PPINOT-VE can be regarded as examples of how a variability management proposal can be extended to support variability in PPIs based on the dimensions identified as part of RQ1. The ultimate goal would be to analyze similarities and/or differences between the different approaches. In this way, it could be concluded whether the application of the extension is entirely dependent on the variability modeling language or only depends on the alternative, out of the two possible ones, for managing variability. In addition, although two different scenarios were used for the analysis, it would also be valuable to apply this proposal to other scenarios that require a higher volume of variability both in the process model and in the definition of PPIs, to evaluate the scalability and ease of adaptation.

X. CONCLUSION AND FUTURE WORK

In this paper, we can conclude that the performance perspective of business processes, which is composed of a set of PPIs, is subject to variability like other process perspectives, and that their variability can and should be managed together, applying either variability by restriction or by extension.

Our extensions of the PPINOT metamodel address this need to develop techniques and tools to facilitate the design, analysis, and automated management of variability related to PPIs. In addition, it ensures syntactically correct PPI variant definitions and management, thus reducing errors in the performance measurement. As far as we know, there is not a similar proposal in the literature.

By using the PPINOT metamodel for modeling variability related to PPIs our proposal inherits features from PPINOT. For example, given a PPI, it is possible to trace the process elements used for its definition, extract information from these relationships, and provides the possibility to define information extraction tools for the automatic calculation of PPIs [23]. In scenarios without variability, PPINOT has been successfully used in real cases [23], [42], [48], similarly, our proposal could be of great help to stakeholders, both to facilitate the task of modeling, defining and analyzing PPIs in variability contexts, and to facilitate the calculation of PPIs and the analysis of results.

Our proposal of varibility modeling is only a first step, but more work can be performed in several directions. First, on the basis of the different PPINOT notations, templates and linguistic patterns, as introduced in [48], can also be extended to include variability in the definition of the performance model in a process family. The two extensions (by restriction and by extension) can also be evaluated to understand the conditions in which each of them work best. Second, tooling support is another line of future work to facilitate the modeling of performance models on process families and to enable comparison between PPIs of different variants. Finally, we also want to explore if the information of the variants of the process can be exploited as contextual information to improve the quality of the predictions of PPIs with variability [51].

REFERENCES

[1] A. Hallerbach, T. Bauer, and M. Reichert, ''Guaranteeing soundness of configurable process variants in Provop,'' in *Proc. IEEE Conf. Commerce Enterprise Comput.*, Jul. 2009, pp. 98–105.

[2] F. Milani, M. Dumas, N. Ahmed, and R. Matulevičius, "Modelling families of business process variants: A decomposition driven method," *Inf. Syst.*, vol. 56, pp. 55–72, Mar. 2016.
[3] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: The Provop approach," *J. Softw. Maintenance Evol.*, vol. 22, nos. 6–7, pp. 519–546, 2010.
[4] J. van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Proc. Work. IEEE/IFIP Conf. Softw. Archit.*, Aug. 2001, pp. 45–54.
[5] M. Sinnema and S. Deelstra, "Classifying variability modeling techniques," *Inf. Softw. Technol.*, vol. 49, no. 7, pp. 717–739, 2007.
[6] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wąsowski, "Cool features and tough decisions: A comparison of variability modeling approaches," in *Proc. 6th Int. Workshop Variability Model. Softw.-Intensive Syst. (VaMoS)*, 2012, pp. 173–182.
[7] F. Milani, M. Dumas, and R. Matulevicius, "Identifying and classifying variations in business processes," in *Proc. BMMDS/EMMSAD CAiSE*, vol. 113, 2012, pp. 136–150.
[8] R. Cognini, F. Corradini, A. Polini, and B. Re, "Extending feature models to express variability in business process models," in *Proc. CAiSE Workshop*, vol. 215, 2015, pp. 245–256.
[9] M. L. Rosa, W. M. P. V. D. Aalst, M. Dumas, and F. P. Milani, "Business process variability modeling: A survey," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 2:1–2:45, 2017.
[10] A. Yousfi, R. Saidi, and A. K. Dey, "Variability patterns for business processes in BPMN," *Inf. Syst. e-Business Manage.*, vol. 14, no. 3, pp. 443–467, Aug. 2016.
[11] C. Rolland and S. Nurcan, "Business process lines to deal with the variability," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci.*, 2010, pp. 1–10.
[12] M. Reichert, A. Hallerbach, and T. Bauer, "Lifecycle management of business process variants," in *Handbook Business Process Management 1*. Berlin, Germany: Springer, 2015, pp. 251–278.
[13] M. Aiello, P. Bulanov, and H. Groefsema, "Requirements and tools for variability management," in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf. Workshops (COMPSACW)*, Jul. 2010, pp. 245–250.
[14] I. Machado, R. Bonifácio, V. Alves, L. Turnes, and G. Machado, "Managing variability in business processes: An aspect-oriented approach," in *Proc. Int. workshop Early Aspects (EA)*, 2011, pp. 25–30.
[15] A. Hallerbach, T. Bauer, and M. Reichert, "Configuration and management of process variants," in *Handbook on Business Process Management 1*. Berlin, Germany: Springer, 2010, pp. 237–255.
[16] M. Moon, M. Hong, and K. Yeom, "Two-level variability analysis for business process with reusability and extensibility," in *Proc. 32nd Annu. IEEE Int. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2008, pp. 263–270.
[17] M. Razavian and R. Khosravi, "Modeling variability in business process models using UML," in *Proc. 5th Int. Conf. Inf. Technol., New Generat. (ITNG)*, Apr. 2008, pp. 82–87.
[18] M. Rosemann and W. M. P. van der Aalst, "A configurable reference modelling language," *Inf. Syst.*, vol. 32, no. 1, pp. 1–23, 2007.
[19] A. Schnieders, "Variability mechanism centric process family architectures," in *Proc. 13th Annu. IEEE Int. Symp. Workshop Eng. Comput.-Based Syst. (ECBS)*, 2006, pp. 289–298.
[20] M. Acher, P. Collet, P. Lahire, and R. B. France, "Managing variability in workflow with feature model composition operators," in *Proc. SOCO*, 2010, pp. 17–33.
[21] M. La Rosa, M. Dumas, A. H. M. T. Hofstede, and J. Mendling, "Configurable multi-perspective business process models," *Inf. Syst.*, vol. 36, no. 2, pp. 313–340, Apr. 2011.
[22] A. Lodhi, V. Köppen, S. Wind, G. Saake, and K. Turowski, "Business process modeling language for performance evaluation," in *Proc. 47th Hawaii Int. Conf. Syst. Sci.*, Jan. 2014, pp. 3768–3777.
[23] A. del-Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés, "On the definition and design-time analysis of process performance indicators," *Inf. Syst.*, vol. 38, no. 4, pp. 470–490, Jun. 2013.
[24] B. Estrada-Torres, P. H. P. Richetti, A. Del-Río-Ortega, F. A. Baião, M. Resinas, F. M. Santoro, and A. Ruiz-Cortés, "Measuring performance in knowledge-intensive processes," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–26, Mar. 2019.
[25] B. Estrada-Torres, A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés, "Identifying variability in process performance indicators," in *Proc. BPM Forum*, 2016, pp. 91–107.
[26] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manage. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, 2007.
[27] M. Becker and R. A. Laue, "A comparative survey of business process similarity measures," *Comput. Ind.*, vol. 63, no. 2, pp. 148–167, Feb. 2012.
[28] C. Liu, Q. Zeng, L. Cheng, H. Duan, and J. Cheng, "Measuring similarity for data-aware business processes," *IEEE Trans. Autom. Sci. Eng.*, early access, Jan. 26, 2021, doi: 10.1109/TASE.2021.3049772.
[29] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira, "A systematic mapping study of software product lines testing," *Inf. Softw. Technol.*, vol. 53, no. 5, pp. 407–423, May 2011,
[30] I. Reinhartz-Berger and A. Sturm, "Enhancing UML models: A domain analysis approach," *J. Database Manag.*, vol. 19, no. 1, pp. 74–94, 2008.
[31] I. Reinhartz-Berger, P. Soffer, and A. Sturm, "Organisational reference models: Supporting an adequate design of local business processes," *Int. J. Bus. Process. Integr. Manag.*, vol. 4, no. 2, pp. 134–149, 2009.
[32] M. Terenciani, D. Paiva, G. Landre, and M. I. Cagnin, "BPMN*—A notation for representation of variability in business process towards supporting business process line modeling," in *Proc. 27th Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2015, pp. 227–230.
[33] A. Delgado and D. Calegari, "BPMN 2.0 based modeling and customization of variants in business process families," in *Proc. 43rd Latin Amer. Comput. Conf. (CLEI)*, Sep. 2017, pp. 1–9.
[34] O. Haugen, B. Møller-Pedersen, J. Oldevik, G. K. Olsen, and A. Svendsen, "Adding standardized variability to domain specific languages," in *Proc. 12th Int. Softw. Product Line Conf.*, Sep. 2008, pp. 139–148.
[35] C. Ayora, V. Torres, V. Pelechano, and G. H. Alférez, "Applying CVL to business process variability management," in *Proc. VARiability You Workshop Variability Modeling Made Useful Everyone (VARY)*, 2012, pp. 26–31.
[36] D. Calegari, A. Delgado, and L. Peña, "Model-driven support for business process families with the Common Variability Language (CVL)," *CLEI Electron. J.*, vol. 23, no. 1, pp. 1–24, Apr. 2020.
[37] F. Chan, "Performance measurement in a supply chain," *Int. J. Adv. Manuf. Technol.*, vol. 21, no. 7, pp. 534–548, May 2003.
[38] J.-P. Friedenstab, C. Janiesch, M. Matzner, and O. Muller, "Extending BPMN for business activity monitoring," in *Proc. 45th Hawaii Int. Conf. Syst. Sci. (HICSS)*, Jan. 2012, pp. 4158–4167.
[39] B. Korherr and B. List, "Extending the EPC and the BPMN with business process goals and performance measures," in *Proc. ICEIS*, 2007, pp. 287–294.
[40] C. Pedrinaci, D. Lambert, B. Wetzstein, T. van Lessen, L. Cekov, and M. Dimitrov, "SENTINEL: A semantic business process monitoring tool," in *Proc. 1st Int. Workshop Ontol.-Supported Bus. Intell. (OBI)*, 2008, pp. 1:1–1:12.
[41] V. Popova and A. Sharpanskykh, "Modeling organizational performance indicators," *Inf. Syst.*, vol. 35, no. 4, pp. 505–527, Jun. 2010.
[42] A. del Río-Ortega, M. Resinas, A. Durán, B. Bernárdez, A. Ruiz-Cortés, and M. Toro, "VISUAL PPINOT: A graphical notation for process performance indicators," *Bus. Inf. Syst. Eng.*, vol. 61, pp. 137–161, Apr. 2019.
[43] J. Saldivar, C. Vairetti, C. Rodríguez, F. Daniel, F. Casati, and R. Alarcón, "Analysis and improvement of business process models using spreadsheets," *Inf. Syst.*, vol. 57, pp. 1–19, Apr. 2016.
[44] D. Diaz, M. C. Cornax, A. Front, C. Labbé, and D. Faure, "Modélisation de la variabilité des indicateurs dans le cadre des administrations de services publics," in *Proc. INFORSID*, 2019, pp. 1–6.
[45] D. Diego, "Integrating PPI variability in the context of customizable processes by extending the business process feature model," in *Proc. IEEE 24th Int. Enterprise Distrib. Object Comput. Workshop (EDOCW)*, Oct. 2020, pp. 80–85.
[46] Meiliana, M. Vianden, and H. Lichter, "Variability model towards a metric specification process," in *Proc. Int. Conf. Comput. Sci. Inf. Tech.*, 2011, pp. 76–79.
[47] D. Suhartono, "Variability model implementation on key performance indicator application," *Int. J. Innov., Manage. Technol.*, vol. 6, no. 1, pp. 77–80, 2015.
[48] A. del-Río-Ortega, M. Resinas, A. Durán, and A. Ruiz-Cortés, "Using templates and linguistic patterns to define process performance indicators," *Enterprise Inf. Syst.*, vol. 10, no. 2, pp. 159–192, Feb. 2016.
[49] S. C. C. APICS, *Supply Chain Operations Reference Model: SCOR Version 11.0*. USA: Supply Chain Council, APICS, CCOR, CPIM, CSCP, DCOR, SCOR, and SCORmark are all registered trademarks of APICS, 2015.

[50] R. G. Poluha, *Application of the SCOR Model in Supply Chain Management*. Youngstown, OH, USA: Cambria Press, 2007.
[51] A. E. Marquez-Chamorro, K. Revoredo, M. Resínas, A. Del-Rio-Ortega, F. M. Santoro, and A. Ruiz-Cortés, "Context-aware process performance indicator prediction," *IEEE Access*, vol. 8, pp. 222050–222063, 2020.

BEDILIA ESTRADA-TORRES received the B.Sc. degree in Computer Science from the Universidad Centroamericana José Simeón Cañas, El Salvador, in 2009, and the master's degree in Software Engineering and Technology and the international Ph.D. degree (Hons.) in Software Engineering from the Universidad de Sevilla, Spain, in 2012 and 2018, respectively. She is currently a Lecturer and a Researcher with the Universidad de Sevilla, and a member of the ISA Research Group. She has collaborated and carried out research stays in Brazil and Estonia, and has participated in Spanish and European research projects. Her research interests include business process management, the analysis, modeling, and management of process performance indicators in different scenarios, such as structured processes, variability in process families, knowledge-intensive processes, and their relationship with decision-making processes.

ADELA DEL-RÍO-ORTEGA received the international Ph.D. degree (Hons.) in Software Engineering and Technology from the Universidad de Sevilla, Spain, in 2012. She is currently an Associate Professor with the Universidad de Sevilla. She took part in more than ten research and development and innovation projects, and has cooperated with several IT companies, as a consultant and researcher. She developed two registered software tools, which generated an industrial value of more than 60k €. She has contributed to more than 20 scientific publications in prestigious journals and conferences, and has running collaborations with various international scholars. Her research interests include business process management and process performance improvement. Further, research interests pertain to the modeling of SLAs, robotic process automation, knowledge-intensive processes, and the decision management.

MANUEL RESINAS received the Ph.D. degree (Hons.) in Software Engineering and Technology from the Universidad de Sevilla, Spain, in 2008. He has cooperated with several IT companies, as a consultant and a researcher. Since 2018, he has been an Associate Professor with the Universidad de Sevilla. He has been a member of the ISA Research Group, where he leads the research line on business process management, since 2010. Since 2019, he has also been leading the information systems line of the SCORE Laboratory, Universidad de Sevilla. He worked on automated negotiation of service level agreements. His current research interests include process compliance and performance analytics, predictive monitoring, collaboration systems and technologies, and human-resource management.

ANTONIO RUIZ-CORTÉS (Member, IEEE) received the B.Sc. degree (Hons.) in Computer Science and the M.Sc. and Ph.D. degrees (Hons.) in Informatics Engineering from the Universidad de Sevilla, Spain, in 1992, 1996, and 2002, respectively.

From 1990 to 1998, he was at Computer Industry. From 1996 to 1998, he was a part-time Lecturer with the Universidad de Huelva. In 1998, he joined the Universidad de Sevilla, as a full-time Lecturer. In 2004, he founded the Applied Software Engineering Group. Universidad de Sevilla. Since 2016, he has been a Full Professor of software and service engineering and heads the SCORE Laboratory, Universidad de Sevilla, since 2019. His current research interests include service-oriented computing, business process management, and testing and software product lines.

Prof. Ruiz-Cortés is elected as a member of the Academy of Europe and has also been the President of the Spanish Society on Software Engineering (SISTEDES), since 2018. He was a recipient of the Most Influential Paper of SPLC, in 2017, and VAMOS Award, in 2020. He also received the Extraordinary Award for the B.Sc., M.Sc., and Ph.D. degrees. He is an Associate Editor of *Computing* (Springer) and *International Journal of Cooperative Information Systems*.