On the Definition and Analysis of Process Performance Indicators

Adela del Río Ortega

PhD dissertation

June 2012

Supervised by Prof. Dr. Antonio Ruiz Cortés and
Prof. Dr. Manuel Resinas Arias de Reyna

University of Seville

To Rafa,
you made this happen, thanks for existing.

To Nachete,
my redheaded bundle of joy to whom this thesis has stolen so much time.

Acknowledgements

After a long time of hard work, finally it's time to look back and thank all the people who supported, challenged and push me to make this thesis possible, and to whom I am eternally grateful.

First of all, thank God, for giving me the gifts and possibilities of carrying out this thesis.

Second, I want to thank my supervisors, Antonio Ruiz and Manuel Resinas, because this thesis is the result of their encouragement, guidance and support. Thanks Antonio, for being not only my supervisor, but also my personal coach, for encouraging me in the tough times and helping me to get the best of myself. Thanks for your useful advice on both, work and life. Thanks Manolo, because you have been my guide in the research world from the beginning, when you taught me how to write research papers; but above all, thanks for teaching me that there is no mountain high enough so as not to be scaled.

My fellow researcher of the ISA group also deserve a big thanks because, in one way or another, they contributed to this thesis. In particular I am grateful to Cristina Cabanillas, for being my travelling companion all these years; to Guti, for being by my side every day and giving me support and advice; to Amador Durán and Beatriz Bernárdez, for our fruitful discussions and their feedback to my work; to Pablo Trinidad, for his willingness and assistance with the technical aspects of this thesis; and to the rest of members of the ISA group, for their companionship and great help along this time. I also want to show my gratitude to other members of the Department of Computer Languages and Systems, especially to Alejandro Fernández-Montes, for making more bearable the long hours of work with his coffee breaks.

This thesis has also fed off the insights and directions of colleagues from other universities and research centres. I must express special gratitude to Professor Mathias Weske, who welcomed me warmly in his lab during my research stay in Potsdam, and provided me extensive feedback during my thesis time.

Por último unas palabras de agradecimiento en español. Gracias a mi familia, mis padres, mis hemanos, mi abuela, etc., por estar siempre ahí. Especialmente, gracias mamá y papá, por enseñarme la cultura del esfuerzo, por confiar plenamente en mis posibilidades, por apoyarme siempre en mis decisiones, y por no escatimar en sacrificios para hacer posible esta tesis. Gracias también a mi familia política, y en especial a Lili, por cuidar de Nachete como si fuera yo. Y gracias a mis amigos, que han entendido mis ausencias en estos últimos meses. Sin su ayuda y apoyo, esta tesis sería inimaginable. Finalmente, y lo más importante, gracias a VOSOTROS que de verdad me habéis dado aliento. Gracias Nachete, mi bichito pelirrojo, por arrancarme sonrisas cuando apenas quedaban fuerzas. Y sobre todo, gracias Rafa, porque esta es "nuestra" tesis. Sin ti, sin tus ideas, sin tu apoyo, sin tus largas horas de paciente espera, sin tu amor, esto no habría sido posible.

Adela del Río Ortega

June 2012

Abstract

A key aspect in any process-oriented organisation is the evaluation of process performance for the achievement of its strategic and operational goals. Process Performance Indicators (PPIs) are a key asset to carry out this evaluation, and, therefore, having an appropriate definition of these PPIs is crucial. After a careful review of the literature related and a study of the current picture in different real organisations, we conclude that there not exists any proposal that allows to define PPIs in a way that is unambiguous and highly expressive, understandable by technical and non-technical users and traceable with the business process (BP). Furthermore, it is also increasingly important to provide these PPI definitions with support to automated analysis allowing to extract implicit information from them and their relationships with the BP. This information can assist process analysts in the definition and evolution of PPIs, as well as in the evaluation and optimization of the BPs associated. The challenge we face in this thesis is to devise a set of techniques and tools to allow such an advanced definition of PPIs and their subsequent automated analysis.

In order to face this challenge we first propose a metamodel that allows unambiguous and highly expressive PPI definitions, as far as we know, it supports PPI definitions that could not be expressed yet, i.e. PPIs not only related to time or control flow, supported by most existing approaches, but also those related to the state of BP elements and to the content or certain restriction of Data, among others. Regarding the understandability, we propose a BPMN-like graphical notation and a set of templates and linguistic patterns inspired in successful approaches from the requirements engineering field. Both representations rely on the metamodel and can be automatically mapped from one to each other. Furthermore, we provide an automatic semantic mapping from the metamodel to Description Logics, that allows the implementation of design-time analysis operations in such a way that DL reasoners' facilities can be leveraged. Finally, we have developed PPINOT tool suite, providing support for all these contributions, as well as the possibility to extract the information required to calculate PPI values from Activiti, an open source BP management platform.

Resumen

La medida del rendimiento de los procesos es un aspecto esencial en cualquier organización orientada a procesos para la consecución de sus objetivos operacionales y tácticos. Un elemento clave para llevar a cabo esta evaluación son los indicadores de rendimiento de procesos (Process Performance Indicators-PPIs), y por ello, es crucial disponer de una definición apropiada de estos PPIs. Tras una revisión pormenorizada de la literatura relacionada y el estudio de la situación actual en diversas organizaciones, concluimos que no existe ninguna propuesta que permita definir PPIs de forma no ambigua y altamente expresiva, entendible por parte de usuarios técnicos y no técnicos, y trazable con los procesos de negocio (Business Process-BP). Además, cada vez resulta más importante dotar a estas definiciones de PPIs de soporte para su análisis automático, permitiendo la extracción de información implícita sobre ellos y sobre su relación con lo elementos del BP. Esta información permite asistir a los analistas de procesos en la definición y evolución de los PPIs, así como en la evaluación y optimización de los BPs asociados. El desafío que abordamos en esta tesis es desarrollar un conjunto de técnicas y herramientas que permitan esta definición avanzada de PPIs y su consiguiente análisis automático.

Para abordar este desafío, proponemos un metamodelo que permite definiciones no ambiguas y altamente expresivas; hasta donde sabemos, permite definir PPIs que hasta ahora no se podían, soportando PPIs no sólo relacionados con el tiempo y el flujo de control, soportados por la mayoría de las propuestas existentes, sino también aquellos relacionados con el estado de los elementos del BP y el contenido o determinada restricción de los datos, entre otros. Con respecto a la comprensibilidad, proponemos una notación gráfica compatible con BPMN, y un conjunto de plantillas y patrones lingüísticos, inspirados en exitosas propuestas del ámbito de la ingeniería de requisitos. Ambas representaciones están basadas en el mencionado metamodelo y pueden transformarse automáticamente de una a la otra. Además, proporcionamos una transformación semántica del metamodelo a lógicas descriptivas (DL), que permite la implementación de un conjunto de operaciones de análisis en tiempo de diseño, de forma que podemos aprovechar las facilidades proporcionadas por los razonadores de DL. Finalmente, hemos desarrollado una herramienta, PPINOT Tool Suite, que proporciona soporte para las contribuciones introducidas anteriormente, así como la posibilidad de extraer la información necesaria para calcular los valores de los PPIs a partir de Activiti, una plataforma de gestión de BPs.

Contents

List of Figures

List of Tables

Part I

Preface

Introduction

There is nothing more difficult to take in hand, more perilous to conduct or more uncertain in its success than to take the lead in the introduction of a new order of things

Niccolo Machiavelli (1469-1527),
Italian writer and statesman, Florentine patriot, author of 'The Prince

In this dissertation, we report on our contributions to develop a set of techniques and tools to support the definition and automated design-time analysis of Process Performance Indicators. In this chapter, we first describe the topics that constitute the context of our research work in Section §1.1. Section §1.2 explains the purpose of this work. In Section §1.3 we detail the problem that the present thesis resolves. Section §1.4 introduces the goals defined for this work. In Section §1.5 we describe the approach followed in this thesis to fulfill the established goals. Section §1.6 explains the context in which the work of this thesis has been performed. Finally, in Section §1.7, we present the structure of this dissertation.

1.1 Research context

Business Process Management (BPM) intends to support *Business Processs (BPs)* using methods, techniques, and software to design, enact, control and analyse operational processes involving humans, organisations, applications, documents and other sources of information [98]. There exists a growing interest in BPs for both, academia and business. Many companies are taking this process-oriented perspective in their business, as a way of identifying which steps really create value, who is involved in the process and which is the exchanged information; ultimately, finding out how to improve, where to increase quality, reduce waste or save time [1].

To achieve this improvement of processes, it is important to evaluate the performance of business processes. In [49], Kronz establishes that "collecting and analyzing performance-related *Key Performance Indicators (KPIs)* is the first prerequisite for holistic process management and form the basis for consistent and continuous process optimization". Hence, KPIs are recognized as a key asset to carry out this evaluation [94]. Actually, nowadays, many methodologies and frameworks such as COBIT [43], ITIL [65] or the EFQM [30] excellence model, confirm this importance by including the definition of these KPIs within their recommendations as a means to evaluate the performance of the existing business processes.

KPIs can be defined as quantifiable metrics that an organisation uses to measure performance in terms of meeting its strategic and operational objectives [63]. They provide critical information to the organisation for monitoring and predicting business performance in accordance with strategic objectives [94]. KPIs can thus be defined for organisations, in order to define and measure progress towards their goals.

Furthermore, according to Kronz [49] "the basis for all process controlling is a process-oriented KPI system that links the process perspective to the essential controlling aspect of the business." We share this point of view and this is the reason why, in our work, we focus on *Process Performance Indicators (PPIs)* (or process oriented KPIs according to the previous citation). According to Chase et al. [14], PPIs are quantifiable metrics that allow to evaluate the efficiency and effectiveness of business processes. They can be measured directly by data that is generated within the process flow and are aimed at the process controlling and continuous optimization.

According to Franceschini et al. [32] and based on the conclusions drawn in our work, four critical elements for PPIs can be identified: (1) their definition, that should be unambiguous and complete; (2) understandability, PPIs should be understood and accepted by process managers and employees; (3) traceability with the business process, enabling to maintain coherence between both assets, BP models and PPIs; and (4) possibility to be automatically analysed, allowing thus not only to gather at runtime the information required to calculate PPI values, but also to infer knowledge at design-time to answer questions like *what are the business process elements related to a given PPI?*.

1.2 Motivation

The BPM lifecycle defines the various phases in support of operational business processes. According to Weske [105], this lifecycle consists of four phases related to each other. In the design and analysis phase, BPs are identified, modelled usually by means of a graphical notation, and analysed applying, among others, simulation techniques. During the configuration phase, if a dedicated software system is used to implement the defined BPs, such a system must be selected, configured, tested and deployed. Then, during the enactment phase, BPs are executed and monitored, producing valuable execution data. This information is finally used in the evaluation phase to improve BP models and their implementations.

As stated before, PPIs are a key asset to carry out the evaluation phase of this BPM lifecycle. Taking this into account, we identify the need to define a lifecycle for the management of PPIs and to integrate it into the whole BPM lifecycle for the following reasons: first of all, PPIs management needs are quite similar to the ones of BPs, also identifying for them several phases, so it makes sense to define a *PPI Management (PPIM)* lifecycle; on the other hand, though PPIs are closely related to BPs, they are not intrinsic to their definition and management, so this PPIM lifecycle must be independent; finally, the main benefit of integrating this PPIM lifecycle into the BPM lifecycle is that PPIs become firts-class citizen in process-oriented organisations, enabling thus a proper evaluation and optimisation of BPs.

We define a PPIM lifecycle also consisting in four phases, each of them integrated into the four ones of the BPM lifecycle as follows. In the design and analysis phase, PPI models must be defined and analysed (*design*). Then, during the configuration phase, the instrumentation of the BP necessary to take the measures must be carried out (*instrumentation*). During the enactment phase, the execution information is used to compute PPI values (*computation*) and monitor (visualise) them by applying *Business Activity Monitoring (BAM)* techniques. Finally, in the evaluation phase, PPI values and related information are analysed by means of business intelligence techniques (process mining, data warehousing, etc.) (*evaluation*).

Providing support to such a PPIM lifecycle is not an easy task. Process oriented organisations, in practice, either define PPIs informally in natural language, with its well-known problems of ambiguity, incompleteness, lack of traceability with the BP, impossibility of automated analysis, etc; or they define them at a very low level, too formal and/or close to the technical view, becoming thus hardly understandable to non-technical users.

1.3 Problem statement

The support of the *PPI Management (PPIM)* lifecycle is an emerging research topic. The above discussion outlines that there is still a number of problems to be considered. This work seeks to improve the support of the PPIM lifecycle. This challenge can be stated by the following three research questions:

Research question 1. How is it possible to model PPIs that provide full support to the whole PPIM lifecycle?

Research question 2. How to depict PPIs so that they are useful for every role involved in the design phase?

Research question 3. Which is the valuable information for the roles involved in the design and instrumentation phases and how can it be extracted?

These research questions are analysed and answered in the following sections.

1.4 Thesis Goals

The main goal of this thesis is to devise a set of solutions that improve both the design and the instrumentation stages of the PPIM lifecycle.

First of all, regarding **research question 1**, one of the main goals of this work is to develop a PPI model that is traceable with the business process elements, unambiguous and complete, and highly expressive, in the sense of allowing to define all of the PPIs found in the related literature and the real scenarios studied. Current PPI definition techniques provide little support for specifying PPIs that fulfills simultaneously all these requirements. In order to enable such a specification, it is required to define *what* to capture and *how* to do it.

Regarding **research question 2**, the second goal of this work is twofold. On the one hand to develop notations understandable by all the roles (technical and non-technical) involved in the PPI design phase. On the other hand, these notations must help to reduce the existing visual gap with respect to the graphical modelling view of the BP, allowing to offer a comprehensive view of both assets (PPIs and BPs), and to maintain the coherence across them. Furthermore, these notations must fulfill the requirements for PPI definitions stated above.

Regarding **research question 3**, one of the goal of this work is to define a catalogue of analysis operations of PPIs that allow to automatically extract implicit information from them and their relationships with the BP. This information can assist process analysts in the definition and evolution of PPIs, as well as in the instrumentation, evaluation and optimization of the BPs associated. Since in our research group we have previous experience with automated analysis in the areas of *Feature Models* and *Service Level Agreements (SLAs)*, we are interested in applying this knowledge to our domain.

Last but not least, an additional goal of this work is to devise a suite of tools able to support all the conceptual solutions provided to meet the above goals, and, as far as possible BPMN compliant.

1.5 Proposal Solution

Our approach to address the previous goals is highly inspired in the solutions proposed to problems of automated management of requirement models [27, 28], feature models [5] and SLAs [84]. In all these cases, the general approach was to extend existing models; to propose graphical notations, or textual ones based on templates and linguistic patterns, in order to represent such extensions; and to interpret analysis operations as queries on a formal representation (usually by means of Description Logics or Constraint Satisfaction Problems) of such models and their instances.

Specifically this thesis provides the following contributions:

A **PPI metamodel**, called PPINOT metamodel, is defined in order to provide traceability with the BP elements, and to allow the unambiguous and complete definition of all of the PPIs found in the related literature and the real scenarios studied; in fact, our model supports the definition of PPIs that, as far as we know, cannot be expressed in any other similar proposal.

Two notations are proposed in order to allow the representation of PPIs understandable by all the roles involved in their design. Concretely, we have developed a BPMN-like **graphical notation** that allows to depict PPIs over BP models; and a **template-based notation** that uses *Linguistic Patterns (L-PATTERNs)* for the definition of PPIs. Since both notations are based on PPINOT metamodel, they inherit its feature of traceability with the BP, unambiguity and completeness, and high expressiveness; while it enables the straightforward transformation from one to eachother.

A **catalogue of analysis primitives operations** is provided in order to derive, at design-time, relationships between PPIs and BP elements, and between PPIs themselves. Furthermore, a semantic mapping of PPINOT metamodel to *Description Logics (DL)* is defined in order to provide an implementation of the aforementioned analysis operations, taking advantage of DL reasoners to infer the required knowledge from PPI definitions.

A **suite of tools** (PPINOT Tool Suite) has been developed in order to support the above contributions. It consists of a graphical editor, a template editor and an analyser. It also provides the possibility to extract the information required to calculate PPI values from Activiti, an open source BP management platform, and to create reports with these values.

1.6 Thesis Context

This thesis has been developed in the context of the research group Applied Software Engineering (Ingeniería del Software Aplicada-ISA) of the Universidad de Sevilla, and it opens a new research line within the BPM area. The work that has made this thesis development possible is in the context of the following research projects and networks:

- ISABEL: Ingeniería de Sistemas Abiertos Basada en LínEas de productos. Proyecto de excelencia de la Junta de Andalucía, referenced as TIC-2533. In the context of this project I was awarded a four-year grant for the development of my PhD thesis, and it set the basis for starting the work related to the first research question (the appropriate definition of PPIs).

- S-Cube: the European Network of Excellence in Software Services and Systems, funded by the European Commission from 01.03.2008 to 29.02.2012. Thanks to our participation in this network, we identified the need to provide PPI definition with automated analysis.

- SETI: reSearching on intElligent Tools for the Internet of services. CYCIT project referenced as TIN2009-07366. In this project we investigated the use of techniques from artificial intelligence (DL reasoning) to address the implementation of the aforementioned automated analysis.

1.7 Structure of this dissertation

This dissertation is organised as follows:

Part I: Preface. It comprises this introduction chapter, in which we introduce our research context, motivate our thesis by presenting the problems addressed, establish our goals and summarise our contributions to fulfill them.

Part II: Background Information. It provides the reader with information regarding the research context in which our work has been developed. In Chapter §2, we introduce the main concepts of BPM, and the treatment these concepts have received in the literature. In Chapter §3, we delve into the *Process Performance Management (PPM)* concept and present a summary of the most relevant approaches in the context of the definition and automated analysis of PPIs.

Part III: Our Contribution. This part is the core of our dissertation and is organised in six chapters. Chapter §5 describes the problems that motivated our research work in this dissertation, analyses current solutions, and concludes that current proposals for supporting the design and instrumentation phases of the PPIM lifecycle (including automated analysis) have a number of drawbacks. In Chapter §6 we present a metamodel (PPINOT metamodel) for the definition of PPIs, that is useful along the whole PPI lifecycle. In Chapter §7 we propose a graphical notation to depict PPIs over BP models based on this metamodel. Chapter §8 provides a set of PPI-templates and L-PATTERNs, based on PPINOT metamodel, that use natural language to define PPIs, making such a definition available to non-technical users.In Chapter §9 we define a set of analysis operations to extract information at design-time from PPI definitions and present a formalisation of the PPINOT metamodel using DL that allows to implement the aforementioned analysis operations. Finally, in Chapter §10 we present PPINOT tool suite, that provides support for the previous contributions and describe the real scenarios where our approach implemented in PPINOT tool suite has been applied.

Part IV: Final Remarks. It concludes this dissertation and highlights some future research directions in Chapter §11.

Part V: Appendices. Appendix §A provides a brief introduction to DL and OWL-DL. The specification of the set of PPIs defined in our motivating scenario is presented in Appendix §B, using the different notations presented: the PPINOT metamodel (using a textual notation), our graphical notation and PPI-templates and patterns. Finally, Appendix §C shows the material used during the experiment described in Section §10.3.1.

Part II

Background Information

Business Process Management

"If you can't describe what you are doing as a process, you don't know what you are doing."

W. Edwards Deming (1900 – 1993),

American statistician and professor

BPM can be seen as both, a management principle and a suite of software technologies for the management of the lifecycle of BPs. In this chapter we introduce the main concepts of BPM. In Section §2.1 we introduce the BPM concept. We delve into BPs and their role in BPM in Section §2.2. Section §2.3 describes a BPM lifecycle as a way to provide a comprehensive view of the concepts and technologies relevant for BPM. We pay special attention to the modelling of BPs as a core activity in BPM, presenting some notations in Section §2.4. Finally, Section §2.5 summarises the chapter.

2.1 Introduction

Business Process Management (BPM) aims at offering a high level managerial perspective of organisations. It can be seen as a principle to manage businesses: A company provides to the market products or services, which are the outcome of a number of activities performed. Business processes are the key instrument to organise these activities and to improve in general their relationships [105].

BPM is gaining increasing interest from both academia and business. Many companies are taking this process-oriented perspective in their business, as a way of identifying which steps really create value, who is involved in the process and which is the exchanged information; ultimately, finding out how to improve, where to increase quality, reduce waste or save time [1].

According to van der Aalst *et.al.*, BPM can be defined as “supporting business processes using methods, techniques, and software to design, enact control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information” [98]. The goal of this chapter is to provide an overview of the major concepts of BPM, focusing on those aspects that are most interesting and useful for this dissertation.

2.2 Business Processes

Weske states in [105] that “BPM has its root in the process orientation trend of the 1990s, where a new way of organizing companies on the basis of *Business Processs (BPs)* was proposed”. Hammer *et.al.* define a business process as a collection of activities that take one or more kinds of input and create an output that is of value to the customer [41]. They do not consider any relationship or constraint between this collection of activities, but Davenport does it in [17], where he defines a business process as “a set of logically related tasks performed to achieve a defined business outcome for a particular customer or market”. He also takes into account this relationship between process activities when he defines a business process as “a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs” and continues “business process have customers (internal or external) and they cross organizational boundaries”. Based on these definitions, Weske defines a business process as “a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations” [105].

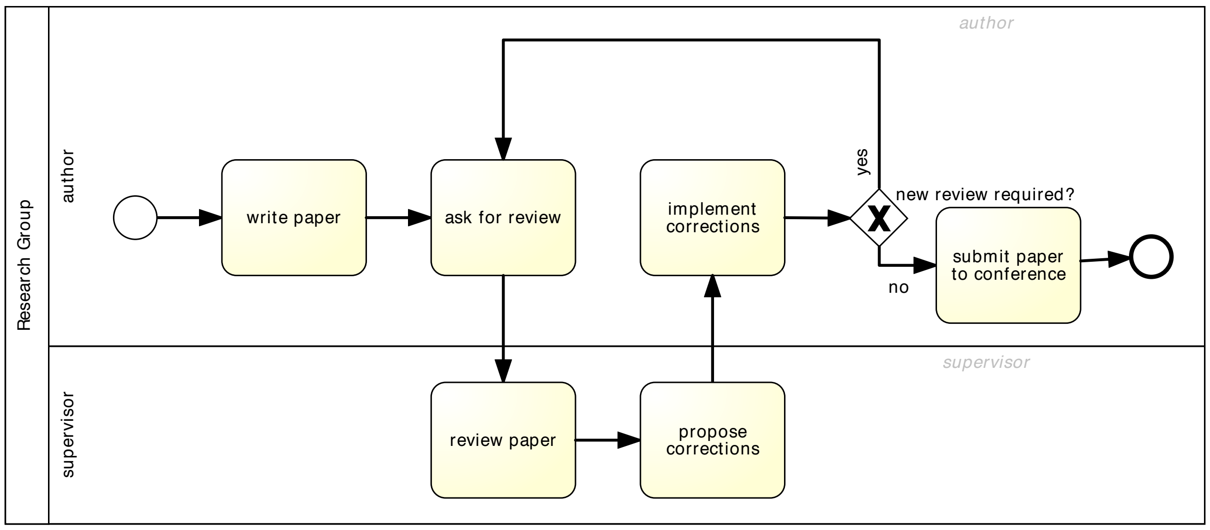


Figure 2.1: Simple business process example (process of submitting a paper to a conference)

As simple example of BP we propose the (simplified) process of submitting a paper to a conference depicted in Figure §2.1. The goal is to submit the paper with the highest possible quality (in such a way that it can be accepted). In this example we can see the set of activities that allows to realize such a goal. The coordination between these activities is defined by the ordering constraints described in the model. Though it is not reflected in the model of Figure §2.1, this BP also interacts with other BPs, for instance the one defined by the conference organization in order to receive the papers submitted and redirect them to the reviewers (not depicted in the Figure).

In this example we can identify some of the five dimensions of BPs [19, 29]. (1) The *functional dimension* describes the activities to be performed in a BP. (2) The *behavioural dimension* specifies the control flow dependencies between these activities, *e.g.* the paper must be written before it can be reviewed. (3) The *organisational dimension* focuses on the people, roles or organisational units involved, *e.g.* the author or the supervisor. (4) The *informational dimension* defines the information that must be produced or consume by activities, *i.e.* the data flow, *e.g.* the paper document is required during the review paper activity. (5) Finally the *technical dimension* makes reference to the different tools or machines that may be required in order to perform certain activities, *e.g.* the activity of submitting the paper to the conference is not feasible if no computer and internet connection (between others) are available.

As stated in [105] and [19] the basis of business process management is the explicit representation of business processes, since it helps to discover weaknesses in the current organisation of activities and serve as starting point to be analised and improved. Furthermore, "process documentation serves educational purposes–new employees entering the organization can quickly take up how things are done, or during organizational change programs, it can be shown how activities should be carried out in the new way" [19]. Nevertheless the BPM also comprises other activities as described in BPM lifecycle presented in the following section.

2.3 Business Process Management Lifecycle

In the literature there is no consensus about the number and the name of the phases in the BPM lifecycle. They vary depending on the granularity for identifying the phases and the way of grouping the functionality in the different phases [64, 97, 98, 105, 106].

In this work we will present the BPM lifecycle desribed by Weske in [105]. He proposes the four-phase BPM lifecycle depicted in Figure §2.2

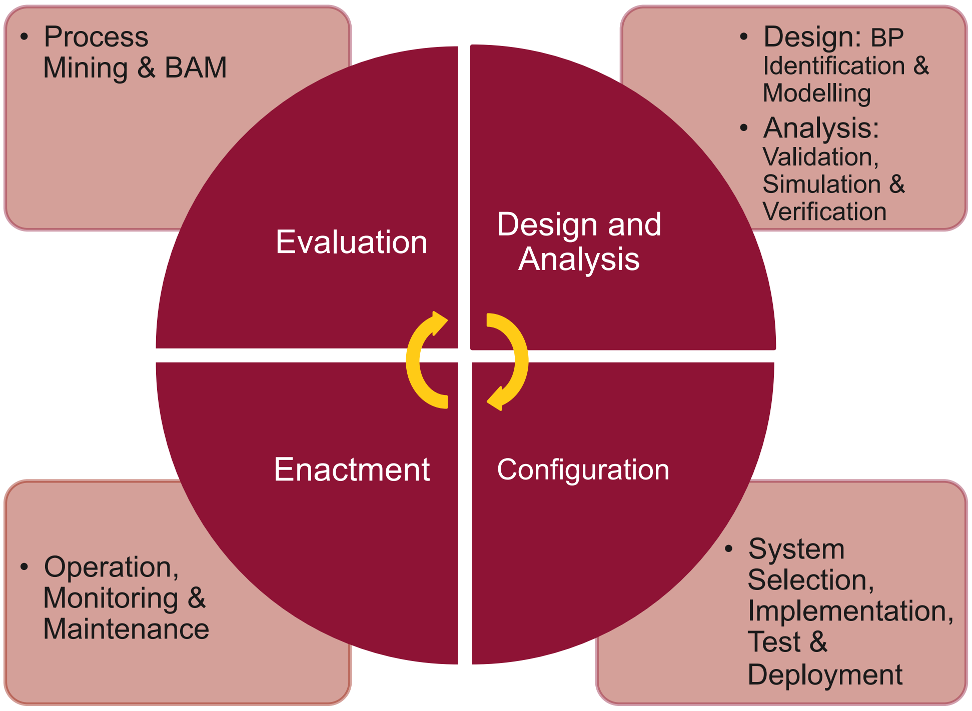

• Process Mining & BAM
• Design: BP Identification & Modelling
• Analysis: Validation, Simulation & Verification
Evaluation
Design and Analysis
Enactment
Configuration
• Operation, Monitoring & Maintenance
• System Selection, Implementation, Test & Deployment

Figure 2.2: Business process management lifecycle as described by Weske in [105]

This lifecycle starts with the *Design and analysis phase*. If no process exists, the goal of this phase is to define a new one; but if there is already an existing process, then the goal is to create an alternative for the current process. The new process need to be identified based, in the first case, on surveys on the organisational and technical environment, and in the second case, on the identified improvement possibilities. In either case, the informal business process description is translated to a particular business process modelling notation (usually a graphical one). Once a BP is defined, it need to be validated to check whether all valid process instances are reflected by its corresponding business process model. Furthermore, simulation techniques can help during the validation by allowing to detect possible undesired execution sequences and also to verify that the process actually exposes the desired behaviour. Finally, verification techniques allow to check correctness properties.

Once the business process model is designed and verified, it needs to be implemented. This is done during the *configuration phase*. It can be done in different ways. If a set of policies and procedures that the employees of the enterprise need to comply with are used to implement it, no system is required. However, if a dedicated software system is needed, it must be selected and configured in order to take into account the interactions of the employees with the system and the integration with existing software systems. This integration with existing systems may involve some implementation work, for instance to attach legacy system to the *Business Process Management System (BPMS)*. Finally this configuration must be tested, where traditional testing techniques from the software engineering area can be applied, and deployed in its target environment.

Next phase is the *enactment*, that encompasses the run time of the business process. On the one hand, a correct orchestration is necessary for the business activities to be performed according to the business process's execution constraints. On the other hand, process monitoring is an important mechanism for providing information about the status of running business process instances (BAM techniques [26] are used for this purpose). During this phase, valuable execution data is gathered. Typically execution logs are used to orderly storage information about processes such as the start or the end of activities.

Finally, the *evaluation phase* uses information collected to evaluate and improve business process models and their implementations. Techniques from the fields of business process intelligence [40], and hence, process mining [102, 103], data warehousing and classical data mining are applied in this phase ().

Note that there not exists a strict temporal ordering in which these phases need to be executed; incremental and evolutionary approaches involving concurrent activities in multiple phases are, thus, common.

2.3.1 Business Process Management System

All these phases of the BPM lifecycle must be supported by BPM products (see Figure §2.3). These software products are called BPM suites or *Business Process Management Systems (BPMSs)*. They are supposed to provide an integrated set of tools to model, simulate,

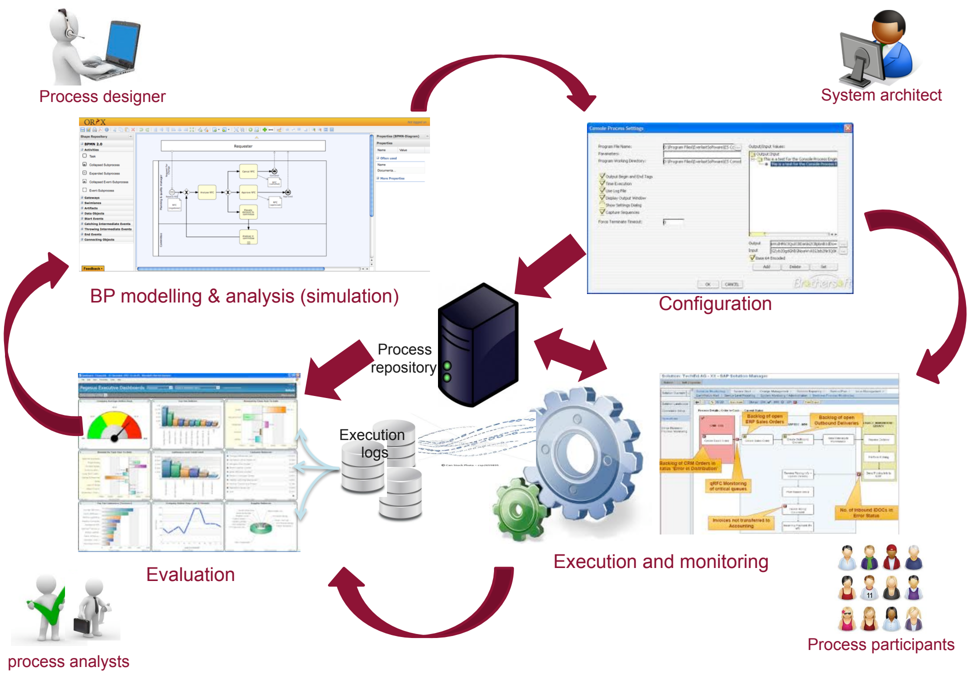


Figure 2.3: Different views and roles involved in a BPMS

deploy, enact, monitor, evaluate and continuously optimise BPs. BPMSs are compound by different systems (see Figure §2.4, taken from [69]). They coordinate tasks and synchronise data across existing systems. They also help coordinate human process activities, streamlinig tasks, triggers, and timelines related to a BP, and assuring they are completed as defined by the BP. A BPMS makes processes more efficient, compliant, agile, and visible by ensuring that every process step is explicitly defined, monitored over time, and optimised for maximum productivity [97].

2.4 Business Process Modelling

As stated before, the basis of BPM is the explicit representation of business processes. This is accomplished during the design and analysis phase of the BPM lifecycle and it is also called business process modelling. It has a long tradition and consequently several research directions can be identified. The most prominent one is the one related to graphical modelling notations. Between them we can highlight *Event-driven Process Chain (EPC)*, *Unified Modeling Language (UML)* Activity Diagrams and *Business Process Model and Notation (BPMN)*. All of them have in common that they support the specification of a process control flow, defining activities, decision points with alternative paths of execution, exception handling, event handling and additional rules and constraints [97].

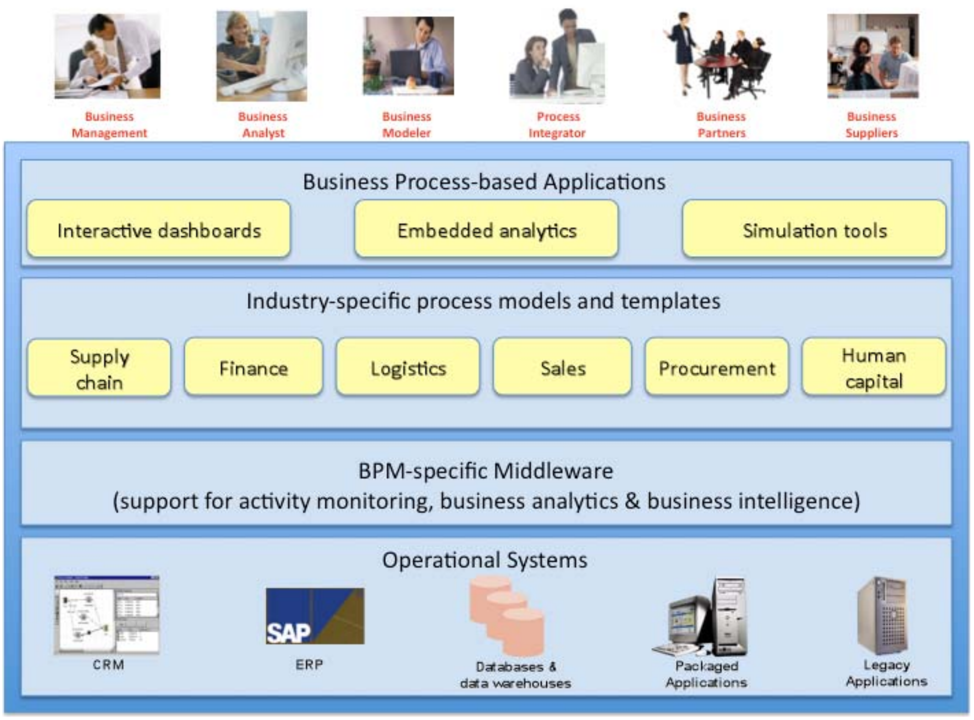


Figure 2.4: Typical parts of a BPMS (picture taken from [69])

EPCs [90] are part of a hollistic modelling approach called the ARIS (Architecture of Integrated Information Systems) framework, that defines several views similar to the dimensions previously defined for BPs. The main views are: the functional view (enterprise goals and subgoals and their relationships), the organisational view (enterprise organisational structure and its instanciation), the data view, the output view (outcome of BPs, *i.e.* products and services), and the control view, that integrates all the previous views and use EPCs to describe BPs [105]. An EPC defines the control flow of the BPs in terms of events and functions. Functions perform some business activity when they are triggered by events. Subsequently, they produce events as they carry out those activities. As such, the control flow is expressed in as a sequence of alternating events and functions [97].

UML [31, 86] is a language proposed by the OMG for the object oriented visual modelling. It is especially focused on the development of software systems. It provides several types of diagrams that can be used in conjunction with activity diagrams (class diagrams, component diagrams, sequence diagrams and use case diagrams between others). Activity diagrams allow to capture the process' control flow requirements by depicting what activities are performed, in what order and under what conditions. These activities are triggered by events as well as generate new events, which can be described using state chart diagrams.

Another BP modelling notation, now considered the *de facto* standard, is BPMN. According to its specification [68], “the primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes”. In BPMN BPs are expressed in business process diagrams. Each BP diagram consists of a set of modelling elements or constructs. They allow to model complex control flow scenarios as well as data flow. Regarding organisational aspects, BPMN provides a poor mechanism to model them (something similar to the activity diargams of UML), based on a two-level hierarchy of swimlanes, in order to organise processes in organistaion, roles, etc.; but no organisation charts or any kind of relationships can be defined. In addition to the graphical notation, a set of attributes, associated either to the complete BP or to particular elements, are used to complement BPs, but there is no graphical representation for most attributes, only some rules provided by the standard.

Apart from these modelling notations, there exist other more formally oriented modelling languages, that allow the verification of BP formal properties like correctness or completeness. One example is *simple finite state automata*, with which processes are described as devices that maintain the state of something at a certain time and can alter this state in reaction to input as well as cause an action or output as a result of a changing state. *Petri nets* [60, 100] offer another graphical technique, and are a special form of graphs constituting of places, transitions, directed arcs and tokens. Places are connected via directed arcs to transitions and vice versa. Places contain tokens, which may represent signals, events, conditions, and so on. Transitions are fired through the presence of tokens in their in-place(s). As a result the distribution of tokens is changed [97]. Petri nets present some limitations when modelling complex BPs, in these cases *Workflow nets* are used [99]. Workflow nets are an extension of petri nets with concepts and notations that ease the representation of BPs, like the possibility for tokens to carry information about the process instance they belong to. The main drawback of all these approaches is that they require expert knowledge to be used.

2.5 Summary

In this chapter we have introduced the concept of BPM. In particular we have provided definitions and an example of a BP and we have described the BPM lifecycle trying to provide an overall understanding of the main concepts and technologies of BPM. We have stopped at BP modelling, since it is considered the core of BPM; in this context we have presented some BP modelling notations, highlighting BPMN, since it is relevant for the context of this dissertation.

For a more complete and rigorous introduction to BPM we refer the reader to [97, 105]

Process Performance Management

"If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it."

H. James Harrington (1929 –),
American performance improvement guru and business man

Process Performance Management aims at measuring, monitoring and analysing the performance of BPs, in order to support decision making for BPs optimisation. In this chapter, we present the concepts of PPM that establish the context for this dissertation. In Section §3.1 we report on the importance of PPM. Then in Section §3.1 we present the historic evolution of PPM, from its corporate strategy orientation to the current process orientation. Section §3.3 describes PPIs as the basis for PPM and highlights the importance of integrate them into the whole BPM lifecycle, driving thus to the PPI management lifecycle presented in Section §3.4. Finally in Section §4.13 we summarise the chapter.

3.1 Introduction

As stated in the previous chapter (Section §2.1) BPM is used, among others, as a way of finding out how to improve BPs, *i.e.*, it assists in the BPs optimisation. To achieve this improvement, it is essential to evaluate the performance of business processes, since it helps organisations to define and measure progress towards their goals. This is the main objective of PPM. According to Heß [42], PPM aims at optimizing the process sequences at work in the company through computer-supported analysis of process structures in conjunction with KPIs, so as to organize them more effectively. Furthermore they consider PPM as the heart of a wider concept called *Corporate Performance Management (CPM)*, or also known as Business Performance Management or Enterprise Performace management. The phrase CPM was coined by Gartner Group to describe the combination of "processes, methodologies, metrics and technologies to measure, monitor and manage the performance of the business" [9]. Nowadays, CPM is used to denote all the long-term, process-oriented modes of action and approaches that have been adopted in companies for the management of BP performance , including dynamic methods such as activity-based cost calculation, the process BSc or process mining as well as static process analysis. In this chapter we provide some concepts related to this issue that are connected with the work developed in this dissertation.

3.2 Historic Evolution of PPM

Similar to other past trends in information systems and business management, PPM has evolved. Traditionally, PPM (or probably more accurate for its beginning, business performance management) focused on the strategic and tactical level (enterprise-oriented), most frequently taking into account only financial measures and using data-driven techniques. Here we can include several approaches: the *Activity-Based Costing (ASC)* [8, 62, 95], which relates resource costs to activities, products and services, giving a realistic view on the overall costs and profitability of the organisation; *Competitive Benchmarking*, which consists on a continuous process of comparing a firm's practices and performance measures with that of its most successful competitor/s.[82, 108]; or the *Balance Scorecard* of Kaplan and Norton [7, 45, 46], that provides a system to describe an organisation's overall performance using financial and non financial indicators. From the point of view of the software support for this kind of performance management, we can highlight systems or techniques such as *Decision Support Systems (DSS)*, *Executive information systems (EIS)* or *Business Intelligence (BI)* projects that used Data Warehouse and OLAP tools [34, 104]

This approach has recently changed with the evidence that “analysis must be linked more closely with the company’s value creation. Traditional Business Intelligence Systems are being replaced by process-oriented Performance Management Solutions” [42], *i.e.* now the focus is on processes and the spectrum of performance-relevant data is broader, since financial and non-financial measures (operational data) are taken into account [38, 50]. The main objective now is “to set up an overall view of the company’s core processes and to establish a cycle for continuously improving process efficiency” [49].

3.3 Process Performance Indicators

According to A. Kronz in [49] “collecting and analyzing performance-related KPIs is the first prerequisite for holistic process management and form the basis for consistent and continuous process optimization”. Later on he also claims that “the basis for all process controlling is a process-oriented KPI system that links the process perspective to the essential controlling aspect of the business. KPIs must enable conclusions to be drawn regarding the effectiveness of the processes and their efficiency”. Hence, PPIs (process oriented KPIs according to the previous citation) become first-calls citizens in BPM since they serve as the basis for the sustained success of the project [89].

Process Performance Indicators (PPIs) can be defined as quantifiable metrics that allow to evaluate the efficiency and effectiveness of business processes. They can be measured directly by data that is generated within the process flow and are aimed at the process controlling and continuous optimization [14].

Often, the terms PPIs and KPIs are used interchangeably, but, in fact, there is no consensus in the literature regarding the relationship between PPIs and KPIs. Some authors do not establish any difference between them [55, 76, 77], while others (see [22, 107] for instance) consider PPIs as a particular case of KPIs, i.e. process-related KPIs. Finally, there are others who attribute different definitions to each one, placing them at different levels, KPIs nearest to the tactical and strategical level, while PPIs nearest to the operational level [14].

Our approach in this dissertations coincides with the second one, i.e., we consider PPIs as a particular case of KPIs defined for measuring the performance of BPs. A set o KPIs are defined for the organisation as a way of measuring the level of fulfillment of its strategic and operational objectives. Furthermore, the operation of such organisation is defined by means of a set of BPs, whose performance are measured through PPIs. Taking to account this picture, depicted in Figure §3.1, every PPI is a KPI, but not vice versa, though, every KPI can become

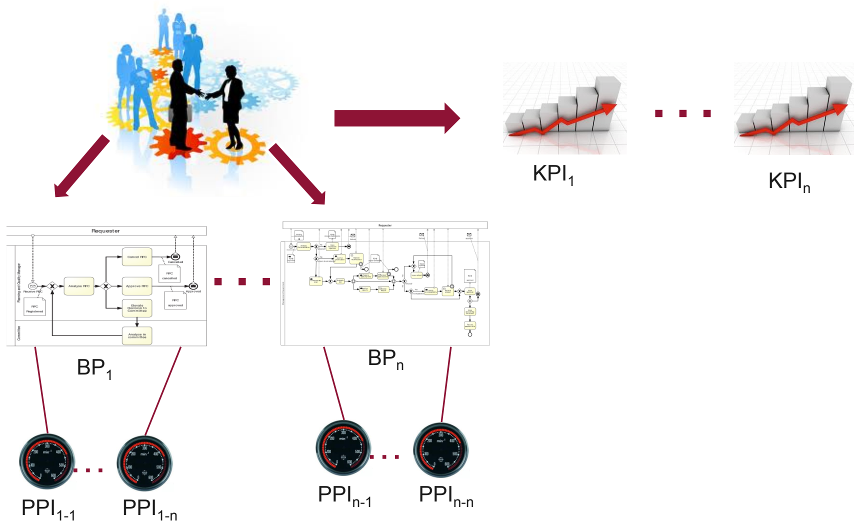


Figure 3.1: Measuring performance in an organisation (PPIs and KPIs)

a PPI if a BP is defined and used to calculate it, and it allows to improve the defined BP. Figure §3.2 illustrates this relationship PPI-KPI. There are two PPIs defined for the process example of Section §2.2, the paper submission, and two examples of KPIs for a research group, considering it as an organisation example. It can be seen that, although both KPIs are somehow related to the process of submitting a paper, they cannot be measured from its execution data.

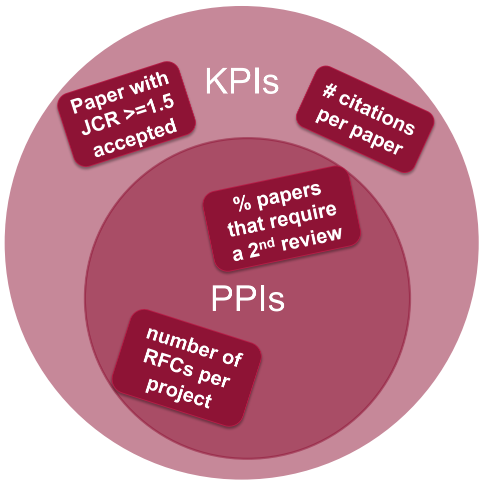

KPIs
Paper with JCR >=1.5 accepted
# citations per paper
% papers that require a 2nd review
PPIs
number of RFCs per project

Figure 3.2: PPIs vs KPIs

In order to define PPIs, it is recommended that they satisfy the SMART criteria [24, 53, 91]. SMART is a nemonic used to set objectives and KPIs/PPIs. There is no clear consensus about what the five keywords mean. In this thesis we consider the following meaning: *Specific* (it has to be clear what the PPI exactly describes), *Measurable* (it has to be possible to measure a current value and to compare it to the target one), *Achievable* (it makes no sense to pursue a goal that will never be met), *Relevant* (it must be aligned with a part of the organisation's strategy, something that really affects its performance) and *Time-bounded* (a PPI only has a meaning if it is known the time period in which it is measured).

Furthermore, as stated in [49], "it is imperative to establish a clear connection between the process and its PPIs, without which it is practically impossible to interpret the PPIs or derive any meaningful measures" [1]. In order to establish such a connection, it is convenient to integrate the management of PPIs into the whole BPM lifecycle [21, 22, 49], providing thus the aforementioned cycle for continuously improving process efficiency.

3.4 PPI Magement Lifecycle

During our research work we identified the challenges to integrate the PPI management into the BPM lifecycle, analysing related work (see [21]). For the sake of simplicity we present here a summary of such work, whose approach coincides in some aspects with the one presented in [49]. We take as starting point the BPM lifecycle presented in Section §2.3 to define a PPIM lifecycle as follows.

As stated in the introduction Chapter (§1), it is desirable to define a lifecycle for the PPI management and to integrate it into the BPM lifecycle for several reasons: first of all, PPIs management needs are quite similar to the ones of BPs, also identifying for them several phases, so it makes sense to define a PPIM lifecycle; on the other hand, though PPIs are closely related to BPs, they are not intrinsic to their definition and management, so this PPIM lifecycle must be independent; finally, the main benefit of integrating this PPIM lifecycle into the BPM lifecycle is that PPIs become firts-class citizen in process-oriented organisations, enabling thus a proper evaluation and optimisation of BPs.

In the following, we present our PPIM lifecycle, depicted in figure §3.3. It coincides in some aspects with the one presented in [49]. It is divided into four phases, each of which is integrated in one of the phases of the BPM lifecycle presented in previous chapter as follows.

In the Design and Analysis phase of the BPM lifecycle, PPIs should be modelled together with the business process. Here the definition and structure of the PPIs as well as its relationship with the BP must be described. This model of PPIs should also enable their analysis by detecting the dependencies amongst them at design time and also using them as part of the business process analysis, for instance in business process simulation techniques (Simulation attempts to

[1]we use here PPIs instead of KPIs as done in the cited work to maintain a coherence along the document regarding the terms used.

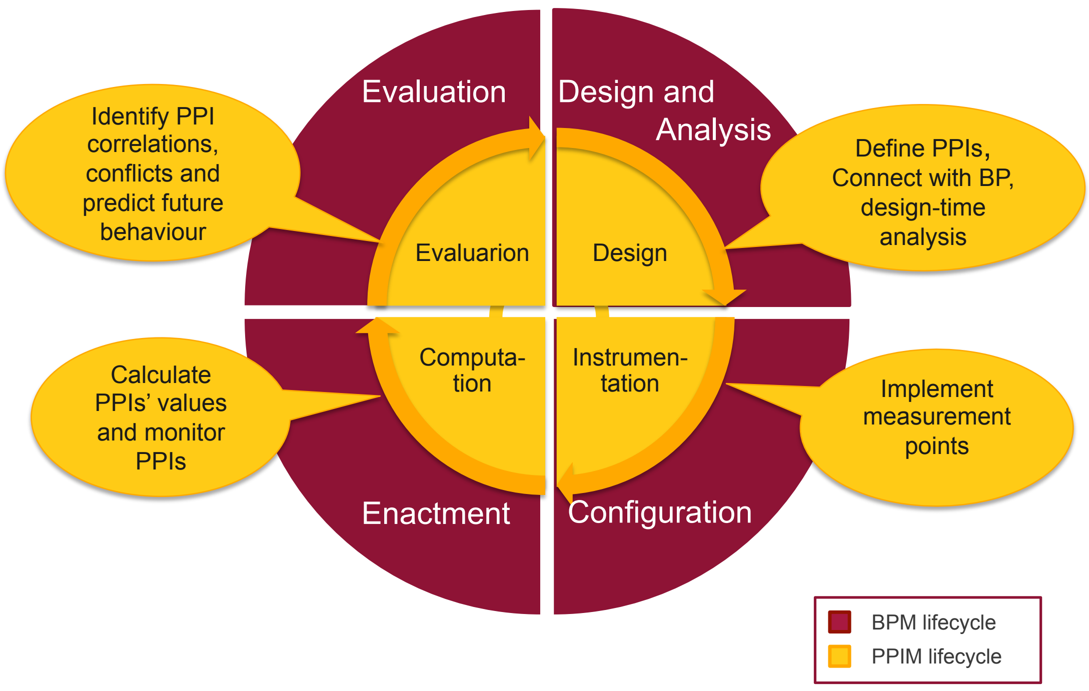

Identify PPI correlations, conflicts and predict future behaviour
Evaluation
Design and Analysis
Define PPIs, Connect with BP, design-time analysis
Evaluarion
Design
Computa- tion
Instrumen- tation
Calculate PPIs' values and monitor PPIs
Implement measurement points
Enactment
Configuration
BPM lifecycle
PPIM lifecycle

Figure 3.3: PPI management lifecycle integrated into the BPM lifecycle

"mimic" real-life or hypothetical behavior on a computer to see how processes or systems can be improved and to predict their performance under different circumstances [101]). We call this PPIM lifecycle phase *Design*.

Then, during the BPM lifecycle Configuration phase, the instrumentation of the process necessary to take the measures must be defined. This means to implement measurement points taking into account the implementation and support of the processes by *Information Technology (IT)* (for instance indicating where data input come from, *i.e.* data sources like databases of ERP systems or workflow management systems or BPMS). This is the *Instrumentation* phase.

During the Enactment phase of the BPM lifecycle, all important events related to the execution of activities (*e.g.* start/end of activities, data produced or changed by activities, etc.) are gathered and recorded in log files. Based on these events, the PPIs' values have to be calculated and the monitoring of these PPIs should be carried out. BAM techniques [26] are used for this purpose (BAM refers to near real time monitoring of business activities, measurement of PPIs, their representation in dashboards, and automatic and proactive notification in case of deviations [97]). These activities make up the *Computation* phase.

Finally, during the BPM lifecycle Evaluation phase, the information available allows to evaluate and improve business process models and their implementations. The monitoring information related to PPIs obtained in the previous phase will help to identify correlations between them and predict future behaviour. Techniques from the fields of process mining [102, 103], business process intelligence [40], data warehousing and classical data mining are applied in this phase (Business process intelligence is based on the application of business intelligence techniques, in particular data/process mining , data warehouse and OLAP tools, to BPs, allowing thus to check conformance, predict the future, recommend appropriate actions and identify improvements points). We also call this PPIM lifecycle phase *Evaluation*.

3.5 Summary

In this chapter we have highlighted the importance of *Process Performance Management (PPM)* inside the business process orientation. We have introduced the main concepts related to PPM, that establish the context for our main contributions in this dissertation. In particular we have presented its historic evolution until arrive to the current picture, where, in order to achieve a holistic process management, it is crucial to integrate PPIs and their management into the whole BPM lifecycle. Finally we have also described the PPIM lifecycle that results of such integration.

4

State of the Art

In my walks, every man I meet is my superior in some way, and in that I learn from him.

W. Somerset Maugham (1803 – 1882), American essayist, lecturer and poet

In this chapter we survey the current proposals in the context of the desing and instrumentation phases of the PPIM lifecycle. Between Section §4.2 and Section §4.11 we describe those approaches that have more relevance to our work, or that more influenced it, putting special emphasis on the way PPIs are defined, the possible PPIs that are supported and the analysis capabilities provided. Then, in Section §4.12, we comment other approaches somehow related to our work in this dissertation.

4.1 Introduction

Many works have been done in the identification and classification of key performance indicators for any company [44] and those relevant for specific domains such as logistics, production, supply chains, etc. (e.g. [7, 13, 48, 96]). Nevertheless, since in this dissertation we focus on PPIs, and more concretely on the design and analysis phases of the PPIM lifecycle, we are interested in those proposals especially related to them. In this chapter we describe the main approaches identified[1], putting especial emphasis on the way PPIs are defined in them, the possible PPIs that are supported and the analysis capabilities provided. A further analysis of these proposals as well as the level of fulfillment of a set of requirements established will be presented in Sections §5.4 and §5.5.

4.2 Castellanos et al. Approach

Castellanos et al.'s approach [12] is implemented in the IBOM platform, that allows, among other things, to define PPIs (they call them business metrics and they are not solely focused on business processes)and perform intelligent analysis on them to understand causes of undesired values and predict future values. The user can define PPIs (through a *Graphical User Interface (GUI)*) to measure characteristics of process instances, processes, resources or of the overall business operations. Specifically, they characterize PPIs through four attributes: name (unique), target entity (objet to be measured), data type (numeric, boolean, taxonomy or SLA) and desirable values (they define green, yellow and red ranges for values or categories). For the computation logic definition, templates are used. These templates map data and metadata about process executions into numeric and boolean measures. Some examples of templates given in [12] are presented below:

A: *Did process* ***P*** *end in state* ***S****?* (boolean template).

B: *Total execution time between the activation of step* ***S1*** *and the completion of step* ***S2*** (numeric template).

C: *Percentage* ***P*** *of the value of numeric output variable* ***V*** (numeric template).

D: *Was step* ***S*** *executed?* (boolean template).

New templates can be added by the user[2]. These templates includes, apart from the previous

[1]The order established between the approaches presented was defined by their date of publication.

[2]The way these new templates can be defined is not described in this paper, so it is not possible to assure the

specification part, readable by humans, an implementation part, specified in SQL, that contains the code to be executed to compute the value. It is noteworthy that this approach is not focused on business processes but on the whole organisation.

4.3 ARIS Approach

ARIS [18] models PPIs (process-oriented KPIs for them) and allow for using the Balance Scorecard approach [46] for modelling cause-and-effect relationships and assign PPIs to the strategic objectives.

Furthermore, in [88], the ARIS Process Performance Manager (PPM) is described. This tool provides a mechanism to define measurement points over BPs defined using EPCs. A measurement point defines a point in the process at which data is collected for calculating PPIs. It gathers the description of the change of status of application objects (e.g. functions, that is the name for activities in EPCs). If the system contains organisational information related to the process, it can also be gathered by measurement points. Figure §4.1 depicts an excerpt of a process example with measurement points defined, taken from [49].

Then, allocation diagrams allow to define which measurement points are used to calculate each PPI, as well as to link multiple PPIs to a new PPI; i.e they maintain calculation rules for PPIs.

This tool also allows to classify PPIs in the so-called trees, in order to group them (e.g. quality, cost, time, or process-oriented groupings). An example of a PPI tree taken from [49] is presented in Figure §4.2.

Finally, it also supports the evaluation or analysis of PPIs. This evaluation interprets PPIs with reference to various criteria -called dimensions- (time period; product; region) and describes relationships[3] between the results and the predefined target values. In addition, the monitoring of PPIs and notification in case of deviations are also supported, as well as some basic process mining techniques can be used to analyse weak points.

flexibility of this platform with respect to the PPI values it supports

[3]It is not defined which kind of relationships.

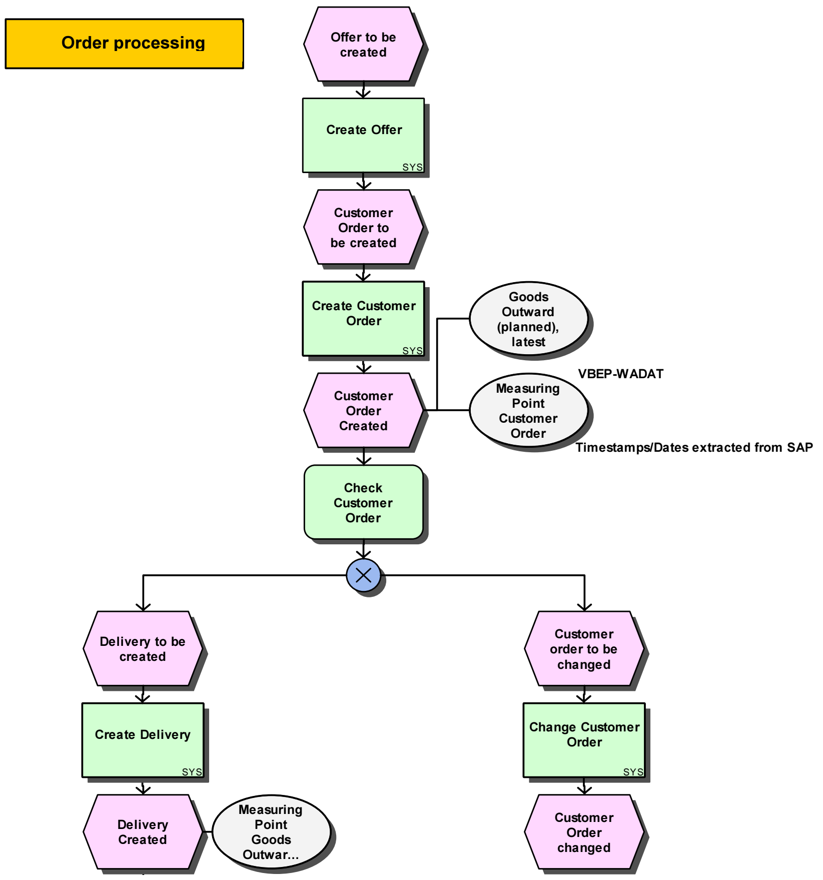


Figure 4.1: Excerpt of an EPC process model example with measurement points, taken from [49]

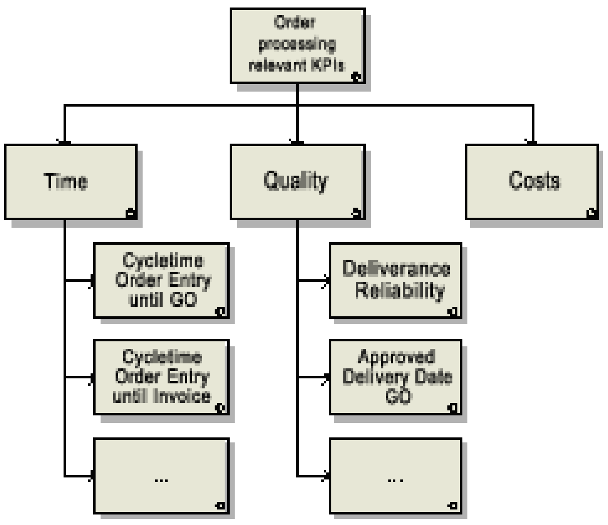

Order processing relevant KPIs
Time
Quality
Costs
Cycletime Order Entry until GO
Cycletime Order Entry until Invoice
...
Deliverance Reliability
Approved Delivery Date GO
...

Figure 4.2: PPI tree example, taken from [49]

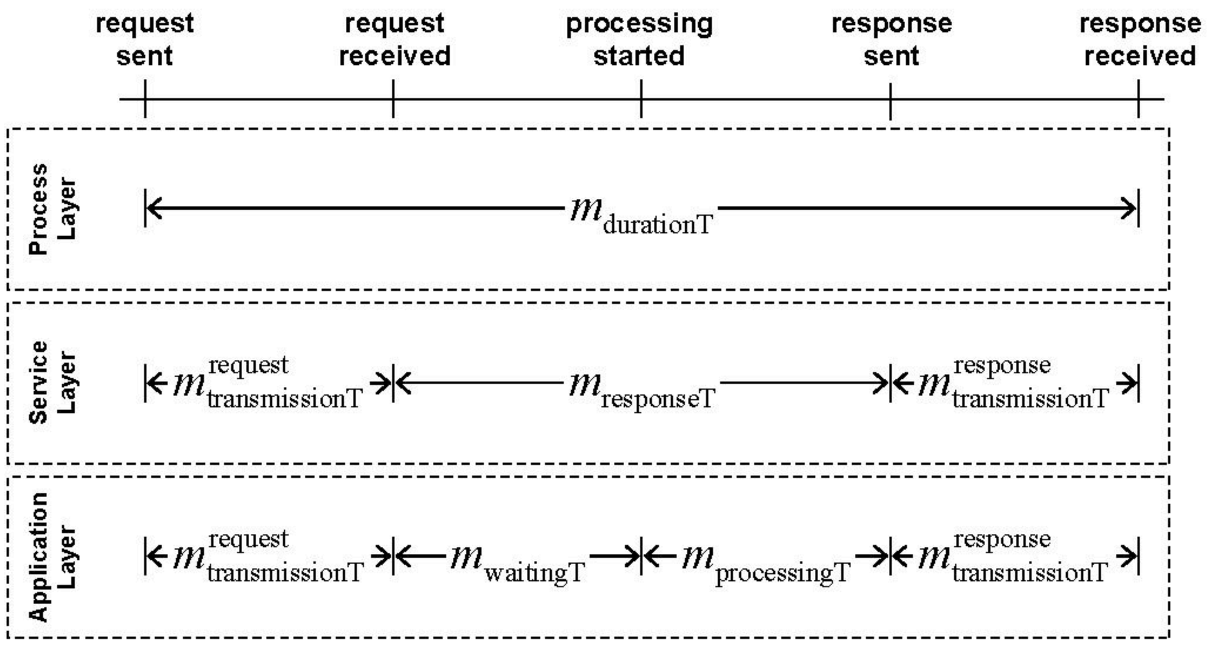


Figure 4.3: Metric dependency Pattern for Duration of Activities [51]

4.4 Mayer et al. Approach

In [51], Mayerl et al. discuss how to derive PPI (they call them metric) dependency definitions from functional dependencies by applying dependency patterns. To this end, they propose a model that distinguishes between a functional part, where they define dependencies between application, service and process layers (based on concepts of BPEL and WSDL), and another part for PPI dependencies, based on concepts of the CIM metrics model [23] and the QoS UML profile described in [67]. Concretely, they define *dimension* (measured directly from the instrumentation), *characteristic* (computed based on other PPI values), *category* (group several characteristics or other categories) and *dependency function*(is associated to a characteristic and gives instruction its value).

They also introduce a mathematical formalism in order to describe dependency functions and characteristics. An example is described in Figure §4.3 that shows a PPI mdurationT to measure the duration of a process activity and also dependencies to other PPIs.

Finally they cover the mapping of these models to a monitoring architecture that contains functions to instrument and collect PPIs, functions to aggregate and compare PPIs with agreed service levels and functions to report SLA compliance and violations. However, they do not delve into the definition of PPIs, they only set the semantics of some elements to consider when defining PPIs.

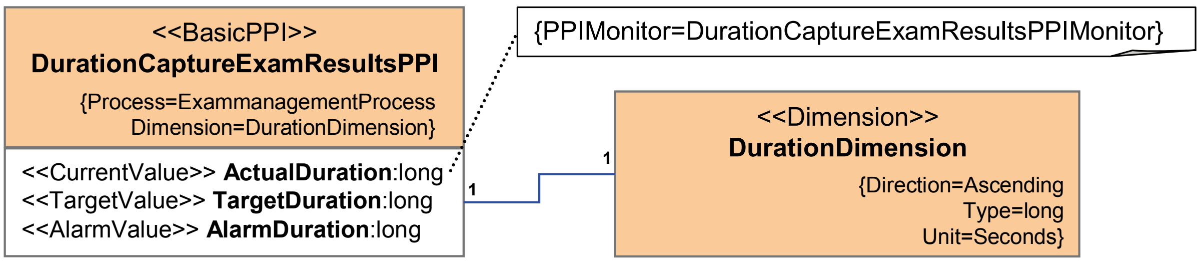

<<BasicPPI>>
DurationCaptureExamResultsPPI
{Process=ExammanagementProcess
Dimension=DurationDimension}
<<CurrentValue>> ActualDuration:long
<<TargetValue>> TargetDuration:long
<<AlarmValue>> AlarmDuration:long
{PPIMonitor=DurationCaptureExamResultsPPIMonitor}
1
1
<<Dimension>>
DurationDimension
{Direction=Ascending
Type=long
Unit=Seconds}

Figure 4.4: PPI specification using Momm et al. approach, taken from [54]

4.5 Momm et al. Approach

Momm et al.'s approach [54] consists of a top-down approach for developing an uniform IT support based on *Servie Oriented Architecture (SOA)* in conjunction with the monitoring aspects required for processing the PPIs. Momm et al build the approach on the principles of the *Model Driven Architecture (MDA)* to enable the support of different SOA platforms as well as an automated generation of the required instrumentation and monitoring infrastructure. Particularly, they present a metamodel for the specification of the PPI monitoring, an extension of the BPMN metamodel for modeling the required instrumentation for the monitoring, and an outline of methodology for an automated generation of this instrumentation. Figure §4.4 depicts the example of the specification of a PPI provided in [54] for the PPI "students' waiting time for their result must not exceed 3 weeks"

A most recent work is presented in [55], where they adapt their approach to web service compositions. In this work they also provide a metamodel for the specification of PPIs, including the following attributes: *name*, *description*, *units*, *type*, *direction* (ascending or descending) and *default*. Actually, they claim to provide a template-based mechanism for defining PPIs, but such a definition is intended for the PPI calculation, being aimed at developers and only taking into account aspects of the implementation level.

4.6 Pedrinaci et al. Approach

Pedrinaci et al. [70] describe a Semantic Business Process Monitoring Tool called SENTINEL. One of its two modules is the Metrics Computation Engine, that is in charge of supporting the automated computation of general purpose as well as user-defined PPIs (metrics for them). For such a definition of PPIs, an ontology is provided. They refine PPIs into two disjoint kinds, functions, that can be evaluated over a fixed number of inputs (for instance a process instance PPI), and aggregations, that take an arbitrary number of individuals of the same kind (e.g. a set of process instances) as input. They also define the concept of population as the way to filter that information to be taken into account to compute the PPI value (to focus e.g. in certain processes or resources). Furthermore, each PPI has a computation expression, defined using Operational Conceptual Modelling Language (OCML) [59].

This tool can also support automated reasoning, though the authors point out that one aspect to be improved is the analysis engines in order to support deviations.

4.7 Wetzstein et al. Approach

KPI	**Computation Expression**
Deadline Adherence	$\dfrac{\sum \begin{array}{l}\text{ShipmentAckEvent.timestamp} <= \text{Order.assuredDeliveryDate} \\ \text{AND OrderDeliveredSuccessfully(Order)} \\ \text{AND ElectronicProductOrder(Order)}\end{array}}{\sum \text{OrderReceived(Order)}} \times 100\%$
Process Duration	Avg (ShipmentAckEvent.timestamp – ReceiveOrderEvent.timestamp)

Figure 4.5: PPI definition examples according Wetzstein et al., taken from [107]

In [107], Wetzstein et al. introduce a framework for BAM as part of the semantic business process management. They state that, since semantic business processes specify the semantics of inputs, outputs, preconditions and postconditions of activities by means of business objects, PPIs can be directly defined over this business objects. For this purpose, they present a PPI ontology using WSML to specify PPIs over these semantic business processes. Concretely, for every PPI, they define the attributes *name*, *description*, *targetValue* (deviations and alerts can also be defined) and *analysisPeriod*. Furthermore, they allow the definition of, on the one hand, instance measures (duration between activities, the state of the a process instance and composition of them can be measured), and on the other hand, aggregated measures (by aggregating several instances), and composed aggregated measures. The computation expressions associated to these PPI definitions are mapped to WSML logical expressions that are used as queries to the WSML reasoner to obtain their value.

Figures §4.5 and §4.6 depict respectively two PPI definitions and a WSML logical expression corresponding to the first PPI.

```
?order[assuredDeliveryDate hasValue ?plannedDate] memberOf Order
    and ?event[activity hasValue ?activity] memberOf ExecutionEvent
    and ?activity[name hasValue “Receive Shipment Acknowledgement”]
    and ?event[timestamp hasValue ?eventDate]
    and (dateLessThan(?eventDate, ?plannedDate)
        or (dateEqual(?eventDate, ?plannedDate))
    and OrderDeliveredSuccessfully(?order)
    and ElectronicProductOrder(?order)
```

Figure 4.6: WSML logical expression example according Wetzstein et al., taken from [107]

4.8 Gonzalez et al. Approach

González et al [39] present MMC-BPM language, a Domain Specific Language to complemet BP models with monitoring, measurement and control (MMC) concerns. It is a declarative specification that contain three main blocks. The Data Block contains the data manipulated in the process and new data required to analyse it. The Event block represents the process domain events used in the definition of the analyses rules and the measurement points in the process. It allows to define process events, that define the moment in the process when a rule must be triggered, and logic events, that allows to group multiple process events. The MMC (or Rule) Block allows to define actions to measure the process and to control its execution. Every action is associated to a logic event.

Figure §4.7 depicts the MMC specification for a loan process example (taken from [39]).

```
MMCspec QualityView process LoanApproval
 //Data
 import dataTypes LoanProcess.xsd;
 include data LoanProcess.pdata;

 int AR; int RR; int TR; double RRA; double RRR;
 //Events
 event requestStateChange parameters boolean request;

 onChange data.NotifyDecision.requestState
  generates requestStateChange using data.NotifyDecision.requestState;
 //Rules
 mmcrule UpdateRequestState onEvent requestStateChange do
  TR = TR + 1;
  if !(event.request) then
   RR = RR + 1; RRR = RR / TR;
  else
   AR = AR + 1; RRA = AR / TR;
  endif
  if RRR*100 > 50 then
   notify 'quality@bank.com' 'RRR is too high' 'review the log.';
  endif
 endRule
endMMC
```

Figure 4.7: MMC specification for a loan process, taken from [39]

4.9 Popova and Sharpanskykh Approach

Popova et al. present in [77] a framework for modeling PPIs (performance indicators for them-PI) within a general organisation modeling framework. They define indicators by assigning values to a set of attributes. These attributes are: *name*, *definition*, *type* (continuous or discrete), *time frame*, *scale*, *min value* and *max value*, *source* (company policies, business plans...), *owner* (role or agent whose performance it measures), *threshold* and *hardness* (soft-qualitative, or hard-quatitative). Figure §4.8 depicts one PPI example extracted from [77].

PI name: **PI27**;
Definition: time to create a new short-term plan after all operational
data is received;
Type: continuous;
Time frame: month;
Scale: REAL;
Min value: 0;
Max value: max_time_CSP;
Unit: hour;
Source: job descriptions;
Owner: daily planning departments;
Threshold: 24 h;
Hardness: hard.

Figure 4.8: PPI example according to Popova et al. approach [77]

Furthermore, an expression is defined over these PPIs. It can be evaluated to a numerical, qualitative or boolean value for a time point. For example, using the above PPI, an associated expression is: $PI27 \leq 48h$.

They formalise these PPIs definitions by means of a variant of the first order sorted predicate language, and define relationships between them. Concretely relations of causality (positive and negative), correlation and aggregation can be defined. In these relations' definitions, they use Temporal Trace Language (TTL) [92] to express the temporal aspect. An excerpt of the PPI relationships for the case study presented in [77] is presented in Figure §4.9.

They also relate PPIs to processes, roles, agents, goals and environmental characteristics, as follows:

$$measures : PPI \times PROCESS$$

$$has\_owner : PPI \times \{ROLE, AGENT\} \times \{very\_low, low, medium, high, very\_high\}$$

$$is\_based\_on : GOAL\_PATTERN \times PPI$$

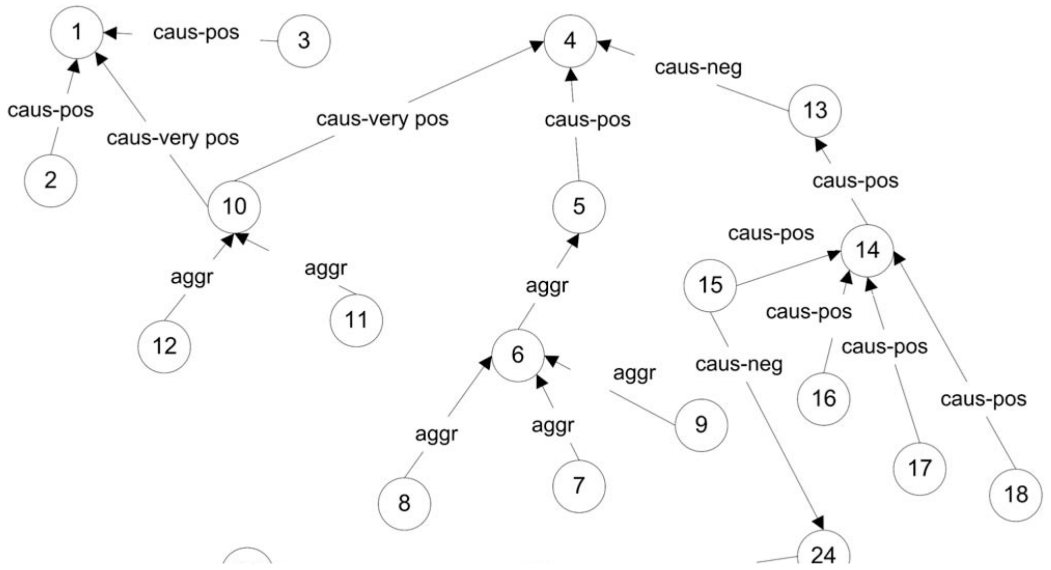


Figure 4.9: PPI relationships example according to Popova et al. approach [77]

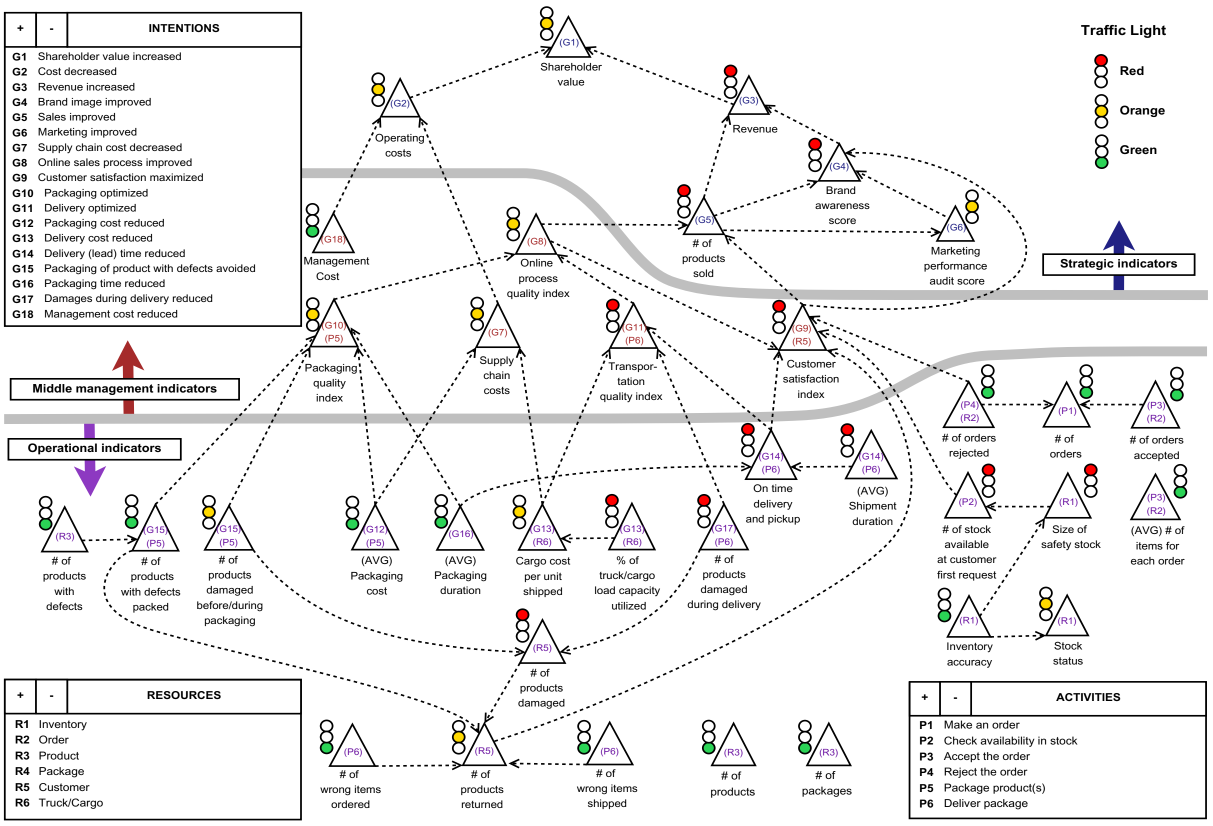
$uses : GOAL\_PATTERN \times PPI\_EXPRESSION$

$env\_influence\_on : ENV\_CHARACTERISTIC \times PPI \times \{pos, neg\}$

In addition, these relations are further investigated in [76, 78], where they present formal techniques for the analysis of executions of organizational scenarios, and discuss analytic issues for consistency and verification checks of the goal structures and between goals and PPIs, respectively.

4.10 Barone et al. Approach

Barone et al. present in [4] the Business Intelligence Model (BIM), whose main goal is to allow business users to conceptualise business operations and strategies, and performance indicators, so that they can be connected to enterprise data through automated tools. They propose to define a global view of a company's workflows and define PPIs on its activities and resources. The set of PPIs defined together with the relationships among them constitute the so-called Indicators Graph. Figure §4.10 shows the indicators graph provided [4] according to its case study.



+	-	INTENTIONS
G1		Shareholder value increased
G2		Cost decreased
G3		Revenue increased
G4		Brand image improved
G5		Sales improved
G6		Marketing improved
G7		Supply chain cost decreased
G8		Online sales process improved
G9		Customer satisfaction maximized
G10		Packaging optimized
G11		Delivery optimized
G12		Packaging cost reduced
G13		Delivery cost reduced
G14		Delivery (lead) time reduced
G15		Packaging of product with defects avoided
G16		Packaging time reduced
G17		Damages during delivery reduced
G18		Management cost reduced

+	-	RESOURCES
R1		Inventory
R2		Order
R3		Product
R4		Package
R5		Customer
R6		Truck/Cargo

+	-	ACTIVITIES
P1		Make an order
P2		Check availability in stock
P3		Accept the order
P4		Reject the order
P5		Package product(s)
P6		Deliver package

Traffic Light
Red
Orange
Green
Strategic indicators
Middle management indicators
Operational indicators
(G1) Shareholder value
(G2) Operating costs
(G3) Revenue
(G4) Brand awareness score
(G5) # of products sold
(G6) Marketing performance audit score
(G18) Management Cost
(G8) Online process quality index
(G10) (P5) Packaging quality index
(G7) Supply chain costs
(G11) (P6) Transportation quality index
(G9) (R5) Customer satisfaction index
(P4) (R2) # of orders rejected
(P1) # of orders
(P3) (R2) # of orders accepted
(G14) (P6) On time delivery and pickup
(G14) (P6) (AVG) Shipment duration
(P2) # of stock available at customer first request
(R1) Size of safety stock
(P3) (R2) (AVG) # of items for each order
(R3) # of products with defects
(G15) (P5) # of products with defects packed
(G15) (P5) # of products damaged before/during packaging
(G12) (P5) (AVG) Packaging cost
(G16) (AVG) Packaging duration
(G13) (R6) Cargo cost per unit shipped
(G13) (R6) % of truck/cargo load capacity utilized
(G17) (P6) # of products damaged during delivery
(R1) Inventory accuracy
(R1) Stock status
(R5) # of products damaged
(P6) # of wrong items ordered
(R5) # of products returned
(P6) # of wrong items shipped
(R3) # of products
(R3) # of packages

Figure 4.10: Indicators graph example according to Barone et al.'s approach, taken from [4]

4.11 Friedenstab et al. Approach

Friedenstab et al. have recently presented in [33] a twofold contribution: on the one hand, a metamodel that extends BPMN in order to include BAM-relevant concepts, including PPIs; on the other hand, a graphical notation for those concepts described in the metamodel. They support the definition of PPIs to measure duration and frequency (the number of times that something happens) of process instances, to compose them by arithmetic operations, and to aggregate them considering a set of instances, delimited by means of the filter. Furthermore, target definitions can be defined for these PPIs and, actions in order to react when these targets are not achieved.

Figure §4.11 depicts the BAM model example provided in [33] to define cycle times, according to the case study presented in that work.

This work does not provide any mechanism for these PPI definition analysis.

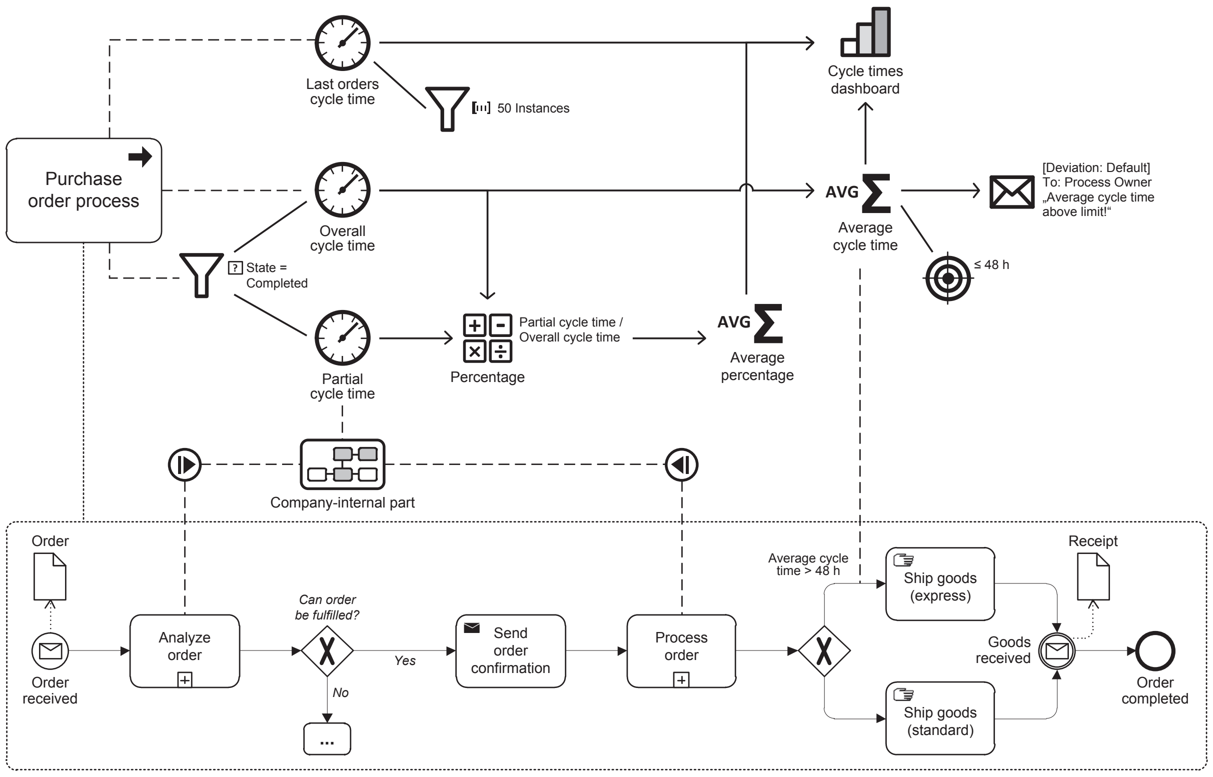


Figure 4.11: BAM model for cycle times according Friedenstab et al., taken from [33]

4.12 Other Related Approaches

In this section we comment other related approaches more restrictive regarding the definition and analysis of PPIs, or taken form other areas that served as inspiration for the work performed in this dissertation.

4.12.1 GRAI/GIM Approach

GRAI [15] is a conceptual model for manufacturing systems that divide them into three subsystems: physical, decision and information systems. Over this conceptual model is build the GIM approach, GRAI Integrated Methodology. Within the modelling formalisms provided by GIM, GRAI Grid [25] is used to build a decision system model. In this formalism, the definition of performance indicators are defined, but not focused on business processes. They establish three parameters or attributes to define performance indicators: name, value domain or dimension and procedure to calculate the value. They also define the relation of these performance indicators with objectives and decision variables.

4.12.2 Soffer et al. Approach

Soffer et al propose in [93] a formal framework that defines a process model on the basis of states and state variables. Furthermore, they also define PPIs (referred to as soft-goals) and their relationships to processes and state variables. Criterion functions (any function on the values of state variables) are used to define these PPIs. They put especial emphasis in relating criterion functions and PPIs to processes, so that PPIs are defined over the the appropriate BPs. They apply their approach to the SCOR model and perform a PPI analysis. This analysis allows to identify, among others, which criterion functions are fully dependent on the process, which are partially dependent on it.

4.12.3 Korherr and List Approach

Korherr et al. extend in [47] the BPMN and EPC metamodels to define business process goals and performance measures. They only allow the definition of cost, quality and cycle time measures, from which only cycle time measure are explicitly connected to the business process elements. They do not delve into the information required to define such measures and to calculate them.

4.12.4 Costello et al. Approach

Costello et al. propose in [16] a model to include the definition of PPIs into process models using XML. They define events associated to what they call process (business activity). These events are mainly intended to the calculation of cycle time PPIs. They also present a mapping of this event-based model to an ontology developed using OWL (the Web Ontology Language). This ontology serves as a basis for defining rules for the calculation of PPI values. Finally, they provide a software implementation called iWISE architecture to support their approach.

4.12.5 García et al. Approach

A work somehow related to this area, but focused on software measurement rather than processes is the one presented in [35]. The authors describe in this work a software measurement ontology called SMO. In [57] they also present a graphical notation for the depiction of software measurements, based on SMO.

4.13 Summary

In this chapter we have presented those related work that provides partial support for the PPIM lifecycle phases described in previous chapter. We have described the way each proposal address the definition of PPIs and the level of analysis support provided.

Part III

Our Contribution

Motivation

The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them.

William Bragg (1862–1942), British physicist and Nobel Prize in Physics 1915

Our goal in this chapter is to describe the problems identified during our research and to motivate the contributions provided in this dissertation. We analyse the current solutions and bring attention to the advances we have achieved to solve these problems. In Section §5.1, we motivate our research. Section §5.2 presents a motivating scenario that will be used along the dissertation. In Section §5.3, we present the main problems addressed in this dissertation. In Section §5.4, we analyse the level of compliance of the presented problems by the current solutions found in the literature. In Section §5.5, we resume and compare the information previously obtained and contextualise our contributions.

5.1 Introduction

Supporting the PPIM lifecycle requires to overcome a number of problems introduced in Chapter §1. The final goal of this chapter is twofold. First, to motivate the need for specifics techniques and tools to improve both the design and the instrumentation phases of the PPIM lifecycle; and to do so we describe in detail the aforementioned problems. Second, to introduce our contributions in this topic, that are the result of an extensive analysis of a variety of PPIs defined by different organisations, a careful study of the related literature, and the application of the knowledge gained in previous experiences with the automated analysis in other areas like feature models and SLAs.

We consider these contributions can promote the progress of the discipline both at a technical and a social level. From a technical standpoint the application of DL to the PPI definitions helps to eliminates ambiguities and provides more semantics about some concepts and relationships, and facilitates their automated analysis by leveraging reasoner services available for such a formalism. From a social standpoints, the techniques proposed for defining, representing and analysing PPIs will thrive even more the aforementioned process orientation in organisations.

5.2 Motivating Scenario: Process of the Request For Change Management

Our motivating scenario for the work developed in this dissertation is presented in this section. It takes place in the context of the Information Technology Department of the Andalusian Health Service. We particularly focus on the business process of managing Request for Changes for existing Information Systems. This process was modelled by the quality office of this department using BPMN, but due to space and in order to make it easier to understand, we have simplified the real process obtaining the diagram depicted in Figure §5.1.

The process starts when the requester submits a Request For Change (RFC). Then, the planning and quality manager must identify the priority and analyse the request in order to make a decision. If the RFC was in the strategic plan or pre-approved, the requester will be asked to submit a release request and the process will continue through the global Project Management Process (PMP). Otherwise, according to several factors like the availability of resources, the requirements requested, and others, the RFC will be either approved, cancelled, raised to a committee for them to make the decision, paralysed or sent to the area manager in order for her to negotiate new requirements.

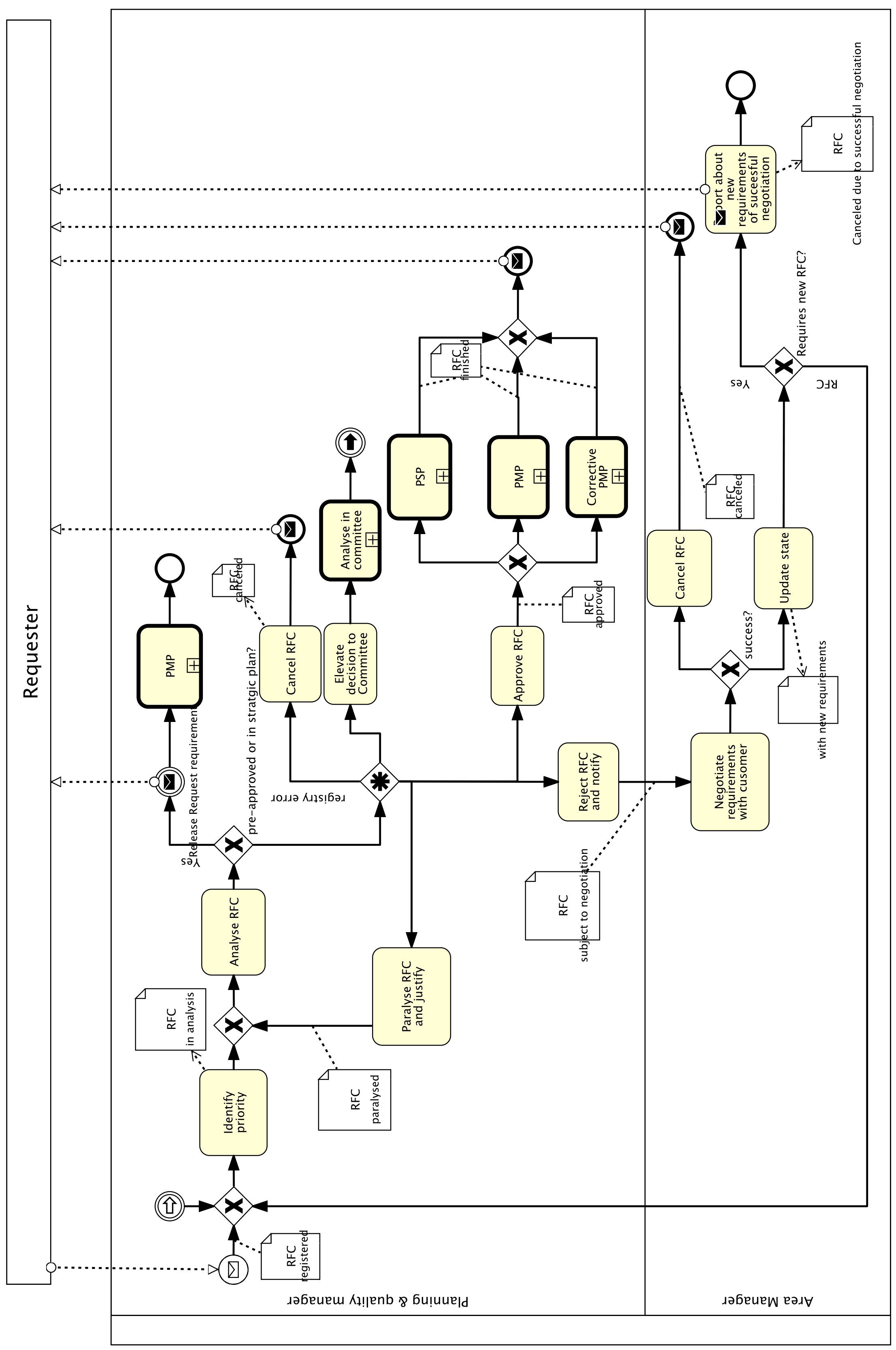


Figure 5.1: Process of the request for change management

Name	Description	Target V	Periodicity
PPI1	RFCs cancelled-registry error from RFCs registered	4%	weekly
PPI2	Average time of committee decision	1 working day	weekly
PPI3	corrective RFCs from approved RFCs	2%	weekly
PPI4	perfective and adaptive RFC from approved RFCs	4 %	weekly
PPI5	Average time of the "analyse RFC" activity	2 working days	weekly
PPI6	Number of RFCs with the state "in analysis"	2 RFCs	weekly
PPI7	Number of RFCs per type of change	corr-20, evol-30, perf-20	monthly
PPI8	Number of RFCs per project	rr.hh-50, diraya-60, pharma-1	monthly
PPI9	Average lifetime of a RFC	3 working days	monthly

Table 5.1: PPIs defined for the RFC management process

In addition, throughout the process, the RFC document can pass through several states: *registered*, *in analysis*, *paralysed*, *cancelled*, *approved*, *subject to negotiation*, *with new requirements* and *cancelled due to successful negotiation*.

Apart from the process model, this department also defined a set of indicators associated with it. They did it using natural language and collected them in tables. Again, for the sake of simplicity, we only show an excerpt of this table (Table §5.1). Every PPI in this table is defined through a name (first column), a description (second column), a target value ("Target V" for short, in the third column) and a periodicity (in the fourth column, corresponding to the scope concept we introduce in section §6.5). Target values reflected in this table are invented due to privacy reasons. In the original version of this table there is also a responsible for all these PPIs, the "planning and quality manager" for all of them. It does not appear in the table due to space constraints.

5.3 Problems

Taking into account the previous scenario, a number of problems were identified that motivated the main contributions of this dissertation. In the following we present these problems, all of them related to the design and instrumentation phases of the PPIM lifecycle and very interrelated to each other. We classify them according to three aspects, the definition of PPIs, their representation, and their automated analysis.

Regarding the **definition of PPIs** , we have identified three problems

P1: Lack of traceability with BPs. The current trend is to define PPIs in some ad-hoc way, usually in a separated document (for instance tables or spreadsheets like in the motivating scenario) without establishing any link or relationship with the elements of the corresponding business process. This leads to several subproblems: difficulty to maintain the coherence across both worlds (BPs and PPIs), *e.g. what happens if an activity is removed and there exists a PPI that is only defined over that activity?*, or *What happens if a PPI is defined by referencing an activity whose name does not correspond to the one in the BP?*; another subproblem is the difficulty to instrument the process in order to gather the information required to calculate PPI values, since it is not clearly defined where in the process PPIs must be measured.

P2: Ambiguity and incompleteness. This problem is derived from the fact of defining PPIs using natural language. In this case, it is very likely to leave out of consideration important information for the computation of PPIs or to use ambiguous expressions, making it difficult to translate such definitions to the instrumentation of a process that calculate real values for that PPIs from the business process execution. Let us suppose we have, in our motivating scenario, the PPI definition in natural language "duration of the analysis activity". *Which "analysis activity" does this PPI refers to?, the "analyse RFC" or the "analyse in committee" activity?*. Furthermore, *when should this PPI value be gathered?* This information is missing in the aforementioned definition.

P3: Limited expressiveness. Most existing research proposals allow the definition of PPIs related to control flow and time; however, those related to the state of the process, to data or to resources are usually disregarded; for instance, PPI3 from Table §5.1 could not be defined. Furthermore they usually present limited expressiveness regarding PPIs calculated using other PPIs (by aggregating several process instances or by applying certain mathematical functions) and grouping values according to certain dimensions, like it is required to define PPI8 from Table §5.1.

Regarding the **representation of PPIs** , two additional problems have been identified.

P4: Lack of understandability by non-technical users. To cope with the limitation described in P2 (ambiguity and incompletenss), there exist several approaches that provide some languages or specifications for the definitions of PPIs, but they are low level approaches, and lead to early formalisation of PPIs, what is detrimental to their understandability by non-tecnical users (like stakeholders or expert-domain).

P5: Visual gap between BP models and PPI definitions. Business processes are usually expressed in a graphical notation to ease communication about these processes between stakeholders. However, PPI definitions are usually expressed either by means of natural language, or using lower level languages, closer to the implementation perspective. In both cases there exists a visual gap between BPs and PPIs, *i.e.* the different departments or roles in charge of, on the one hand, the modelling and execution of business processes, and, on the other hand, the definition and consecution of goals and its associated indicators, have partial views: nobody has a comprehensive view of both worlds. This leads again to a problem with the maintenance of coherence across them.

Finally, regarding the **analysis of PPIs** , one of the main problems identified is the following one.

P6: Not amenability to automated design-time support. It is highly desirable to have PPI definitions that allow to perform a design time analysis in order to obtain information that can assist BP analysts during that definition and the subsequent evolution. Until now, the effort on the automated analysis has been driven to the development of techniques and tools to automatically analyse BPs and PPIs at runtime (specially BAM techniques) and post mortem, *i.e.* using the information obtained from those process instances already finished (*e.g.* techniques from the fields of business intelligence, data warehousing and process mining). Unfortunately, little efforts have been conducted to automatically analyse PPIs at design time.

Apart from the previous set of problems, we have established two additional restrictions or requirements for our proposal.

R2: Tool support. Though some industry products have been developed to provide support for the definition and analysis of PPIs, like for instance the ARIS Process Performance Manager (PPM) [88], it leaves some aspects unconvered. To the best of our knowledge, there not exists any tool that supports a definition of PPIs that copes with all the aforementioned problems. This might be due to several reasons: the market immaturity, the low performance that the usage of formalisms leads to, or most probably that, a priori, the ROI (Return Of Investment) is not assured for this kind of solutions.

R2: *Business Process Management (BPM)* standards support. From the research approaches that propose some mechanism for the definition of PPIs, most of them do not support their connection to BP models defined using standard languages and notations like BPMN or *Business Process Execution Language (BPEL)*. Usually, they are highly coupled to particular definitions of BPs, loosing thus the desired flexibility and standardisation for this issue.

5.4 Analysis of Current Solutions

Taking as starting point the set of approaches introduced in Chapter §4, we analyse them here according to the fulfillment of the requirements derived from the problems described in the previous section.

Regarding the *traceability between PPI definitions and BP elements*, most of the approaches take it into account, although many of them do not provide evidences that this traceability can allow to maintain coherence in both directions, *i.e.*, that it is not only used to establish the points in process where PPIs have to be measured. Castellanos et al [12] PPIs are specified by means of templates, where the user must define the connection with the process elements by fulfilling parameters, but it is not clear the way this parameters are obtained from the business process (no model for such specification is presented); in fact, this is not a process oriented-approach. ARIS, in its comertial product PPM [88] also allows to maintain this traceability by defining measurement points in the business process. Mayer et al [51] define some kind of relationship between what they call metrics definitions and activities of services that implement processes. In Momm et al clearly define in [54] the traceability between PPIs and BP elements by means of both, a metamodel for the definition of PPIs and a metamodel for the specification of its monitoring model. Pedrinaci et al outline in [70] this traceability, though they do not delve into the detail. Wetzstein et al also define this traceability, although only to BP activities and in the context of semantic business processes. González et al. [39] define the relation with the business process elements within the data and event blocks of their specification. Popova et al establish in [75] the relationship between the PPI and the process it measures, and then, in [76] they describe properties over process execution traces by means of the order-sorted predicate Temporal Trace Language and assign them to PPIs. This is somehow a way of maintaining the aforementioned traceability. Barone et al.[4] roughly address this issue by including in the indicators constructs, within the indicators graph, the resources and activities to which they are related. Finally, Friedenstab et al.'s approach [33] is heavily based in our approach presented in [22] and also provides this traceability by means of the metamodel defined.

The *ambiguity and incompleteness* problem is addressed by virtually all the research proposals[1], since most of them are based on formalisms. In fact, this problem has been found in the real scenarios analysed, where PPIs are defined using natural language.

The *expressiveness* of the different approaches varies according to the possible kind of PPIs that need to be defined. Time and count (relative to the number of occurrences of certain event) measures can be defined in most approaches, and also most of them allow to aggregate them. However, only Wetzstein et al. [107] and Barone et al [4] allow the definition of PPIs related to the state of the process (we intuit that also ARIS PPM provides some mechanism to do it, but the information is not explicitly available). Regarding PPIs defined over data only González et al. [39] support them, and over resources, only ARIS PPM support them. The possibility to derive PPI definition from other existing ones by means of certain mathematical function is provided by half of the approaches (cf Table §5.2). With respect to the definition of what we call *scope*, to filter the process instances that must be used to calculate PPIs, Popova et al partially address this issue by defining temporal properties in [74], and the approaches of Pedrinaci et al and Firedenstab et al also provide solutions for it. Nevertheless, all of them leave out of consideration some of the scope definitions found in the real scenarios studied during this dissertation.

In relation to the *understandability by non-technical users*, Castellanos et al allow in IBOM defining PPIs by instantiating what they call metric templates, but, until what we can deduce from the information available, these templates do not offer the right level of abstraction required by non-technical users. Another approach that try to get closer to this kind of users is [33], by offering the aforementioned graphical notation. The main problem is the need to have a deep knowledge on BP models and the learning curve such a graphical notation may have for this kind of users. Barone et al. [4] also propose some kind of graphical representation for PPIs though it is quite restrictive (very little elements are used).

With respect to the *visual gap between BP models and PPI definitions*, there only exist three aproximations: the first one is ARIS PPM, that offers the possibility to indicate graphically over BPs defined in EPCs measurement points. The second one has been recently presented by Friedenstab et al [33] and is based in our approach to define PPIs presented in [22]. They

[1]Barone et al. is the only exception, since they do not provide evidences of nay formal foundation for their model.

propose an extension of BPMN to graphically model BAM aspects over BPs but it is quite incipient since is not implemented in any tool nor has been proven in any real scenario. The last one is the proposal of Barone et al. [4], but they do not depict these PPIs over BPs, but they provide indicators graphs.

The *automated analysis at design time* is only partially addressed by Popova et al. They allow the definition and analysis of relationships between PPIs, but do not offer mechanism to derive information regarding the relationship between PPIs and BP elements.

Regarding the requirement of providing *tool support*, most of the approaches analysed present some kind of *software implementation*, but not all of them are intended to be used by final users, but, in many cases, they have been developed as a proof of concept of the research conducted. Obviously, this is not the case of iBOM (HP) of PPM (ARIS), that are commercial products. Pedrinaci et al also have implemented their approach in SENTINEL, but they point out the limitation that such a tool in its current version only provides a predefined set of PPIs (though they plan to augment its flexibility by allowing the definition of domain-specific PPIs).

Finally, regarding the requirement of providing *support to the current BPM standards*, on the one hand, the approaches of Mayer et al. and Pedrinaci et al. include standards like BPEL and WSDL, actually more focused on web services, *i.e.* at an implementation level. On the other hand, Momm et al. , Wetzstein et al. and Friedenstab et al. take as starting point the BPMN specification. González et al. provides support for both, BPEL and BPMN.

5.5 Discussion

The result of the previous analysis of related works is summarised in two ways. First, the Kiviat diagram shown in Figure §5.2 represents the level at which the presented problems are overcome by the existing proposals.

Second, this information is presented with more detail in Table §5.2. The proposals analysed are located at the different rows, and columns correspond to the problems and requirements presented in this chapter, taking into account that problem P3 (expressiveness) has been subdivided into 8 subproblems according to the different issues to consider during PPI definition, and problem P6 (automated analysis) into 2 subproblems, according to the different families of operations described in this dissertation. We use the following notation: A ✓ sign means that the proposal successfully addresses the issue; a $\sim$ sign indicates that it addresses it partially; N/A means the information is not available; and a blank cell indicates that it does not contemplate the issue. We use ✓* for the feature aggregated measures because those approaches that ad-

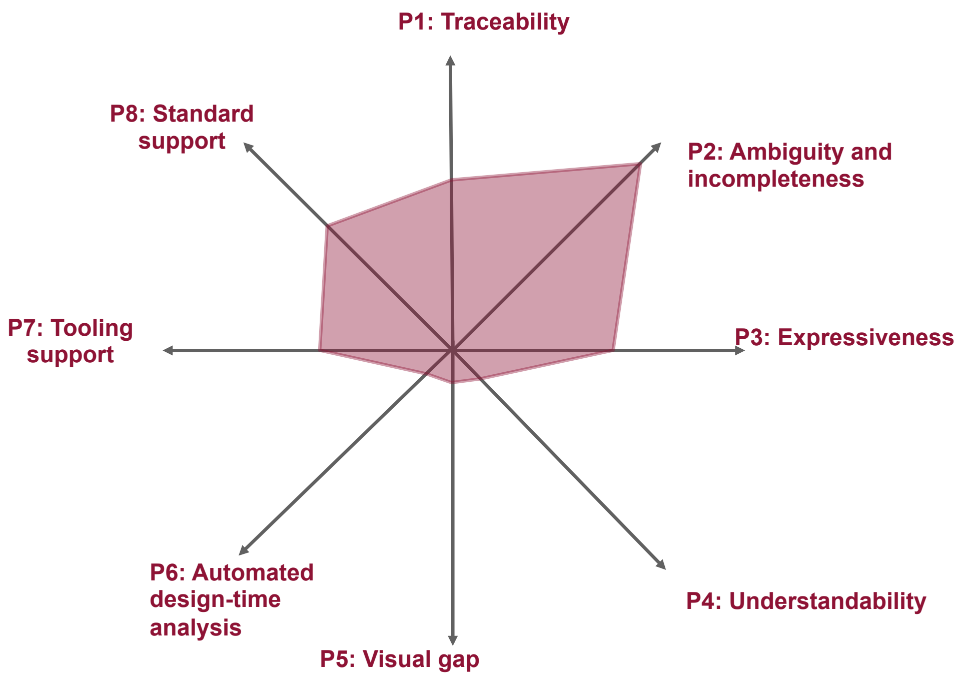


Figure 5.2: Kiviat diagram: level at which the related approaches overcome the existing problems

dresses this feature does not take into account the possibility of grouping by certain dimension (`isGroupedBy` in our proposal).

Both, Figure §5.2 and Table §5.2 show that the problem of ambiguity and incompleteness is the most addressed, since most of the approaches analysed are research works, while the understandability, visual gap and automated design-time analysis problems are the most disregarded. Furthermore, Table §5.2 let us confirm that none of the approaches address simultaneously all the problems and requirements identified (Section §5.3), and conclude that a definition, representation and analysis of PPIs that overcome the aforementioned problems and requirements constitute still an unresolved challenge.

In this dissertation we address this challenge by means of the following contributions: (1) a metamodel (PPINOT metamodel) for the definition of PPIs that is useful along the whole PPI lifecycle; (2) a graphical notation to depict PPIs over BP models based on this metamodel; (3) a set of PPI-templates and L-PATTERNs to improve the definition of PPIs without needing to have a deep knowledge of BP notations; (4) a formalisation of the PPINOT metamodel using DL that allows to define a set of families of analysis operations to extract information at design-time from that PPI definitions; (5) A software tool suite that provides support for all the previous contributions (except for some analysis operations partially implemented yet).

Proposal	P1	P2	P3								P4	P5	P6		R1	R2
			P3.1	P3.2	P3.3	P3.4	P3.5	P3.6	P3.7	P3.8			P6.1	P6.2		
Castellanos et al. [12]	∼	✓	✓	✓	∼	N/A		✓*	∼		∼				✓	
ARIS PPM [88]	∼	✓	✓	✓	∼		✓	✓	✓	∼	N/A	∼			✓	
Mayerl et al. [51]	∼	✓	✓	✓	N/A			✓*	✓					∼		✓
Momm et al. [54]	✓	✓	✓	✓				✓*							∼	✓
Pedrinaci et al. [70]	N/A	✓	✓	✓	N/A			✓*	✓	✓			∼	∼	✓	✓
Wetzstein et al. [107]	✓	✓	✓	✓	✓			✓*	✓	∼				∼	∼	✓
González et al. [39]	✓	✓	✓	✓	∼	✓	∼	✓*	✓						∼	✓
Popova et al. [77]	✓	✓	✓	N/A	N/A	N/A		✓*		✓	N/A			✓	∼	
Barone et al. [4]	∼		✓	✓	✓		✓	✓*	✓		∼	∼				
Friedenstab et al. [33]	✓	✓	✓	✓	∼			✓*	✓	✓	∼	✓				✓
PPINOT	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓

(P1) Traceability
(P2) Ambiguity and incompleteness
(P3.1) Time measures
(P3.2) Count measures
(P3.3) Condition measures
(P3.4) Data measures
(P3.5) Resource measures
(P3.6) Aggregated measures
(P3.7) Derived measures
(P3.8) Definition of scope
(P4) Understandability
(P5) Visual gap
(P6.1) PPI-BP Interaction
(P6.2) Relationships among PPIs
(R1) Tooling support
(R2) Standards support

Table 5.2: Comparison of the analysed approaches and our proposal

5.6 Summary

In this chapter, we have presented the main problems that motivated this dissertation. We have analysed the related literature on the definition and analysis of PPIs and observed that no proposal exists that overcomes all the problems identified. We have also emphasised the value and originality of our main contributions.

PPINOT Metamodel

When you can measure what you are talking about and express it in numbers, you know something about it.

Lord William Thomson Kelvin (1824-1907),
British mathematical physicist and engineer

To overcome the limitations related to the definitions of PPIs introduced in Chapter §5, we present here PPINOT metamodel. Its main advantages are: (1) it enables a seamless relationship between PPIs and business process models, which makes the use of PPIs along the business process lifecycle easier; (2) it eliminates the ambiguity and incompleteness problem of natural language; (3) it allows the definition of commonly used PPIs that, to the best of our knowledge, cannot be defined with other similar proposals, specially those related to data; (4) it provides these definitions of PPIs with the amenability to be automatically analsyed at design-time, as we describe in Chapter §9. The chapter is organised as follows: in Section §6.1 we introduce it; Section §6.2 briefly comments a set of considerations we make with respect to BP models; then, Sections §6.3, §6.4 and §6.5 describe in detail the metamodel; finally, we summarise the chapter in Section §6.6.

6.1 Introduction

When managing PPIs, the first obstacle is to delimit a PPI conceptually since there is no consensus about the key elements and their relationships that need to be taken into account when defining PPIs. Consequently, it is necessary to establish a representation method simple and easy to understand, but also expressive enough to accommodate the different domains and situations where PPIs can be defined. In this chapter, we tackle this problem by introducing the PPINOT metamodel. This metamodel is the result of an extensive analysis of a variety of PPIs defined by different organisations, a careful study of the related literature and a process of successive refinements of the metamodel after applying it to different scenarios.

The following sections detail the main features of the metamodel, which has been driven by these requirements:

- Must allow the definition of SMART PPIs: As with other indicators, it is recommended that PPIs satisfy the SMART criteria [91]. SMART is an abbreviation for five characteristics of good indicators, namely: *Specific* (it has to be clear what the indicator exactly describes), *Measurable* (it has to be possible to measure a current value and to compare it to the target one), *Achievable* (it makes no sense to pursue a goal that will never be met), *Relevant* (it must be aligned with a part of the organisation's strategy, something that really affects its performance) and *Time-bounded* (a PPI only has a meaning if it is known the time period in which it is measured). Therefore, the metamodel must allow the definition of PPIs according to the SMART criteria.

- Must have a high expressiveness: The metamodel must be able to express all of the PPIs found in both the literature review and in the organisations whose PPIs have been analysed so that it can provide a solid basis for defining PPIs in any organisation.

- Must be compatible with BPMN: The metamodel must be compatible with BPMN since it is the standard *de facto* in the industry to define business processes.

- Must be extensible: The metamodel must provide mechanisms to be extended according to several variation points, namely: the type of measure used by the PPI, the expression of its target value and the definition of its scope in terms of the subset of process instances used to calculate the value of the PPI.

6.2 Business Process Model Considerations

First of all we establish some considerations regarding process models we will use later on in our proposal. These considerations will be refined when applying our PPINOT Metamodel to a concrete language or notation.

Every business process includes BP elements that can be flow elements or dataObjects. Examples of flow elements in BPMN are activities or events.

Any BP element when instantiated has a state associated that changes along the process instance evolution. The set of state values for every BP element changes depending on the language or notation. Except for dataObjects, whose state values are usually defined by the user, we consider that there exist one or a set of values corresponding to the start or activation of the BP element, and one or a set of values for its end. For instance, in BPMN 2.0 activities can have the following states: ready, active , withdrawn, completing, completed, failing, failed, terminating, terminated, compensating and compensated. In this case, the start corresponds to the change to state active, and the end can correspond to withdrawn, completed, failed, terminated or compensated.

6.3 Conceptual Modelling of PPIs

As stated before, *Process Performance Indicators (PPIs)* can be defined as quantifiable metrics that allow to evaluate the efficiency and effectiveness of business processes. They can be measured directly by data that is generated within the process flow and are aimed at the process controlling and continuous optimization [14].

Consequently, the PPINOT metamodel (c.f. Figure §6.1) is defined according to this definition and taking into account the requirement that it must allow de definition of SMART PPIs. In particular, its attributes are defined as follows:

- `identifier: string`. Every PPI must be uniquely identified by an identifier, that usually will be a number preceded by PPI.
- `name: string`. This attribute provides a descriptive name for every PPI.
- `relatedTo: Process`. This attribute makes reference to the process for which the PPI is defined.

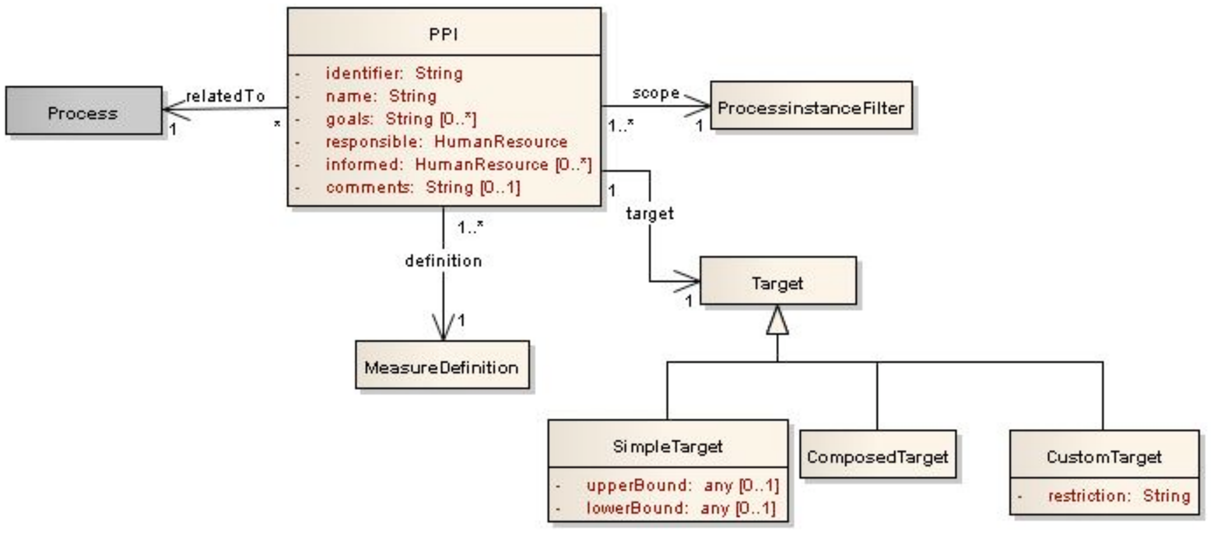


Figure 6.1: PPIs in PPINOT Metamodel

- `goals: string [0..*]`. This attribute allows the user to establish the strategic or operational goal/s that the PPI is related to. It highlights the relevance of the PPI (connecting to the *Relevant* characteristic of the SMART criteria). It can be fulfilled with an expression in natural language. A more formal definition of the relationship between the PPI and the organisational goals is out of the scope of this paper. Some approaches regarding this issue can be found in [77, 78]

- `definition: MeasureDefinition`. This attribute provides a definition about how the indicator is measured. With this field, the two first characteristics of the SMART criteria (*Specific* and *Measurable*) are fulfilled. In the next section, a detailed description of `MeasureDefinition` is provided.

- `target: Target`. Every PPI has an associated target value to be reached indicating the consecution of the previously defined goals. In order to fulfill the *Achievable* characteristic of the SMART criteria, this target value must be reasonable, based on previous experiences and predictions based on simulations. PPINOT allows the definition of three kinds of target values: a simple target, a composed target and a custom target. A simple target is used to specify the lower bound and/or upper bound that make up the range within which the PPI value should be (If only an upper bound is defined, it acts as a maximum; if only a lower bound is defined, it acts as a minimum; finally, if both bounds are set, they define a range within which the PPI value must be). The composed target allows to define several target values or ranges, for those cases where the value of the PPI is a map (*e.g.* PPI7 and PPI8 from our case study). Finally the custom target offers the possibility to define a restriction that the PPI value must fulfill (allowing a higher case mix for the target value, e.g. utility functions can be defined, or a metamodel of preferences like the one presented in [36] can be used).

- `scope: Filter`. This attribute indicates the subset of instances of the perviously specified process that must be considered to compute the PPI value. This field is related to the *Time-Bounded* characteristic of the SMART criteria.
- `responsible: HumanResource`. This attribute holds the human resource in charge of the PPI. This human resource can be a person, a role, a department or an organisation. A more detailed definition of the types of human resources are out of the scope of this metamodel. However, some approaches regarding this issue can be found in [10, 11].
- `informed: HumanResource [0..*]`. This attribute represents the human resources that are interested in the PPI, i.e., who must be informed. This human resource can be also a person, a role, a department or an organisation. Unlike the responsible, which must be only one person, there may be many people informed about the state of the PPI.
- `comments: String`. Other information about the PPI that cannot be fitted in previous fields can be recorded here.

In the following sections, we detail how `MeasureDefinition` and `Filter` are defined in the PPINOT metamodel. Furthermore, we also introduce the concept of `Condition`, which is necessary to express the relationship between `MeasureDefinitions` and the business process.

6.4 Measure definitions

Figure §6.2 depicts the two dimensions into which the definition of measures for PPIs can be classified. The first dimension (Y axis) is the number of process instances necessary to calculate the PPI value. There are two possible values in this dimension, namely: *single-instance measures* if a single process instance is used to calculate the measure, and *multi-instance measures* if the PPI value is calculated using a set of process instances. Usually, most PPIs are defined using multi-instance measures. The second dimension (X axis) is the way in which the PPI value is calculated. In this case PPIs can be defined over the following types of measures:

- *time measure*: it reflects the duration between two time points in the process.

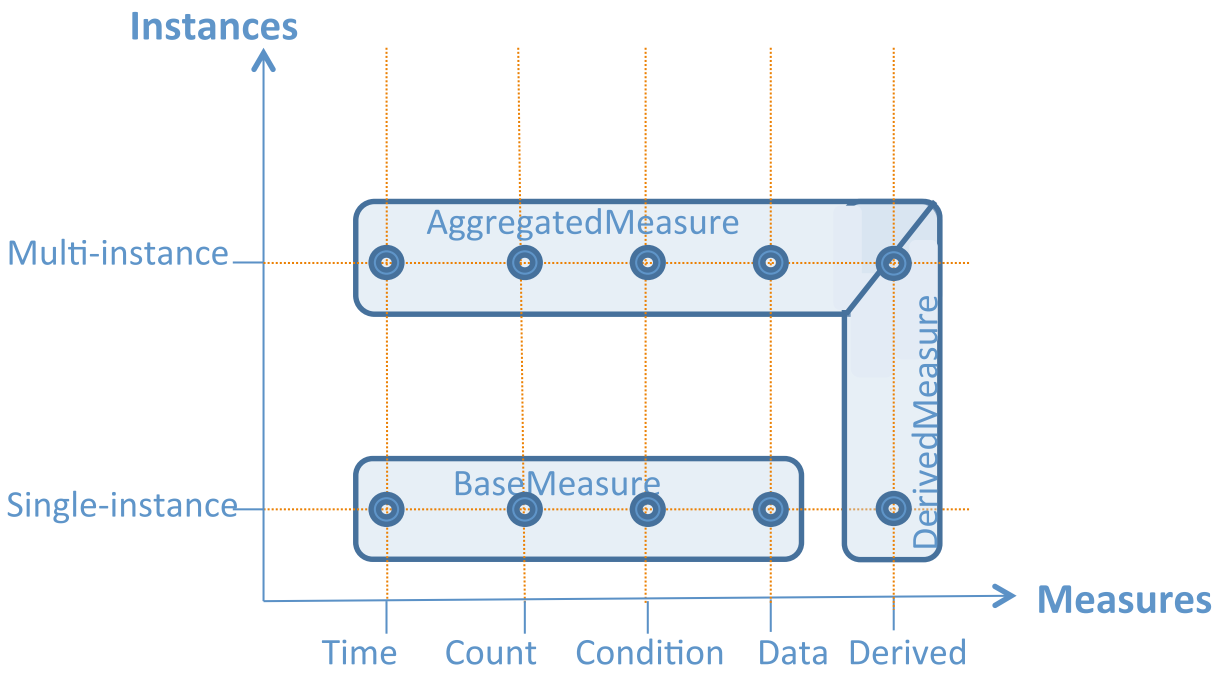


Figure 6.2: PPI dimensions

- *count measure*: it counts the number of times certain condition is satisfied.
- *condition measure*: it checks if certain condition is (for running instances) or has been (for finished instances) met.
- *data measure*: it takes the value of a data property of certain dataObject.
- *derived measure*: it is calculated by performing a mathematical function over any number of measures previously defined.

These dimensions are captured in the PPINOT metamodel by means of three classes that extend `MeasureDefinition`: `BaseMeasure`, `AggregatedMeasure` and `DerivedMeasure` (cf. Figure §6.3). The relationship of these classes with the dimensions are depicted in Figure §6.2. A `BaseMeasure` represents single-instance measures that measures time, count, conditions and data. An `AggregatedMeasure` represents multi-instance measures that can be defined as an aggregation of single-instance measures (i.e. multi-instance measures that measures time, count, conditions and data). Finally, a `DerivedMeasure` represents either a single-instance or a multi-instance measure that calculates the value of the PPI by performing a mathematical function over other measures.

The reason for considering `DerivedMeasure` as a top-level class in the metamodel is twofold. First, the way in which the value of the PPI is calculated is conceptually different from

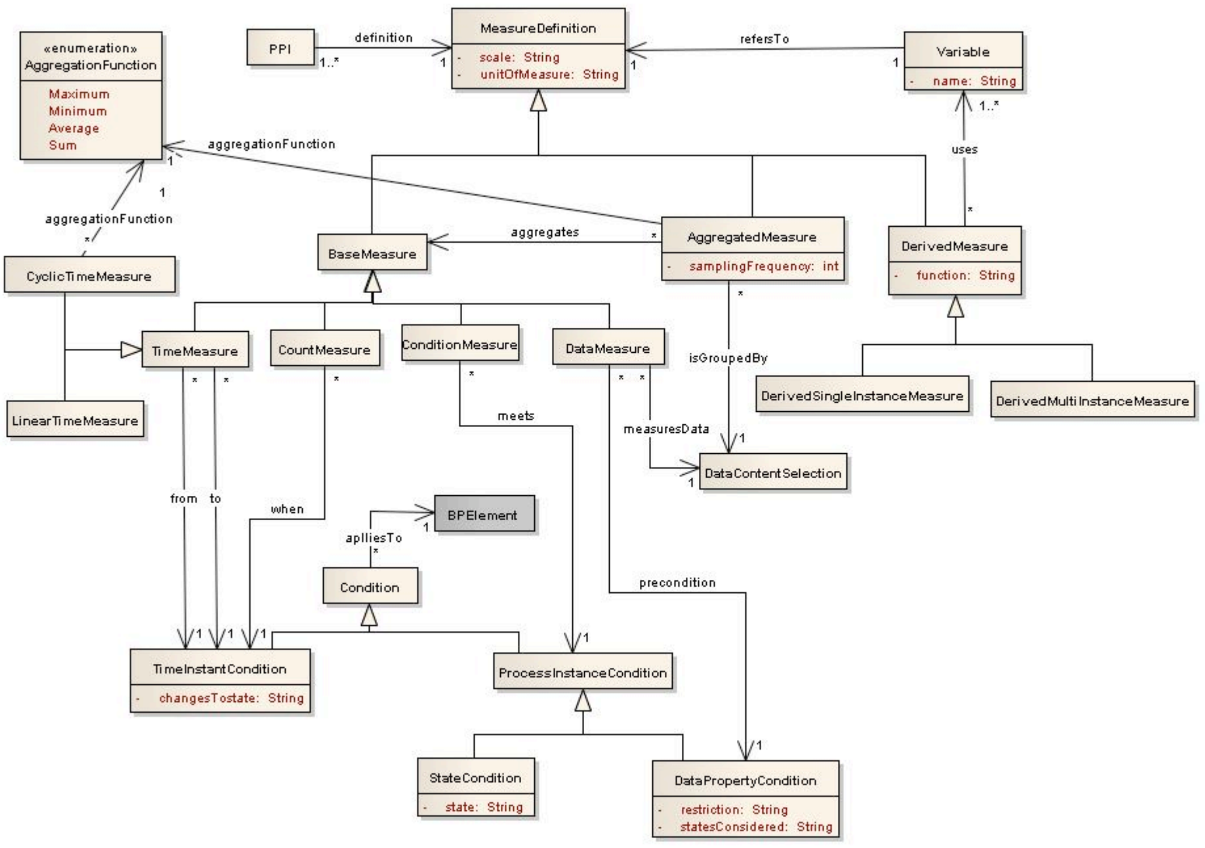


Figure 6.3: Measure definition in PPINOT Metamodel

the other four types of measures. Second, a derived multi-instance measure cannot always be defined as an aggregation of derived single-instance measures. For instance, a derived multi-instance measure such as $\frac{max(timeinanalyseincommittee)}{max(timeinprocess)}$ cannot be defined as $max(\frac{timeinanalyseincommittee}{timeinprocess})$.

In the following sections, we detail how each type of measure definition can be specified in the PPINOT metamodel.

6.4.1 Time Measure

A time measure measures the duration of time between two time instants. These two time instants can correspond with the change to a certain state of a BP element activity, pool, or dataObject, or with the triggering of a certain event. For instance, the PPI “duration of RFC analysis” can be expressed as follows:

The duration between the time instant when activity analyse RFC changes to state active and the time instant when activity analyse RFC changes to state completed.

In the PPINOT metamodel, the two time instants are represented by means of associations `from` and `to` between class `TimeMeasure` and class `TimeInstantCondition`. This latter class is used to model time instants by defining the BP element to which the condition applies (association `appliesTo`) and its change of state (attribute `changesToState`) or trigger in case of an event.

Although this definition is enough for modelling usual time measures, there is one consideration that needs to be done if the time measure is taken between elements located within a loop. In this case, two ways of measuring time can be considered, namely:

- Linear (class `LinearTimeMeasures`), in which the measure is defined taking into account the first occurrence of the time instant condition `from` and the last occurrence of the time instant condition `to`.
- Cyclic (class `CyclicTimeMeasures`), in which the measure is defined by aggregating the values for the time between the pairs of the time instant conditions of each iteration. The kind of aggregation is defined by means of attribute `aggregationFunction`

6.4.2 Count Measure

A count measure measures the number of times something happens, where the something that may happen can be the change to a certain state of a BP element or that a certain event is triggered. An example of use of a count measure for the definition of PPI "RFC successfully analysed" is:

The number of times activity analyse RFC changes to state completed.

In the PPINOT metamodel, the aforementioned "something happens" is modelled by means of association `when` between class `CountMeasure` and class `TimeInstantCondition`.

6.4.3 Condition Measure

A condition measure measures the fulfillment of certain condition in both running or finished process instances. There are two types of conditions depending on whether it is referred to the state of a BP element or to a restriction of a dataObject. An example of use of the former condition corresponds to the definition of PPI "RFC under analysis" and it is presented below:

The fulfillment of the condition activity analyse in committee is currently in state active.

Similarly, an example of use of the latter condition corresponds to the definition of PPI "RFC with priority high" and can be expressed as follows:

The fulfillment of the condition priority=high over the dataObject RFC.

In the PPINOT metamodel, the condition whose fulfillment is being measured is modelled by means of association `meets` between class `ConditionMeasure` and class `ProcessInstanceCondit` In addition, `ProcessInstanceCondition` is refined into the two types of conditions, namely `StateCondition` and `DataPropertyCondition`. In the first one the condition must include the state (attribute `state`). In the second one, the condition can include restrictions on both the content of the dataObject (attribute `restriction`) and its state (attribute `statesConsidered`). Note that the PPINOT metamodel does not prescribe any specific language for defining the restrictions on the content of the dataObject.

6.4.4 Data Measure

A data measure measures the value of a certain part of a dataObject. An example of use of this measure corresponding to the definition of PPI "information systems that an RFC affects to" is presented below:

The PPI is defined as the value of information systems of RFC.

In the PPINOT metamodel, the data measure is modelled by means of attribute `measuresData` that selects the part of a dataObject that is being measured (*e.g.* information systems in the previous example). Furthermore, attribute `precondition` allows one to specify a condition that the dataObject must fulfill when the measure is performed (for instance to be in certain state). Finally, note that the PPINOT metamodel does not prescribe any specific language for defining attribute `measuresData` since it depends on the way the dataObject is modelled. For instance, if the dataObject is an XML document, `measuresData` could be an XPath expression pointing to a specific part of the XML.

6.4.5 Derived Measure

A derived measure is defined as a mathematical function over one or more measure definitions. There are two types of derived measures depending on whether the measure definitions are single-instance measures or whether they are multi-instance measures. An example of this measure corresponding to the definition of PPI "percentage of RFCs approved from registered" is:

*The PPI is defined as the mathematical function (a/b)*100 where a is the number of times dataObject* RFC *is in state approved and b is the number of times dataObject* RFC *is in state registered.*

In the PPINOT metamodel, derived measures are modelled by means of attribute `function`, which defines the mathematical function and association `uses`, which is used to relate the variables used in `function` with their corresponding measure definitions.

6.4.6 Aggregated Measure

An aggregated measure is defined by aggregating one of the previous measures in several process instances. Furthermore, when aggregating measures it is possible to group them by the content of a certain dataObject. An example of this measure that corresponds to the definition of PPI "number of RFCs per project" is presented below:

The PPI is defined as the sum of the number of times event Receive RFC is triggered and is grouped by project of RFC.

In the PPINOT metamodel, aggregated measures are modelled using attribute `aggregationFunction`, which defines the kind of aggregation is being applied, association `aggregates`, which specifies the single-instance measure that is being aggregated, and attribute `isGroupedBy`, which may define the grouping of the measure. Finally, a sampling frequency can be defined, so that we do not need to measure every instance, but one out of X, being X the sampling frequency. This makes sense in environments where taking a measure is hard or costly (*e.g.* when the measure can not be obtained automatically).

6.5 Scope

The scope of a PPI can be seen as a filter that selects the process instances that are considered for the computation of the PPI. In particular, if the PPI is defined as a single-instance measure, the filter selects the set of instances whose value must be compared to the target value. If the PPI is defined as a multi-instance measure, the filter determines the process instances that have to be taken into account when computing the value of the PPI.

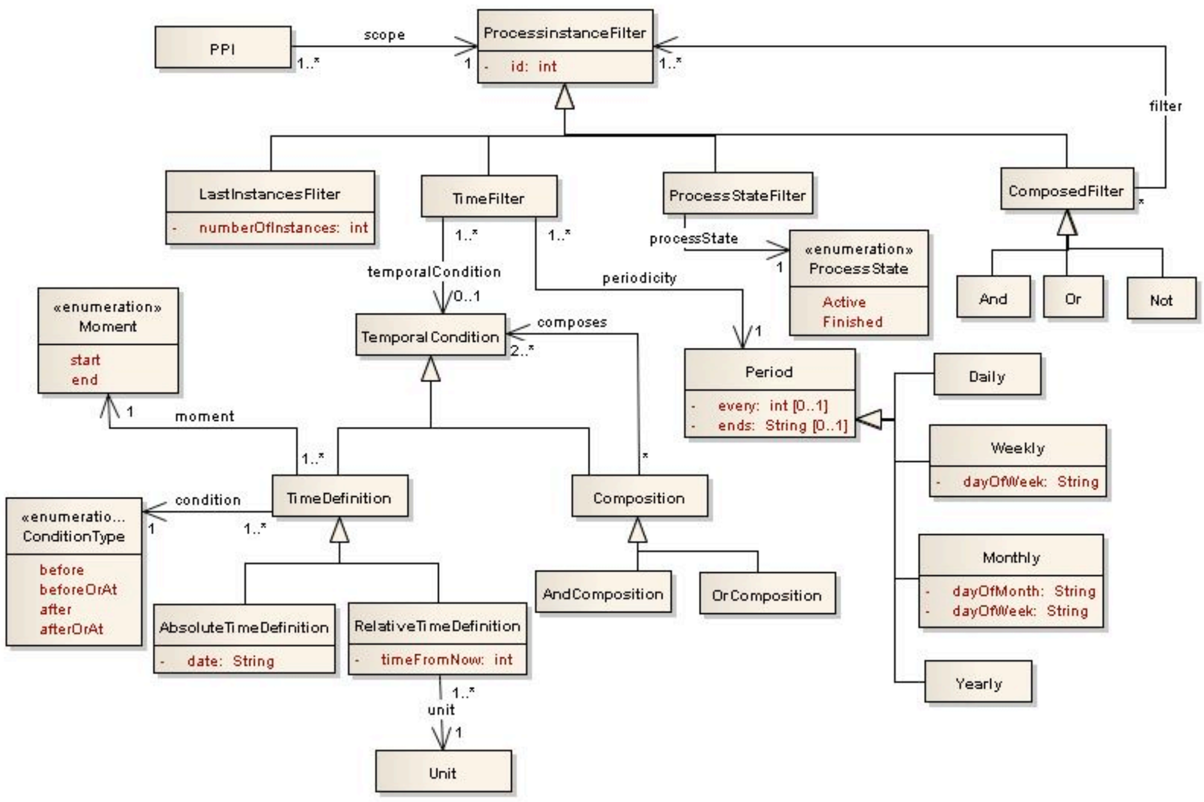


Figure 6.4: Filter definition

Since there are different types of filters that can be applied to the process instances, the PPINOT metamodel (cf. Figure §6.4) allows the definition of the scope based on:

- The last instances that have been executed (`LastInstancesFilter`)
- A temporal condition over the process instances (`TimeFilter`)
- The state of the process instances (`ProcessStateFilter`)
- Any combination of them using *and*, *or* and *not* (`ComposedFilter`).

In addition, for the particular case of the `TimeFilter`, two associations are defined. First, the temporal condition, which allows the selection of instances that started or finished “before”, “before or at”, “after” or “after or at” certain point in time. This point in time can be a concrete date or a time window defined by the time from now and the unit. And second, the periodicity, which indicates the frequency with which the PPI is calculated. As usual, there are four types of periodicity: daily, weekly, monthly and yearly. If a weekly periodicity is selected, the day of the week must be completed, and for the case of monthly periodicity, whether to take into account the day of the month (e.g. 3rd January) or the day of the week (e.g. third tuesday of the month) must be selected. The frequency of such periodicity (e.g. every 2 months, for a monthly periodicity) and when to finish taking such measure (ends 31-12-2014) can also be specified.

6.6 SUMMARY

In this chapter, we have presented PPINOT metamodel, that allows to define PPIs and their relationship with the BP elements. This mechanism to define PPIs is expressive enough to allow the definition of a wide range of PPIs, including PPIs not supported by existing approaches (those related to data objects or using derived measures, or even those with a very restrictive analysis period). We summarise the problems we overcome with this contribution by means of the Kiviat diagram depicted in Figure §6.5.

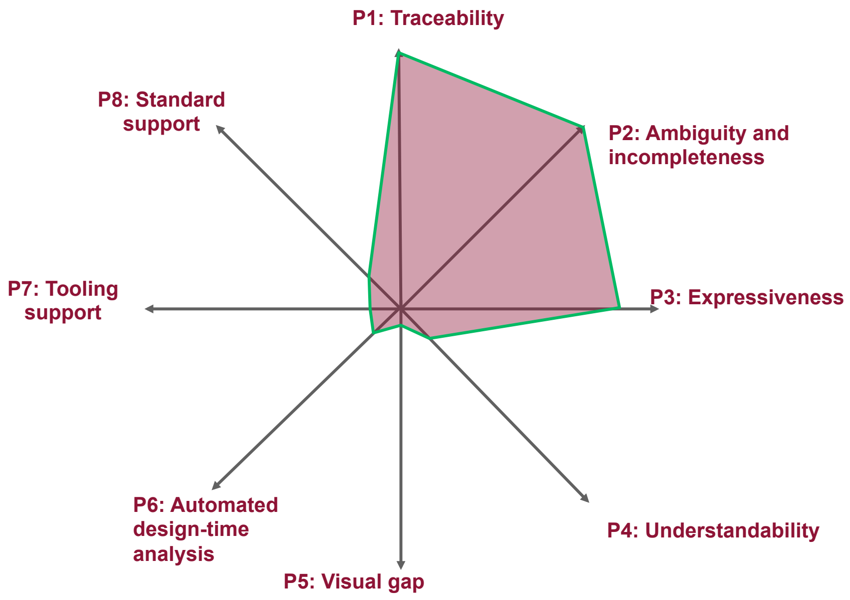


Figure 6.5: Kiviat diagram for the problems overcome by PPINOT Metamodel

PPINOT Graphical Notation

I prefer drawing that talking; drawing is faster, and leaves less room for lies.

Le Corbusier (1887–1965),
French architect, designer, urbanist and writer

This chapter is devoted to presenting a graphical notation to depict PPIs based on the metamodel presented in the previous chapter. This notation has been called PPINOT graphical notation and aims at providing the understandability property required to PPIs. Furthermore it allows to provide a comprehensive view of BP models and PPI definitions, making thus explicit the traceability between bot of them and facilitating to maintain the coherence across them. The chapter is organised as follows: Section §7.1 introduces the subject; in Section §7.2, the constructs of PPINOT graphical notation, including nodes and connectors, as well as the connection rules are described. We provide information regarding the notation design rationale in Section §7.3; then in Section §7.4 a set of guidelines in order to assist the user to define PPIs using PPINOT graphical notations are outlined; finalyy, we summarise the chapter in Section §7.5.

7.1 Introduction

There exists a partial view from the different departments or roles in charge of, on the one hand, the modelling and execution of business processes, and, on the other hand, the definition and consecution of goals and its associated indicators: nobody has a comprehensive view of both worlds, making thus very difficult the maintenance of coherence across them. In Chapter §6 we have presented a metamodel for the definition of PPIs over BPs; however, there exists a visual gap between BP models and this model of PPIs. In the same manner as explicit BP models expressed in a graphical notation (such as, for instance, BPMN) ease communication about these processes, allowing thus stakeholders to communicate efficiently, and refine and improve them; a graphical notation that allows the depiction of PPIs together with the corresponding BP models will also bring this benefit together with this comprehensive view, absent up to know.

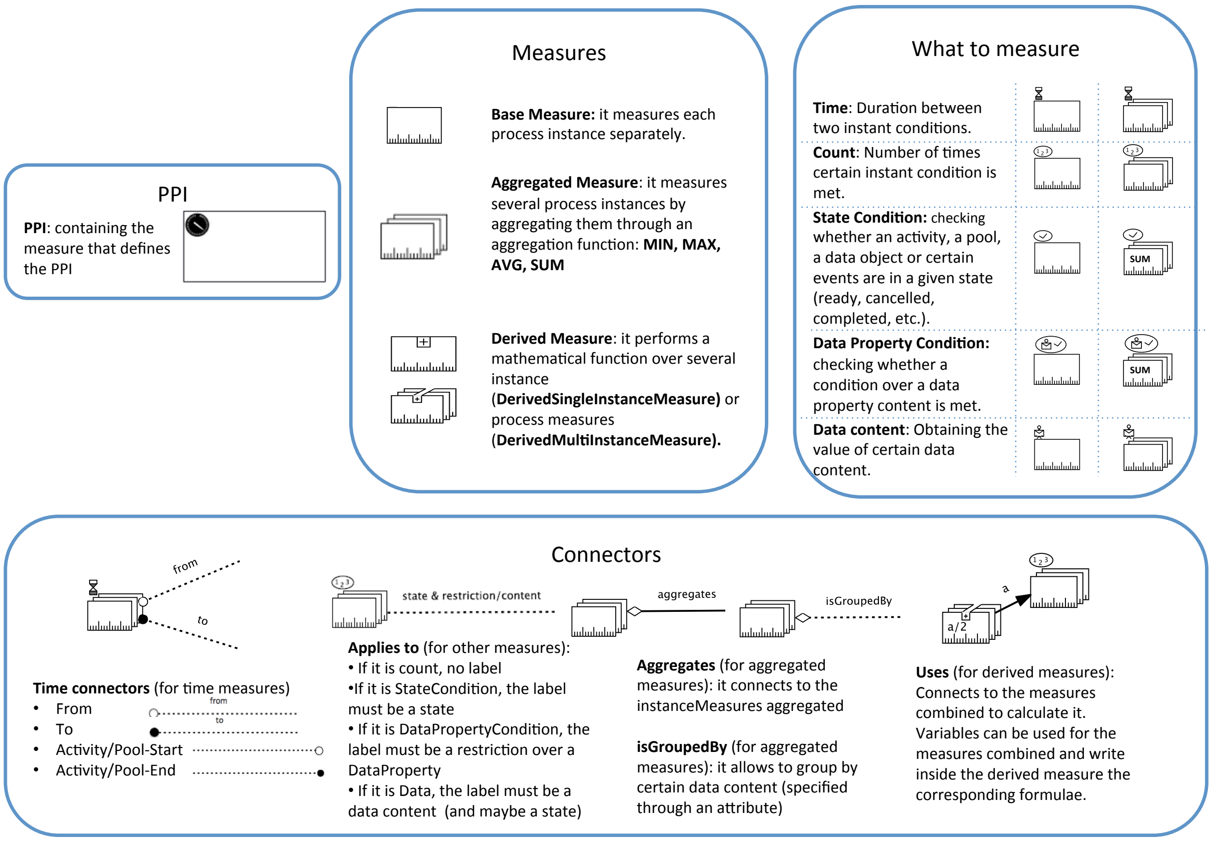
In this chapter, we present PPINOT graphical notation, that allows the depiction of PPIs over BPMN *Business Process Diagrams (BPDs)*, based on PPNIOT metamodel, described in Chapter §6. It is a graph-based notation defined attending to a set of requirements and principles for designing cognitively effective visual notations established by several authors. We also provide a simple guide to assist the user in defining PPIs using PPINOT graphical notation

7.2 Notation Constructs

Figure §7.1 shows an overview of our proposed notation. Connections with BPMN elements are depicted by links, which can be decorated with a label giving the link type (e.g."from", "isGroupedBy"). In the following subsections we present each type of construct in our notation and the corresponding connectors that allows to link them with the BPMN diagram and to establish some relations between them.

7.2.1 PPI

First we present the construct for the PPI, that will be in most cases the container of the other constructs. A PPI is depicted by a rectangle, having on the upper left corner a circle representing an indicator (as represented in Figure §7.1). Its attribute `name` will appear on top of this shape, in the middle. Within that shape, the measure that defines the PPI is placed (sometimes there will be only one measure, in case of `BaseMeasures` and "simplified" `AggregatedMeasures` (see Section §7.2.3 for more details), and sometimes, more than one, connected trhough links `uses`, in case of the rest of `AggregatedMeasures` and `DerivedMeasures`).



PPI

PPI: containing the measure that defines the PPI

Measures

Base Measure: it measures each process instance separately.

Aggregated Measure: it measures several process instances by aggregating them through an aggregation function: **MIN, MAX, AVG, SUM**

Derived Measure: it performs a mathematical function over several instance (**DerivedSingleInstanceMeasure)** or process measures (**DerivedMultiInstanceMeasure).**

What to measure

Time: Duration between two instant conditions.

Count: Number of times certain instant condition is met.

State Condition: checking whether an activity, a pool, a data object or certain events are in a given state (ready, cancelled, completed, etc.).

Data Property Condition: checking whether a condition over a data property content is met.

Data content: Obtaining the value of certain data content.

Connectors

Time connectors (for time measures)
- From
- To
- Activity/Pool-Start
- Activity/Pool-End

Applies to (for other measures):
• If it is count, no label
•If it is StateCondition, the label must be a state
• If it is DataPropertyCondition, the label must be a restriction over a DataProperty
• If it is Data, the label must be a data content (and maybe a state)

Aggregates (for aggregated measures): it connects to the instanceMeasures aggregated

isGroupedBy (for aggregated measures): it allows to group by certain data content (specified through an attribute)

Uses (for derived measures): Connects to the measures combined to calculate it. Variables can be used for the measures combined and write inside the derived measure the corresponding formulae.

Figure 7.1: PPINOT Graphical Notation (overview)

The element `PPI` has the set of attributes previously described in Chapter §6 and are summarised in Table §7.1. These attributes are not represented graphically

We take PPI9 "process duration" as example of PPI. It is depicted in Figure §7.2

7.2.2 BaseMeasure

The `BaseMeasure` element has the set of attributes depicted in table §7.2. Depending on what to be measured, there are four types of `BaseMeasures`. In each case, a different icon is added on the upper left corner to the `BaseMeasure` shape. In the following we list them:

- `TimeMeasure`

  In this case, where time is measured, the icon added is an hourglass. There aare two links to the BPMN diagram (BPD). They are called `Time connectors` (see §7.2.5 for more detail).

  The element `TimeMeasure` inherits the attributes of `BaseMeasure` (see Table §7.2). Table §7.3 presents the additional attributes of the `TimeMeasure` element.

Attribute Name	**Description/Usage**
identifier: string	To uniquely identify the PPI
name: string	Descriptive name of the PPI
goals: string	To establish strategic or operational goals
target: string	Set the restriction which is the objective to achieve
scope: string	To indicate the subset of instances taken into accout to calculate the PPI value
responsible: HumanResource	Human resource in charge of the PPI
informed: HumanResource [0..*]	Human resources interested in the PPI
comments: string	other information to add

Table 7.1: Attributes of PPIs

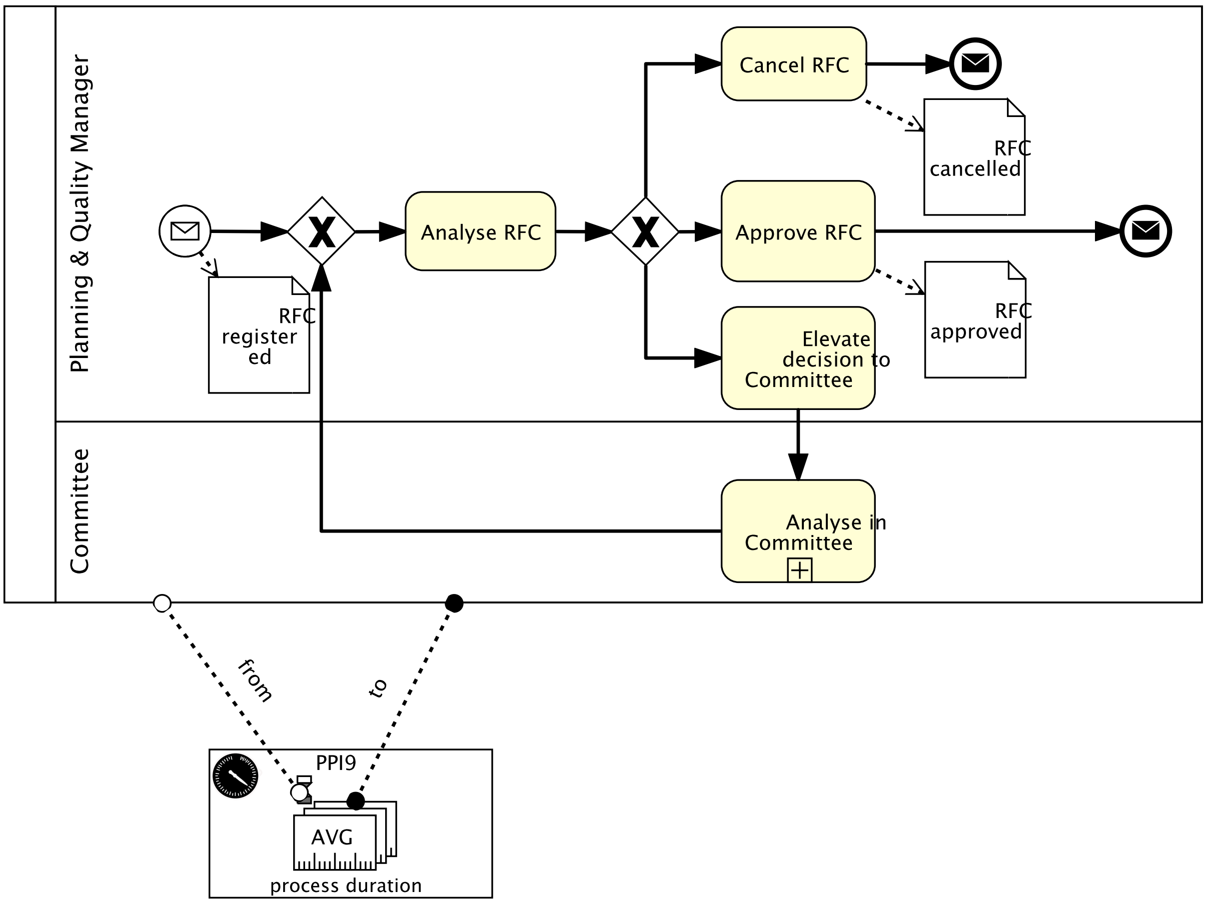


Figure 7.2: PPI example

Attribute Name	Description/Usage
name: string	The name of the `BaseMeasure`
scale: string	This attribute identifies the domain for the measure, i.e. a set of values with defined properties, e.g. natural, integer, float, map, [0..100]
unitOfMeasure: string	The unit of the `BaseMeasure`, e.g. seconds, hours or euros

Table 7.2: Attributes of `BaseMeasures`

Attribute Name	Description/Usage
timeMeasuretype: string = LinearTimeMeasure {LinearTimeMeasure | CyclicTimeMeasure}	This attribute allows to dedifferentiate between `LinearTimeMeasures` and `CyclicTimeMeasures`.
singleInstanceAggFunction: string[0..1] newline{avg | max | min | sum}	this attribute defines the aggregation function applied if `timeMeasuretype` = `CyclicTimeMeasure`

Table 7.3: Attributes of `TimeMeasures`

As explained in Section §6.4.1 `TimeMeasures` can be subdivided into `LinearTimeMeasures` and `Cyclic- TimeMeasures`; graphically, the difference between them is that the `CyclicTimeMeasure` has a symbol representing a loop (different from the one of BPMN) on the right of the hourglass icon.

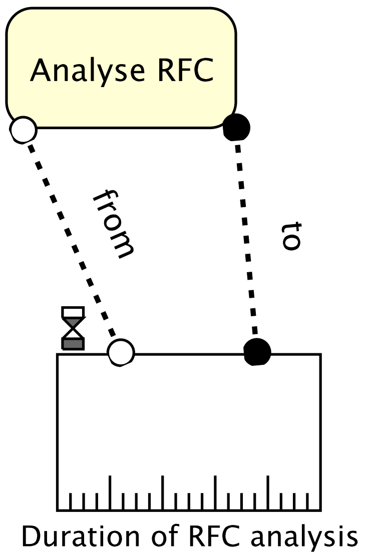


Figure 7.3: Linear time measure example

From the `TimeMeasure` example "duration of the analysis of an RFC", we can distin-

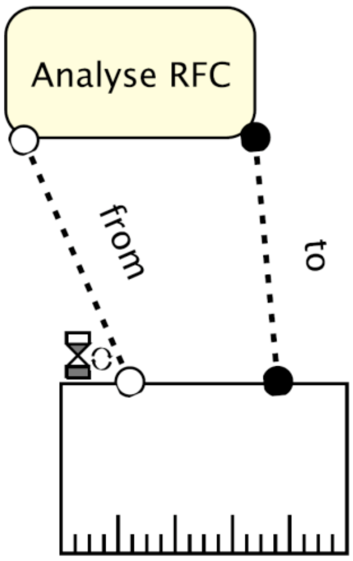


Figure 7.4: Cyclic time measure example

guish between the `LinearTimeMeasure`, depicted in Figure §7.3, and the `Cyclic-TimeMeasure`, depicted in Figure §7.4 (in this case, the value of attribute `singleInstance-AggFunction` must be fulfilled, *e.g.* avg for the PPI "average duration of the analysis of an RFC in a process instance").

- `CountMeasure` This measure counts, therefore, the icon selected to represent it is an ellipse with the numbers 1, 2 and 3 inside (as shown in Figure §7.1). The `CountMeasure`'s shape is connected to the BPD through a connector called `appliesTo` (see Subsection §7.2.6 for further explanation). This connector will go to the flow element whose condition is being counted.

  The element `CountMeasure` inherits the attributes of `BaseMeasure` (see Table §7.2).

  An example of `CountMeasure` is "number of times an RFC is successfully analysed" (depicted in Figure §7.5).

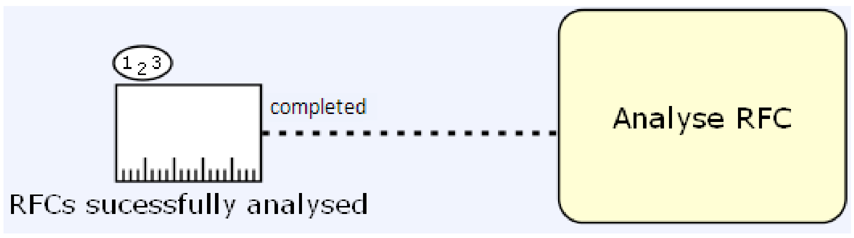


Figure 7.5: Count measure example

- `ConditionMeasure`

This type of measure checks certain condition, therefore, the icon added to the upper left corner of the BaseMeasure's shape is an ellipse with the checklist symbol inside (as shown in Figure §7.1). This measure is connected to the BPD, as in the previous case, through the connector appliesTo. The element ConditionMeasure inherits the attributes of BaseMeasure (see Table §7.2).

As detailed in Section §6.4.3, ConditionMeasures can be subdivided into StateCondition and DataPropertyCondition. Graphically, the difference between them is that in the case of DataPropertyCondition, there is an envelope before the checklist symbol inside the ellipse.

The StateConditionMeasure example "check if RFC under analysis" is depicted in Figure §7.6.

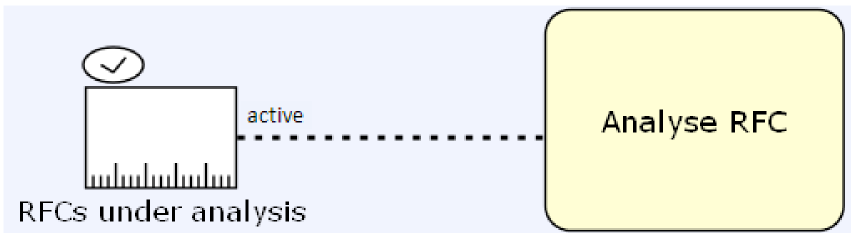


Figure 7.6: State condition measure example

The DataPropertyConditionMeasure example "check if RFC with priority high" is depicted in Figure §7.7.

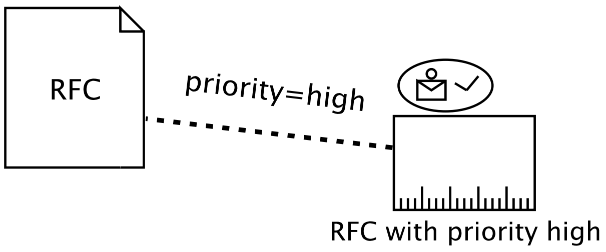


Figure 7.7: DataProperty condition measure example

- DataMeasure

  In this case, where the value of certain part of a dataObject is taken, the icon added is a stick figure carrying an envelope (see the icon in Figure §7.1). Again, as in the previous cases, the connection to the BPD is through the connector appliesTo. The element DataMeasure inherits the attributes of BaseMeasure (see Table §7.2).

  The DataMeasure example "number of Information Systems an RFC affects to" is depicted in Figure §7.7.

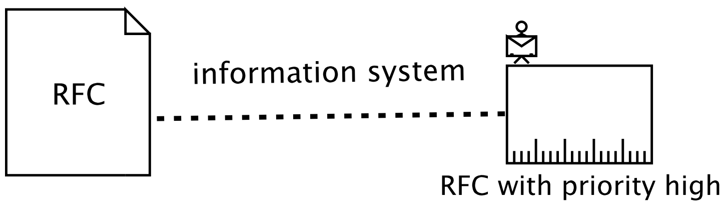


Figure 7.8: Data measure example

7.2.3 AggregatedMeasure

The `AggregatedMeasure`'s shape is obtained by superimposing three shapes of the `BaseMeasure` as depicted in Figure §7.1. Inside the front rectangle (ruler) of this shape, the value of attribute `AggregationFunction` is written ("Min","Max", "Avg" or "Sum").

This `AggregatedMeasure` must be related to the `SingleInstanceMeasure` it aggregates; this relationship is established through a connector called `aggregates` (see subsection §7.2.7 for further explanation).

The element `AggregatedMeasure` has the set of attributes depicted in Table §7.4.

Attribute Name	**Description/Usage**
name: string	The name of the `AggregatedMeasure`
scale: string	This attribute identifies the domain for the measure, i.e. a set of values with defined properties, e.g. natural, integer, float, map, [0..100]
unitOfMeasure: string	The unit of the `AggregatedMeasure`, e.g. seconds, hours or euros
aggregationFunction: string[0..1] { avg | max | min | sum }	This attribute defines the aggregation function applied to the `BaseMeasure` aggregated in the `AggregatedMeasure`

Table 7.4: Attributes of `AggregatedMeasures`

The `AggregatedMeasure` example " number of RFCs rejected in the last year" is depicted in Figure §7.9.

When the `SingleInstanceMeasure` aggregated is a `BaseMeasure`, then, this connection can be omitted and the `AggregatedMeasure` inherits its links to the BPD as well as its type (time-cyclic or linear-, count, stateCondition, dataPropertyCondition or data); in such a

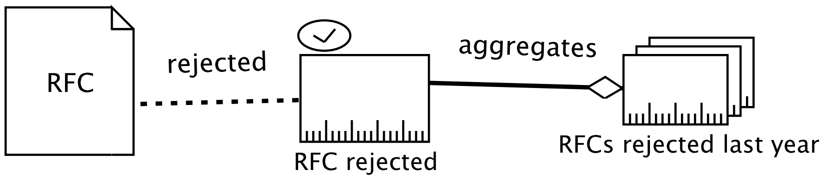


Figure 7.9: Aggregated measure example

case, the corresponding icon will be added again on the upper left corner of the `AggregatedMeasure` shape and they take the name `TimeAggregatedMeasure`, `CountAggregatedMeasure`, `StateConditionAggregatedMeasure`, `DataPropertyConditionAggregatedMeasure` and `DataAggregatedMeasure` respectively. Figure §7.10 shows the previous `AggregatedMeasure` example depicted following this simplification (a `StateConditionAggregatedMeasure` is depicted in this case).

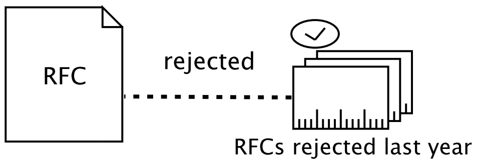


Figure 7.10: Aggregated measure example

For the case of `TimeAggregatedMeasures` the set of attributes of `TimeMeasures` are inherited (see Table §7.3).

7.2.4 DerivedMeasure

As stated in Section §6.4.5, `DerivedMeasures` can be subdivided into **`Derived-SingleInstanceMeasures`** and **`DerivedMultiInstanceMeasure`**.

In either cases, there will be as many links as measures combined to calculate the value. These connectors are called `uses` (see §7.2.9 for more details).

There exists the possibility to add a formula with the corresponding mathematical function that allows to calculate the value of the `DerivedMeasure`. It must be written inside its shape, using variables representing the combined measures.

The set of attributes of `DerivedMeasures` are depicted in Table §7.5.

The shapes are different according to the previous mentioned subdivision.

- `DerivedInstanceMeasure`

Attribute Name	Description/Usage
name: string	The name of the DerivedMeasure
scale: string	This attribute identifies the domain for the DerivedMeasure, i.e. a set of values with defined properties, e.g. natural, integer, float, map, [0..100]
unitOfMeasure: string	The unit of the DerivedMeasure, e.g. seconds, hours or euros
function: string[0..1]	This attributes identifies the mathematical function applied to calculate the value of the DerivedMeasure

Table 7.5: Attributes of DerivedMeasures

The shape depicting this DerivedInstanceMeasure is similar to the one used for the BaseMeasure, but the ruler has up and in the middle a square with a plus symbol inside, trying to convey the need to perform a mathematical function to obtain the value of this measure (see Figure §7.1)

An example of DerivedInstanceMeasure is "percentage of time spent in activity *analyse RFC* with respect to the duration of the process ", and is depicted in Figure §7.11.

- DerivedProcessMeasure

  This case is very similar to the previous one, but now, since these are ProcessMeasures, the depiction is obtained by superimposing three of the previous shapes (the ones for the DerivedInstanceMeasure).

  An example of DerivedProcessMeasure is "percentage of approved RFCs with respect to all the registered RFCs", and is depicted in Figure §7.12.

7.2.5 Time Connector

A Time Connector is used to represent the point in the process where time starts and ends to be measured. Each Time Connector has only one source and one target. The source must be a TimeMeasure or an AggregatedTimeMeasure, and the target must be one of the following elements: activity, pool, event or dataObject.

A Time Connector is depicted through a dashed line. Depending on the value of the attribute conditionType, it acquires different forms: if conditionType = from, it is decorated with the label "from" and an empty circle at the initial end, and it will be connected to the flow element where the time starts to run. If conditionType = to it is decorated

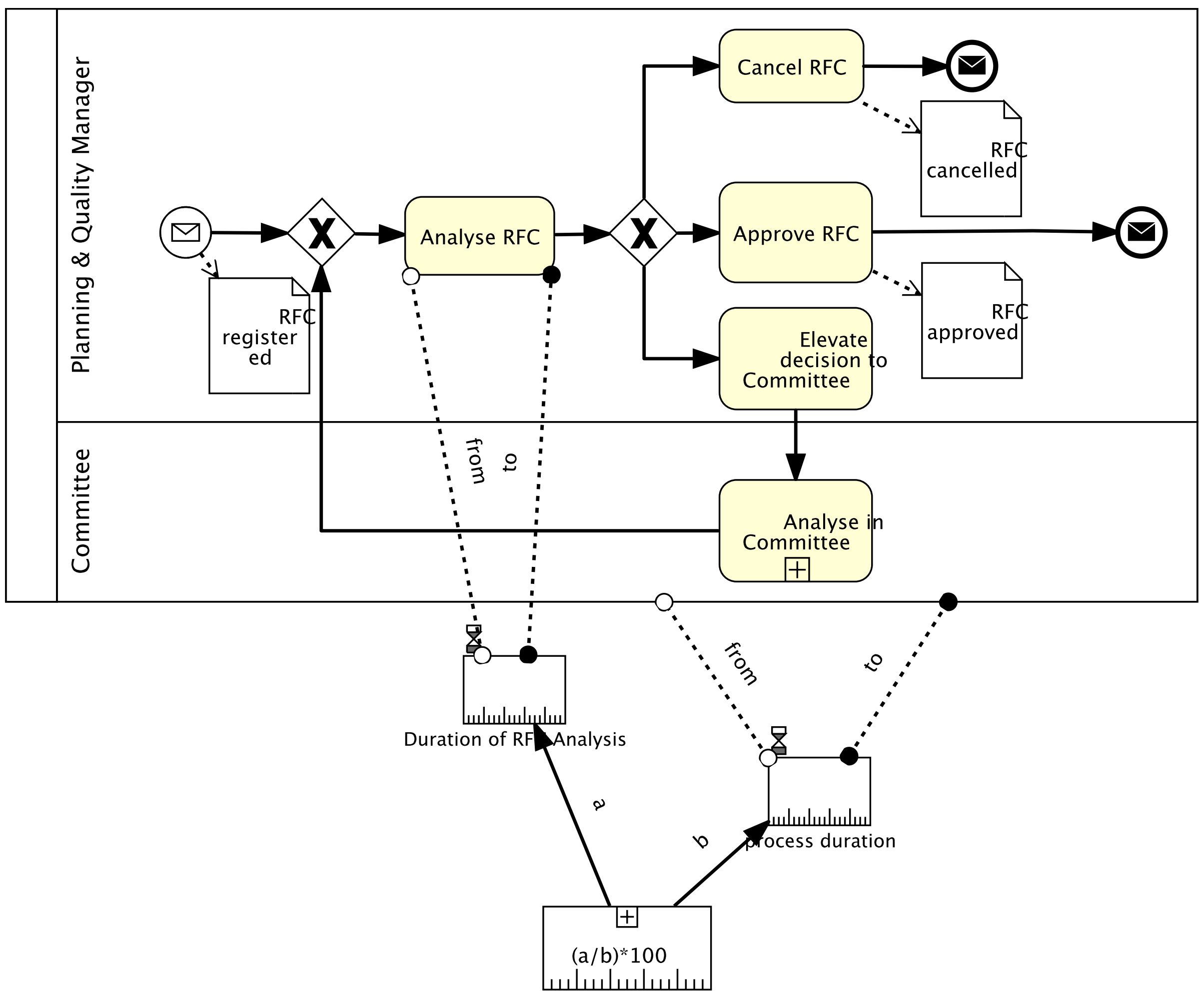


Figure 7.11: Derived instance measure example

with the label "to" and a filled circle at the initial end, and will be connected to the flow element that closes the measured time period.

If a `Time Connector`'s target is the start of an activity or a pool, an empty circle must be attached to the final end (the part connected to the flow element) and the label "start" appears above the connector, near to the target; if, on the contrary, the `Time Connector`'s target is the end [1] of an activity or a pool, a filled circle has to be added to the final end of the link and the label "end" appears above the connector, near to the target. If any other state is indicated in attribute `state`, its value appears above the `Time Connector` near to the target element.

There must be only two `Time Connectors` per `TimeMeasure` or `AggregatedTimeMeasure`, one `from` and one `to`, except for the case where the target is a dataObject, that there can be

[1] with end we refer to an activity or a pool that has reached one of the following states: completed, compensated, failed and terminated

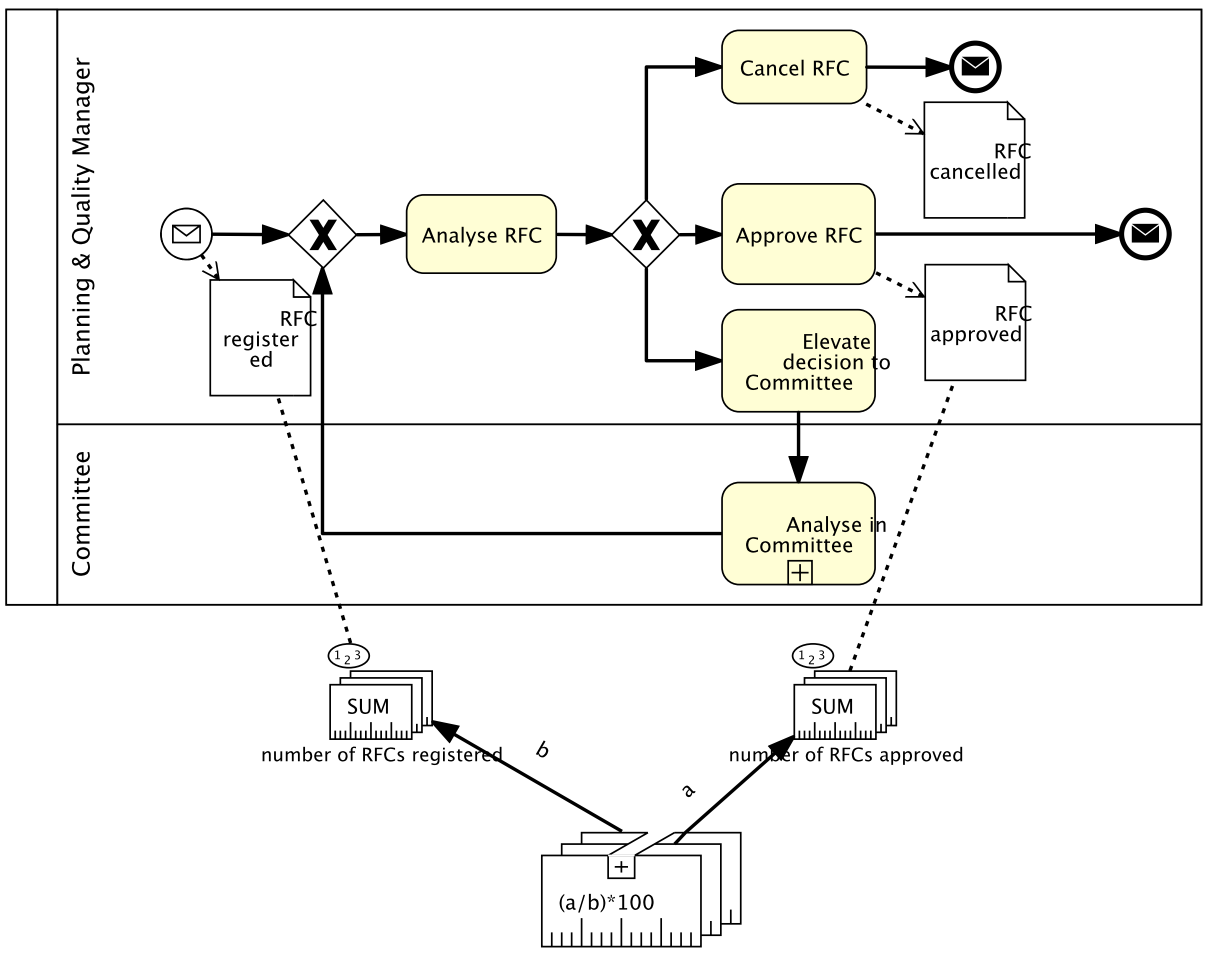


Figure 7.12: Derived process measure example

more than one `from` or more than one `to` connecting to the same dataObject, but with different states.

Some examples of `Time Connectors` are depicted in Figures §7.3 , §7.4 , §7.11, and §7.2.

7.2.6 Applies To

An `Applies To` is used to establish the connection between any `BaseMeasure` (except `TimeMeasure`) or any `AggregatedMeasure` (except `TimeAggregatedMeasure`) and an element of the business process. It is depicted through a dashed line. The label of this connector changes depending on the kind of measure that acts as a source:

Attribute Name	**Description/Usage**
conditionType: string = from { from | to }	This attribute is used to specify if the `Time Connector` refers to the start or end point of the `TimeMeasure/TimeAggregatedMeasure` connected
when: string { start | end | other}	This attribute is used to specify if the `Time Connector` points to the start or end of an activity or a pool
state: string	This attribute identifies the state the target BP element of the `Time Connector` must change to, If it is different from start or end (corresponds to the changesToState in PPINOT metamodel)

Table 7.6: Attributes of `Time Connectors`

`CountMeasure` or `CountAggregatedMeasure` In this case, the target can be an activity, a pool, an event or a dataObject. The same situation described in §7.2.5 related to the start or the end of activities and pools can take place, and the way to proceed is exactly the same (empty circle at the final end of the connector and the label "start" near to the target for the start case, and filled circle at the final end of the connector and the label "end" near to the target for the end case; if any other state is indicated in attribute `state`, its value appears above the `Applies To Connector` near to the target element). There must be only one source and one target per `Applies To`. There must be also only one `Applies To` per `CountMeasure`, except for the case where the target is a dataObject, that there can be more than one `Applies To` connecting to the same dataObject, but with different states. An example of this case of `Applies To` is depicted in Figure §7.5.

`StateConditionMeasure` or `StateConditionAggregatedMeasure` In this case, the target can be an activity, a pool, an event or a dataObject. The label must be the value of the attribute `state`, and appears above the connector, near to the target. There must be only one source and one target. Furthermore there must be also only one `Applies To` per `CountMeasure/CountAggregatedMeasure`. Two examples of this case of `Applies To` are depicted in Figure §7.6 and Figure §7.10.

`DataPropertyConditionMeasure` or `DataPropertyConditionAggregatedMeasure` In this case the target must be a dataObject. Several label's combinations are possible. One label with the value of attribute `restriction` must appear under the connector, near to the target. Furthermore, if the user wants to specify a `state` for the dataObejct to be measured, its value must appear above the connector, near to the target. There must be only one source and one target per `Applies To`. There can be more than one `Applies To` per `DataPropertyConditionMeasure/DataPropertyConditionAggregatedMeasure`, but always connecting to the same dataObject with different states. An example of this case of `Applies To` is depicted in Figure §7.7.

`DataMeasure` or `DataAggregatedMeasure` In this case the target must be a dataObject. Several label's combinations are possible. One label with the value of attribute `measuresData` must appear under the connector, near to the target. Furthermore, if the user wants to specify a `state` for the dataObejct to be measured, its value will appear above the connector, near to the target. There must be only one source and one target per `Applies To`. There can be more than one `Applies To` per `DataMeasure`/ `DataAggregatedMeasure`, but always connecting to the same dataObject with different states. An example of this case of `Applies To` is depicted in Figure §7.8.

The set of attributes of the element `Applies To` are shown in Table §7.7

7.2.7 Aggregates

An `Aggregates` connector is used to establish the relationship between an `AggregatedMeasure` and the `InstanceMeasure` it aggregates. Thus, there must be only one source and one target per `Aggregates`, and the source must be an `AggregatedMeasure` and the target must be an `InstanceMeasure`.

It is depicted by a solid line with an empty diamond at the initial end (the part connected to the source, an `AggregatedMeasure`).

The element `Aggregates` has no attributes.

An example of `Aggregates` is depicted in Figure §7.9

Attribute Name	**Description/Usage**
when: string { start | end | other}	This attribute is used to specify if the `Applies To Connector` points to the start or end of an activity or a pool
state: string	This attribute identifies the state the target BP element of the `Applies To Connector` must be in, If it is different from start or end
restriction: string	This attributes is used to specify the restriction a Dataproperty must fulfill if the source of the `Applies To Connector` is a `DataPropertyConditionMeasure` or a `DataPropertyConditionAggregatedMeasure`
measuresData: string	This attributes is used to specify the data content whose value is obtained when the source of the `Applies To Connector` is a `DataMeasure` or a `DataAggregatedMeasure`

Table 7.7: Attributes of `Applies To`

7.2.8 IsGroupedBy

An `isGroupedBy` connector is used to group `AggregatedMeasures` by certain data content. There must be only one source and one target per `IsGroupedBy`, and the source must be an `AggregatedMeasure` and the target must be a dataObject.

This connector is depicted by a dashed line with an empty diamond at the initial end (the part connected to the source- an `AggregatedMeasure`). It has an “isGroupedBy” label above it (in the middle), and a label with the value of attribute `dataProperty` must appear above the connector, near to the target. A label with the value of attribute `state` can appear under the connector, near to the target, to indicate the sate the DataObject must be in to be taken into account. There can be more that one isGroupedBy per `AggregatedMeasure` whenever the target is always the same dataObject but with different states.

Table §7.8 represents the set of attributes of the `isGroupedBy` connector.

Figure §7.13 shows an example of `isGroupedBy`.

Attribute Name	**Description/Usage**
state: string	
dataProperty: string	This attribute is used to specify the property of the Dataobject used to group the `AggregatedMeasure` that acts for source in the `IsGroupedBy Connector`

Table 7.8: Attributes of `IsGroupedBy`

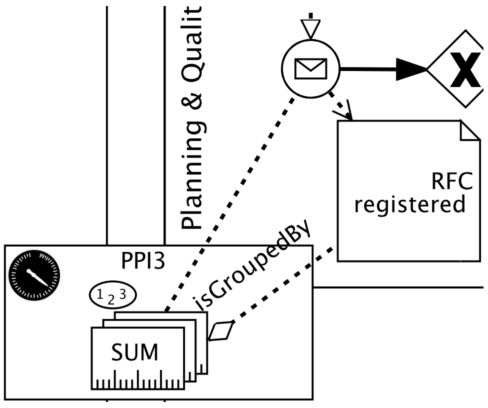


Figure 7.13: IsGropuedBy example corresponding to the `CountAggregatedMeasure` "number of RFC registered per project"

7.2.9 Uses

An `Uses` connector allows to represent the `MeasureDefinitions` used to calculate a `DerivedMeasure`. Thus, there must be only one source (always a `DerivedMeasure`) and there can be several targets (`SingleInstanceMeasures` if the source is a `DerivedSingle-InstanceMeasure` and `MultiInstanceMeasures` if the source is a `DerivedMulti-InstanceMeasure`).

`Uses` connector's depiction is a solid arrow. A label with the value of attribute `variable` can appears above the connector, in the middle. This value makes reference to the variables used in the formula (`function` specified in the `DerivedMeasure`).

Table §7.9 sums up the attributes of the connector `Uses`.

Figure §7.11 and Figure §7.12 show examples of `Uses`.

Attribute Name	**Description/Usage**
variable: string[0..1]	This attribute identifies each of the `MeasureDefinitions` used to calculate the value of the `DerivedMeasure` that acts for source of the `Uses Connector`. This `MeasureDefinitions` are referred to by means of the variables used in the `function` of the above mentioned `DerivedMeasure`

Table 7.9: Attributes of `Uses`

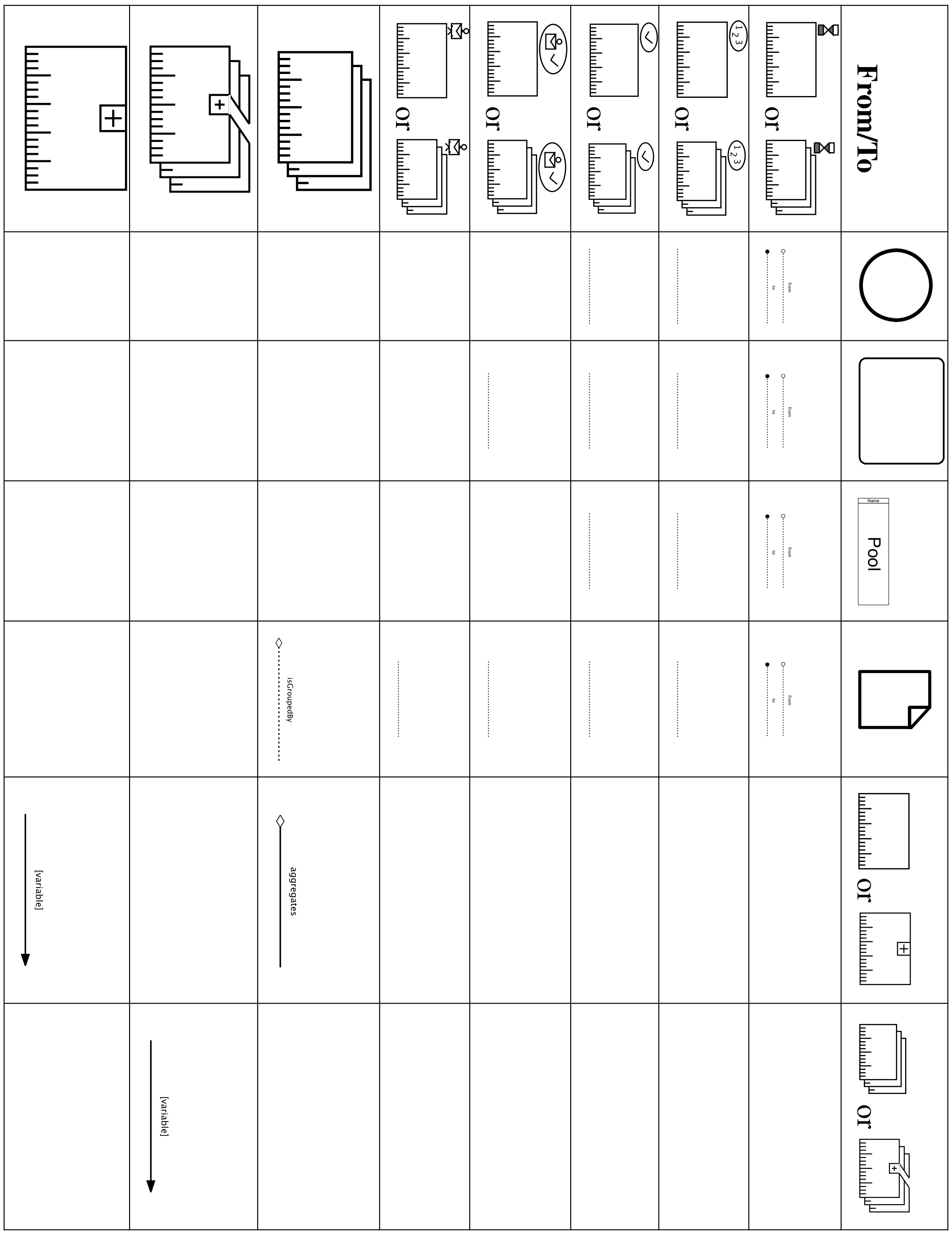
7.2.10 Connection Rules

Table §7.10 displays the different `MeasureDefinitions` and the BP elements and shows how these objects can connect to one another through the presented `Connectors`. The connector symbol in each cell indicates that the object listed in the row can connect to the object listed in the column. This table summarizes the information presented in the previous subsections (from Subsectiion §7.2.2 to Subsection §7.2.9.

7.3 Design Rationale

Having presented the constructs of our graphical notation in Section §7.2, we now turn to explain the design decisions we took when defining our graphical notation. As stated before, we defined it attending to certain requirements or best practices established by several authors. Concretely we took into account the twelve requirements set by Rumbaugh in [85] and the set of nine principles for designing cognitively effective visual notations presented by Moody in [56]. Furthermore, we seek to develop a graphical notation consistent with BPMN 2.0 [68], since it is conceived as an extension of BPMN and intended to be used together with it (in order to define PPIs together with the corresponding business process modelled in BPMN). Although we tried to fulfill all of them, sometimes it was difficult because there exist contradictions between the various authors. Tables §7.11 and §7.12 shows the correspondence between Rumbaugh's requirements and Moody's principles, when possible, as well as their fulfillment by our notation. We use the following notation: A ✓ sign means that our notation successfully addresses the issue (principle or requirement); a $\sim$ sign indicates that it addresses it partially; a ✗ sign indicates that it does not contemplate the issue; and N/A means the issue is not applicable to our proposal.

In order to satisfy the first two rows (`semiotic clarity` means that there should be a 1:1 correspondence between semantic constructs and graphical symbols, and `graphic`



From/To
or
Pool
Name
from
to
isGroupedBy
aggregates
[variable]

Table 7.10: PPINOT Connection Rules

	Rumbaugh [85]	**Moody [56]**	**PPINot**	**Comments**
1	Clear mapping concepts to symbols	Semiotic Clarity	✓	Deliberate symbols deficit
2	No overloading of symbols	Graphic Economy	✓	We leave concepts off-diagram to simplify
3	Uniform mapping concepts to symbols	Perceptual Discriminability	✓	
4	Distinction not too subtle			
5	Easy to draw by hand	-	∼	Some symbols not so easy to draw but intuitive
6	-	Semantic Transparency	✓	
7	Looks good when printed	-	✓	
8	Must fax & copy well. No colours	-	✓	

Table 7.11: Compliance with requirements and principles by PPINOT notation (I)

`economy` means that the number of different graphical symbols should be cognitively manageable), in our notation, each of the key concepts (PPI, each type of`BaseMeasure`, `AgregatedMeasures`, `DerivedMeasure`, etc) was mapped to a distinct symbol. We only left without symbols those concepts that, not being crucial for the understandability of the depictions, could raise the complexity of it (for instance the scope).

Regarding the third row, taking into account that `perceptual discriminability` refers to the fact that different symbols should be clearly distinguishable from each other; we strived to select symbols that emphasize similarities between related concepts (e.g. between `DerivedSingleInstanceMeasure` and `DerivedMultiInstanceMeasure`), whilst using clearly distinct symbols for concepts clearly dissimilar (e.g. symbols for the different types of `BaseMeasures`, time, data, etc.).

In order to accomplish the principle of `semantic transparency` (row 6), which aims to use visual representations whose appearance suggests their meaning, we provide a correspondence between the icons selected and the concepts they represent (e.g a ruler to represent measures). Most of these icons are also easily drawn by hand, fulfilling thus the principle of

	Rumbaugh [85]	Moody [56]	PPINot	Comments
9	-	Visual Expressiveness	∼	Among the 8 possible visual variables, we only use shape and texture
10	Consistent with past practice	-	N/A	No past practice
11	Self consistent	-	✓	
12	-	Dual Coding	✓	Text complements graphics
13	Users can remember it	-	∼	Subjective requirements Need for an experiment to prove them
14	Common cases appear simple	-	∼	
15	Suppressible details	Manageable Complexity	∼	Different levels of abstraction for PPIs
16	-	Cognitive Fit	✗	Only one dialect simple enough to everyone and every media
17	-	Cognitive Integration	N/A	Only one type of diagram

Table 7.12: Compliance with requirements and principles by PPINOT notation (II)

Rumbaugh in row 5, but in some cases (*e.g.* the `DerivedMeasure`), it is not so like that; in these cases, the decision was made in favour of compliance with the semantic transparency principle.

Our notation does not rely on shading, line thickness, colours, or any other distinction that are subtle, confusing or that do not fax/copy well. With these decisions, we fulfill criteria from rows 4, 5, 7 and 8 of Rumbaugh, but we leave partially uncovered the principle of `visual expressiveness` from row 9 (use the full range and capacities of visual variables). We considered it could be useful to allow the user to use these mechanisms (colours, shading, etc.) to emphasize concepts of the business domain, not of the notation domain (following also the BPMN recommendations).

Related to the consistency with past practice (row 10), since, to the best of our knowledge, there is no graphical notation for this role, many of the concepts used do not exist and is important to have new and clearly distinct symbols for new and clearly distinct concepts (criteria from row 11). Anyway, we did take into account some concepts of BPMN, for example not using the classical clock but an hourglass to depict `TimeMeasures` (since BPMN uses such symbol for Timers).

Attending to the `dual coding` principle (row 12), in most cases, the text is used to complement graphics and not to distinguish them (for instance in connectors from and to, The text reinforces what we already expressed graphically by means of full and empty circles).

Regarding requirements from rows 13 and 14, they are somehow a bit subjective. Due to this fact, an experiment has been conducted. More information in section §10.3.1

Relative to row 15 (being `complexity manageable` to include explicit mechanisms for dealing with complexity, understanding this as the number of elements on a diagram), we addressed it in some way (*e.* the user can decide wether to explicit the mathematical function within the `derivedMeasure` or wether to model the measures an `aggregatedMeasure` aggregates or not to show them); however, we did not delve into this detail yet: there exists, thus, the possibility to overload a diagram with too much PPIs. For this issue we plan to extend the notation with a new PPI-view, where only PPIs would appear (without the BPMN diagram), and the relationships between them. We are also working on presenting the user those PPIs interesting for them, i.e., to allow the user to group PPIs according to certain property (e.g. all the PPIs defined for certain activity, or those regarding time) and only those PPIs will appear at the diagram.

Finally, we do not provide different dialects depending on the audience the notation is addressed to (experts versus novices) or the media in which it is represented (whiteboard, computer, etc), leaving thus uncovered the `cognitive fit` principle. We decided to use a standard notation, as it is done in BPMN. Furthermore, we do not include explicit mechanisms to support integration of information from different diagrams (`cognitive integration` principle) since there are not different views or diagrams in our approach nor in BPMN, preserving thus the alignment with this notation we extend.

7.4 GUIDELINES

In this section we intend to provide a simple guide to assist the user in defining PPIs using PPINOT graphical notation. The idea is to show the way to translate a definition of a PPI in natural language to the one using our graphical notation.

As stated in Section §6.1, in order to define good PPIs, is recommended that they satisfy the SMART criteria. Having PPIs defined in natural language, this SMART criteria is not always fulfilled. To be able to translate such PPI definition into the graphical notation (fulfilling thus the aforementioned characteristics), the user will have to answer three simple questions: *What to measure?*, *How to measure?* and *When to measure?*

1. **What to measure?**

   This question is related to the elements of the process on which to perform the measure, and the concept that needs to be measured over that element: time, repetitions, a certain condition, etc. In most cases, this information can be obtained by identifying noun or noun phrases.

2. **How to measure?**

   Here the user has to establish the way of obtaining the measure, *i.e.* if the measure can be performed directly over the elements of a process instance (taken from the execution of the process), or they can be obtained by aggregating several instances or applying a mathematical function.

3. **When to measure?**

   Finally, it is necessary to specify the instances to be measured, the moment in time and/or how often they must be measured. This implies to define, as stated before, a scope (using filters).

Table 7.13: subset of PPIs defined for the RFC management process

Description	**Periodicity**	**id**
Average time of committee decision	yearly	PPI2
Percentage of corrective changes	monthly	PPI3
Number of RFCs per project	yearly	PPI8

We take several examples related to the PPIs defined for the Request for Change Management process shown in Table §7.13 (that is an excerpt of Table §5.1).

- **Example 1: PPI2 - Average time of committee decision**
  - What to measure? looking at the definition of the PPI, we can identify two noun phrases: “time” and “committee decision”. Then, we can derive two facts: first,

that we need to measure time; it is noteworthy that whenever time is measured, a start (`from`) and an end (`to`) must be defined; the second fact is related to the process element over which we want to measure time. Looking to the process, it can be deduced that the committee decision is made during the activity "Analyse in committee", thus the start and the end correspond to the start and the end of this activity.

  – How to measure? This means how to obtain the value from the process. Let's see which of the three possibilities (directly, aggregating or through mathematical function) is the one to choose in this case: If we need to measure the average, it implies that more than one instance is taken into account. Moreover, it also indicates that we must aggregate several instances (`AggregatedMeasure`) using the aggregation function `average`.
  – When to measure? Finally, we have to identify how many instances we have to take into account when aggregating. This is defined in Table §7.13 through the "periodicity' =Yearly". That is, we need to aggregate every instance whose start is later or equal to the first of January and the end is before or equal to the thirty-first of December, every year.

- **Example 2: PPI8 - Number of RFCs per project**
  – What to measure? In this case, we have again three nouns: "number", "RFCs" and "project". The first one represents the concept we need to measure, *i.e.* number of times or repetitions. The second one is related to the element of the process over which to perform the measure. This can be done in different ways: directly in the start event (when receiving the request for change), over the dataObject "RFC registered" or even over the start of the whole pool of the process (since whenever the process is instantiated, a new RFC appears); we choose the first case, thus, the element is the start event "Receive RFC". Finally, the third noun does not correspond to this question and can induce to an error; the explanation comes in next paragraph.
  – How to measure? Since we need to count the number of times an RFC is received, we have to aggregate several instances. Furthermore, this time we are not interested in the average value as in the previous case, but in the total number; hence, we have to use the aggregation function `sum`. Regarding the noun "project" , this indicates we want to group the RFCs received according to the different project they belong to. Therefore, this aggregation must be done grouping by the data content "project" contained in the dataObject "RFC"
  – When to measure? To delimit the number of instances we have to take into account

when aggregating, we look again at the "periodicity" column in Table §7.13, this time is "monthly", what means that whenever a month ends, we must count the number of RFC received in that period and group them by project.

- **Example 3: PPI3 - Percentage of corrective changes**
  - What to measure? We can identify two noun phrases in the description: "percentage" and "corrective changes". The first one is directly related to the second question, since the fact of measuring a percentage implies to use a mathematical function applied to two other measures to calculate it. The issue in this case is to identify these two measures and follow the same process as in previous cases. For this example these two measures can be defined as "number of RFCs received" (this information is implied in the description of the PPI) and "number of corrective RFCs received" (deduced from the second noun phrase). The first one is almost the same as the preceding example, *i.e.* we measure "number of times of repetitions" over the start event "Receive RFC". In the second measure, we check which of the RFCs received fulfill the "condition" of being a corrective change; this information is contained in the data object "RFC received".
  - How to measure? As stated in the previous question, a mathematical function must be applied over the two mentioned measures. The mathematical function is $\frac{a*100}{b}$, where a' corresponds to "number of corrective RFCs received" (following the same process as the two previous example, we get that this is a `DataPropertyCondition-AggregatedMeasure`) and b to "number of RFCs received" (`CountAggregatedMeasure`)
  - When to measure? Again, as in the first example, we have that "periodicity' =Yearly", therefore, for both measures (a and b,) we have to aggregate the instances included in the whole year and then apply the function to that values.

7.5 Summary

In this chapter we have described PPINOT Graphical notation, our proposal on the graphical definition of PPIs over BPMN models. With this proposal we address the problem of the existing visual gap between BP models and PPI definitions. Furthermore it makes the definition of PPIs understandable for non-technical users. We have also described the principles that our design decision making is based on. Finally, we have also presented a set of guidelines to assist the user in the task of defining PPIs with our proposal. We summarise the problems we overcome with this contribution by means of the Kiviat diagram depicted in Figure §7.14.

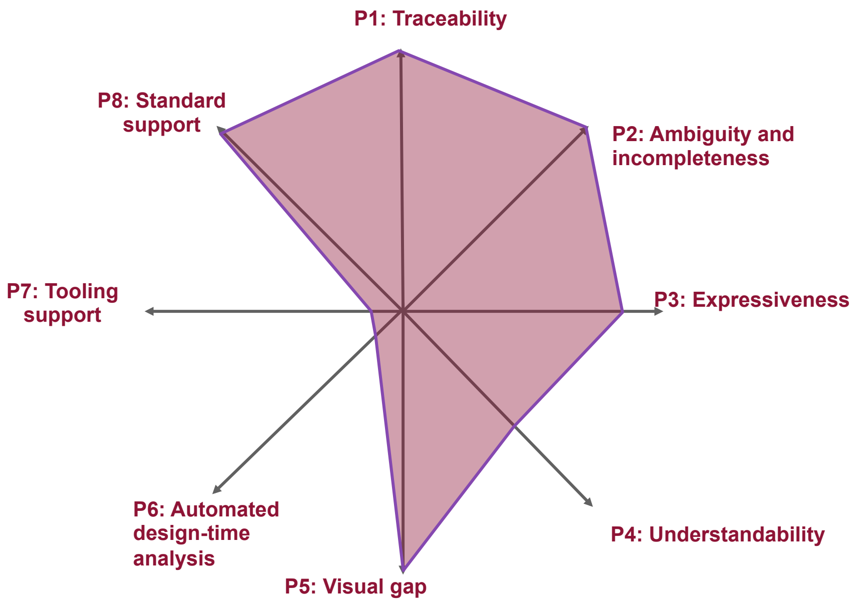


Figure 7.14: Kiviat diagram for the problems overcome by PPINOT Graphical Notation

PPI Templates and L-Patterns

I believe scientists have a duty to share the excitement and pleasure of their work with the general public, and I enjoy the challenge of presenting difficult ideas in an understandable way.

Antony Hewish (–)

In this chapter we propose a novel mechanism to improve the definition of PPIs using templates and L-PATTERNs based on PPINOT metamodel, introduced in Chapter §6. The main benefits of this approach are that it is easy to learn, promotes reuse, reduces ambiguities and the possibiity of missing information, is understandable to all stakeholders and maintains traceability with the process model. Furthermore, since the translation to a formal model based on Description Logics is straightforward (see Section §9.4), it makes it possible to perform automated analysis on these PPI definitions and infer knowledge regarding the relationships between them and to the process elements. Concretely, in Section §8.1, we introduce this chapter. In Section §8.2, we present PPI templates for its definition, making use of L-PATTERNs explained in Section §8.3. Finally, Section §8.5 summarises the chapter.

8.1 Introduction

According to Chapter §5, there not exists any notation that overcomes the problems described and fulfill the established requirements. The main reason is they are commonly considered conflicting characteristics; especially understandability to non-technical users and automated analysis. This is why in practice, either PPIs are defined in an informal an ad-hoc way, usually in natural language, with its well-known problems of ambiguity, lack of coherence, traceability with the process, incompleteness (missing information), and not amenable to automated analysis; or they are defined from an implementation perspective, at a very low level, too formal and/or close to the technical view, becoming thus hardly understandable to non-technical users.

In the previous chapter, the graphical notation presented addresses this challenge, but there are still some limitations to cope. The first one is, as outlined before, the visual scalability of our notation: as PPIs depicted grows, the model readability decreases. The second limitation is the learning curve it can represent to high-level managers or domain experts to get used to this kind of graphical notation. In this chapter we make a proposal to improve the definition of PPIs, that can complement PPINOT graphical notation, coping the aforementioned limitations. This is a novel approach based on templates and L-PATTERNs, which has been successfully used in the areas of requirements engineering [27, 28] and SLAs [84].

8.2 PPI Templates

The use of templates for the definition of PPIs helps to structure the information in a fixed form, in such a way that ambiguity disappears, it serves as a guide to know what missing information must be search and reuse is promoted.“In addition, filling blanks prewritten sentences, L-PATTERNs, is easier and faster than writing the whole paragraph describing what need to be measured in a process”. In this section, two templates for the definition of PPIs are described. They cover all of the sorts of PPIs described in §6. The notation used is the following: words between “$<$” and “$>$” must be properly replaced, words between “{” and “}” and separated by | represents options; only one option must be chosen. Finally words between “[” and “]” are optionals.

PPI–<*id*>	***<PPI descriptive name>***
Process	*<process id>*
Goals	*<strategic or operational goals the PPI is related to>*
Definition	The PPI is defined as ◦ *<TimeMeasure>* ◦ *<CountMeasure>* ◦ *<ConditionMeasure>* ◦ *<DataMeasure>* ◦ *<DerivedMeasure>* ◦ *<AggregatedMeasure>*
Target	The PPI value must ◦ be greater than [or equal to] *<lower bound>* ◦ be less than [or equal to] *<upper bound>* ◦ be between *<lower bound>* and *<upper bound>* [inclusive] ◦ fulfill the following constraint: *<target constraint>*
Unit of measure	*<unit of measure of the PPI>*
Scope	The process instances considered for this PPI are described in the scope *<S–x>*
Source	*<source from which the PPI can be taken>*
Responsible	{ *<role>* | *<department>* | *<organization>* | *<person>*}
Informed	{ *<role>* | *<department>* | *<organization>* | *<person>*}
Comments	*<additional comments about the PPI>*

Table 8.1: Template for PPI specification

8.2.1 PPI Template

Table §8.1 depicts the template for the definition of PPIs. It has been defined in order to fulfill the SMART criteria for the definition of PPIs and is heavily based on the metamodel described in §6. Actually there exists a clear correspondence between the fields of this template and the attributes of PPIs and measures described in the PPINOT metamodel. In the following we briefly describe the meaning of the template fields:

- **Identifier and descriptive name**: every PPI must be uniquely identified by a number and a descriptive name. In order to help rapid identification, PPIs identifiers start with PPI.

- **Process**: this field makes reference to the process for which the PPI is defined, and must contain the process identifier.

- **Goals**: this field allows the user to establish the strategic or operational goal/s that the PPI is related to. It can be fulfilled with an expression in natural language.

- **Definition**: this field uses an L-PATTERN that must be completed by the user. The different L-PATTERNs depending on the kind of PPI will be presented in Section §8.3.

- **Target**: this field is used to define the target value to be reached. This field uses an L-PATTERN that allows the user to define the three kinds of target values previously defined: the simple target, where the L-PATTERN must be completed with the lower bound and/or upper bound that make up the range within which the PPI value should be; the composed target, where the user can complete the L-PATTERN by defining several lower bound and/or upper bound (for the case of map values); and the custom target, where the L-PATTERN must be completed with the restriction that the PPI value must fulfill.

- **Unit of measure**: this field allows to define the unit of measure of the PPI (in the meta-model, this is an attribute of the measureDefinition). Sometimes, there is no unit of measure (*e.g.* percentages).

- **Scope**: this field corresponds to the attribute with the same name. It uses an L-PATTERN that must be completed with a scope-pattern that will be described in section §8.2.2. As a simplification, when it is a scope of the type last instances, can be completed directly with the number of instances.

- **Source of information**: this field makes reference to the source where the required information to obtain the PPI is gathered.

- **Responsible and Informed**: these fields corresponds to the analogous PPI attributes in the metamodel.

PPI–001	**Average time of RFC analysis**
Process	Request for change (RFC)
Goals	• BG–002: Improve customer satisfaction • BG–014: Reduce RFC time–to–response
Definition	The PPI is defined as *the average of the duration of the* Analyse RFC *activity*
Unit of measure	day
Target	The PPI value must be less than or equal to *one working day*
Scope	The process instances considered for this PPI are the last *100* ones
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	*Most RFCs are created after 12:00.*

Table 8.2: PPI specification example

- **Comments**: other information about the PPI that cannot be fitted in previous fields can be recorded here.

An example of use of this PPI template is shown in Table §8.2. This table represents the definition of the PPI "average time of RFC analysis" related to the business process of RFC management introduced in Section §5.2.

8.2.2 Template and L-Pattern for the Scope

We have defined the scope as a filter that selects the process instances that are considered for the computation of the PPI. Table §8.3 depicts the template for the definition of a scope. The meaning of the template fields is the following:

- **Identifier and descriptive name**: the same as in PPI template, except scope identifiers start with S.
- **Conditions**:this field allows to define state and/or temporal conditions over process instances to be considered when calculating the PPI value. These conditions can be combined using *and*, *or* and *not*. The user must complete the L-pattern according to the desired conditions with:

S–<*id*>	***<S descriptive name>***
Conditions	This scope includes process instances ○ [not] the last <*n*> ones [and | or] ○ [not] in state <*state*> [and | or] ○ [not] that started { before | after } [or at] { <*date*> | <*t*> <$unit_T$> ago } [and | or] ○ [not] that finished { before | after } [or at] { <*date*> | <*t*> <$unit_T$> ago }
Periodicity	The set of process instances is re–calculated ○ daily { every <*d*> days | every day } ○ weekly on <*days of week*> ○ monthly on { <*day of month*> | the <n^{th}> <*day of week*> } ○ yearly on { <*month*>, <*day of month*> | the <n^{th}> <*day of week*> of <*month*> }
Comments	<*additional comments about the scope*>

Table 8.3: Template for the Scope (S) specification

- the number of instances
- the state that process instances must be (or not be) in to take them into account, in the case of a process state condition,
- the temporal condition related to the start of the process. Concretely, the user can select instances that started “before«, “before or at”, “after” or “after or at” certain point in time. This point in time can be a concrete date (that must be completed) or a time window (defined by completing the time from now and the unit).
- the temporal condition related to the end of the process. The same conditions as in the previous case but related to the end of the process can be selected.

• **Periodicity**: this fields indicates the periodicity with which the PPI is calculated. It is defined using an L-PATTERN where the user has to chose between a daily, weekly, monthly and yearly periodicity. If a weekly periodicity is selected, the day of the week must be completed, and for the case of monthly periodicity, whether to take into account the day of the month (e.g. 3rd January) or the day of the week (e.g. third tuesday of the month) must be selected. The user can also indicate the frequency of such periodicity (e.g. every 2 months, for a monthly periodicity) and when to finish taking such measure (e.g. ends 31-12-2014).

An example of use of this template is presented in Table §8.4.

S-1	**Holidays period**
Conditions	This scope includes process instances in state completed and started after or at 01-08-2012 and ended before or at 31-08-2012 or started after or at 23-12-2012 and ended before or at 04-01-2013
Periodicity	yearly

Table 8.4: Example of an scope definition

8.3 L-Pattern catalogue for PPIs

In this section we present a catalogue of L-PATTERNs for the definition field of the PPI template. As stated before, L-PATTERNs are very used sentences in natural language that can be parameterised and integrated into templates. We propose the use of L-PATTERNs, since filling blanks in prewritten sentences is easier and faster than writing the whole paragraph describing what need to be measured in a process.

As extensively explained in Chapter §6, a PPI can be defined over a time, a count, a condition, a data, and a derived measure. Furthermore it can be defined at a single-instance or multi-instance level. Attending to these two dimensions, several sorts of measures can be used to define a PPI. In the following, the set of L-PATTERNs identified for these different sorts of measures that define a PPI are described.

8.3.1 Time Measure

When the PPI measures time, the following L-PATTERN is defined. The user must complete it with the point in the process where the PPI starts to count the time, and the point in the process where the PPI stops. This two points can correspond, as stated in Section§6.4.1 with time instants where a BP element activity, pool, or dataObject changes to certain state, or a time instant where certain event is triggered.

The PPI is defined as the duration between the time instant when {<BP element> <BP element name> *changes to state* <state> | *event* <event name> *is triggered} and the time instant when {*<BP element> <BP element name>*changes to state* <state> | *event* <event name> *is triggered}.*

An example of the L-PATTERN used for the definition of PPI "duration of RFC analysis" (using a time measure) is the following one:

The PPI is defined as the duration between the time instant when activity analyse RFC changes to state active and the time instant when activity analyse RFC changes to state completed.

8.3.2 Count Measure

When the PPI counts the number of times something happens, the L-PATTERN used to define the PPI is the following one. The user must complete the *BP element* and the *state* if the PPI measures a change of state, and the *event* if the PPI measures the trigger of an event.

The PPI is defined as the number of times {<BP element> <BP element name> *changes to state* <state> | *event* <event name> *is triggered}.*

An example of use of this L-PATTERN corresponding to the definition of PPI "RFC successfully analysed" (that uses a count measure) is presented below:

The PPI is defined as the number of times activity analyse RFC changes to state completed.

8.3.3 Condition Measure

When the PPI measures the fulfillment of certain condition, depending on whether this condition is referred to the state of a BP element or to a restriction of a dataObject, one of the two following L-PATTERNs must be chosen. The user must fullfill the *BP element* and the *state* in the first case, and the *restriction*, the *state* and the *dataObject* in the second one.

The PPI is defined as: <BP element> <BP element name> *{is currently | has finished} in state* <state>.

The PPI is defined as: dataObject <dataObject name> *[in the state* <state>*] satifies:* <restriction on dataObject properties>.

An example of use of the first L-PATTERN corresponding to the definition of PPI "RFC under analysis" (that uses a state condition measure) is presented below:

The PPI is defined as: activity analyse in committee is currently in state active.

An example of use of the second L-PATTERN corresponding to the definition of PPI "RFC with priority high" (that uses a data property condition measure) is presented below:

The PPI is defined as the fulfillment of the priority=high over the dataObject RFC.

8.3.4 Data Measure

When the PPI measures the value of certain content of a dataObject, the following L-PATTERN is used. The user must complete it with the *data content* and the *dataObject*.

The PPI is defined as the value of <data content> *of* <dataObject name>.

An example of use of this L-PATTERN corresponding to the definition of PPI "information systems that an RFC affects to" (that uses a data measure) is presented below:

The PPI is defined as the value of information systems of RFC.

8.3.5 Derived Measure

When the PPI is calculated by applying a mathematical function over some other measures, there appear two possible L-PATTERNs:

- one for the case of derived single-instance measures:

  The PPI is defined as the mathematical function <mathematical function with variables x_1, ..., x_n>, *where* <x_1> *is* <definition of the corresponding {time measure | count measure | condition measure | data measure | derived single-instance measure}>, ..., <x_n> *is* <definition of the corresponding {time measure | count measure | condition measure | data measure | derived single-instance measure}>

- one for the case of derived multi-instance measures:

The PPI is defined as the mathematical function <mathematical function with variables x_1, ..., x_n...>, *where* <x_1> *is* <definition of the corresponding {aggregated measure | derived multi-instance measure}>, ..., <x_n> *is* <definition of the corresponding {aggregated measure | derived multi-instance measure}>

Table §8.5 shows another example of use of the PPI template, for a PPI defined by a derived process measure, taking as starting point the same process. It refers to the scope definition S-1, depicted in Table §8.4

PPI–005	**Percentage of RFCs approved from registered**
Process	Request for change (RFC)
Goals	• BG–002: Improve customer satisfaction
Definition	The PPI is defined as *the mathematical function (a/b)*100 where a is the number of times dataObject RFC is in state approved and b is the number of times dataObject RFC is in state registered*
Unit of measure	none
Target	The PPI value must be greater than or equal to *90 %*
Scope	The process instances considered for this PPI are described in the analysis period *S-1*
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	*first values of this PPI are dated in 2007.*

Table 8.5: PPI specification example (defined over a derived process measure)

8.3.6 Aggregated Measure

When the PPI value is calculated by aggregating one of the previous measures in several process instances, the L-PATTERN varies according to the single instance measure that is aggregated. In the following we describe all the possible L-PATTERNs. The parts that need to be completed for the user in each L-PATTERN are those explained before in the corresponding subsection. Furthermore, as stated in Section §6.4.6, in all the cases it is possible to group them by certain data content, and hence the last part of the following L-PATTERNs is added.

- Time aggregated measure:

  The PPI is defined as the {sum | maximum | minimum | average} of the duration between the time instant when {<BP element> <BP element name> *changes to state* <state> | *event* <event name> *is triggered} and the time instant when {*<BP element> <BP element name> *changes to state* <state> | *event* <event name> *is triggered} [and is grouped by* <data content> *of* <dataObject> *] .*

- Count aggregated measure:

  The PPI is defined as the {sum | maximum | minimum | average} of the number of times {<BP element> <BP element name> *changes to state* <state>*, event* <event name> *is triggered} [and is grouped by* <data content> *of* <dataObject>*].*

  An example of this L-PATTERN corresponding to the definition of PPI "number of RFCs per project" is presented below:

  The PPI is defined as the sum of the number of times event Receive RFC is triggered and is grouped by project of RFC.

- Condition aggregated measure:

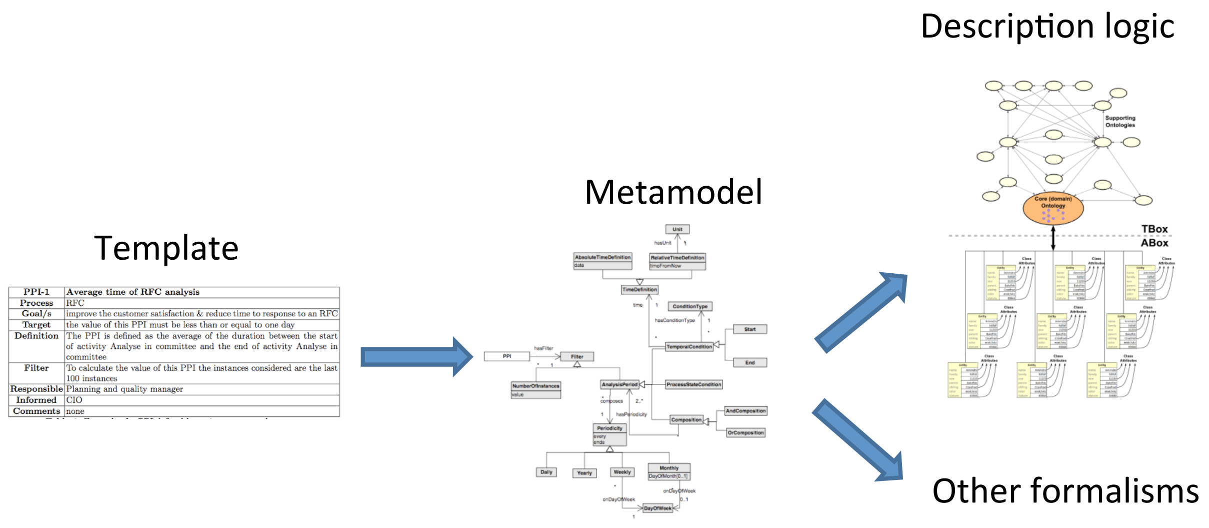
  The PPI is defined as the {sum | maximum | minimum | average} of the number of times {<BP element> <BP element name> *{is currently, has finished} in state* <state>*, the* <restriction> *over the dataObject* <dataObject name> *[in state* <state>*] is fulfilled} [and is grouped by* <data content> *of* <dataObject>*].*

- Data aggregated measure:

  The PPI is defined as the {sum | maximum | minimum | average} of the value of <data content> *of* <dataObject> *[and is grouped by* <data content> *of* <dataObject>*].*

- Derived aggregated measure:

  The PPI is defined as the {sum | maximum | minimum | average} of the value of <mathematical function over single-instance measures> *[and is grouped by* <data content> *of* <dataObject>*].*



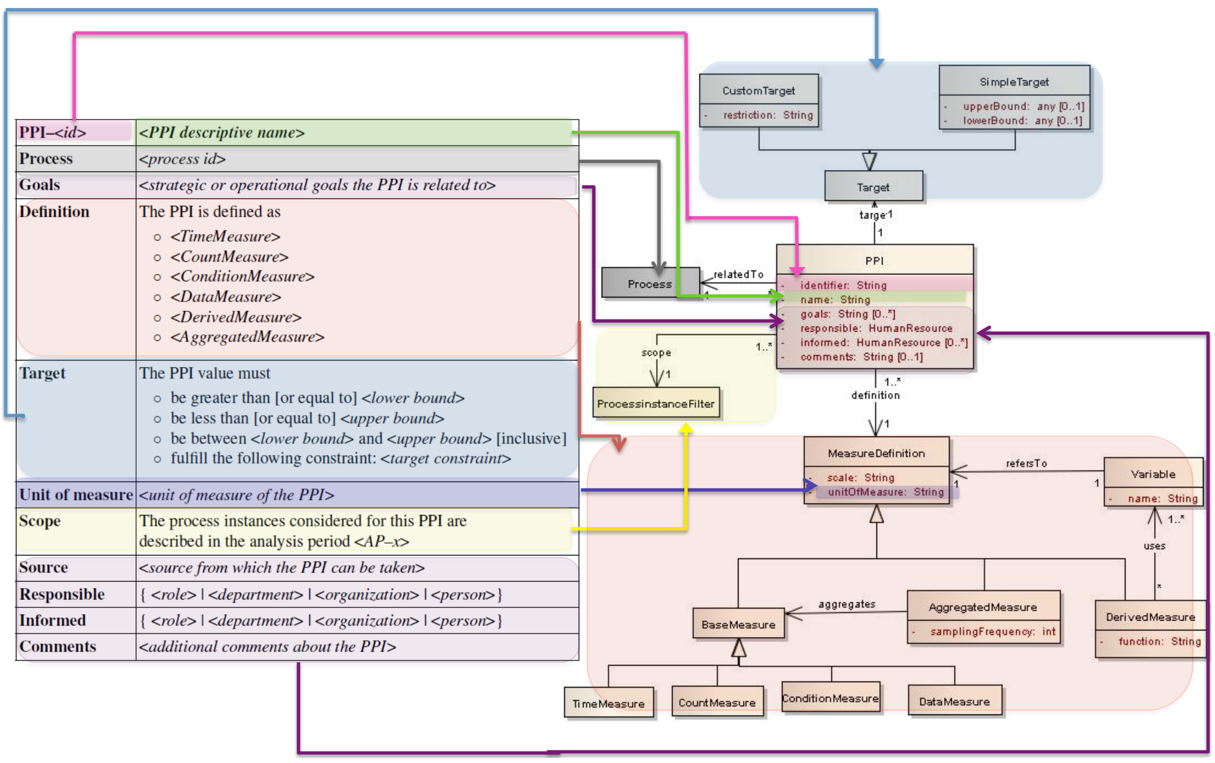
PPI-1	**Average time of RFC analysis**
Process	RFC
Goal/s	improve the customer satisfaction & reduce time to response to an RFC
Target	the value of this PPI must be less than or equal to one day
Definition	The PPI is defined as the average of the duration between the start of activity Analyse in committee and the end of activity Analyse in committee
Filter	To calculate the value of this PPI the instances considered are the last 100 instances
Responsible	Planning and quality manager
Informed	CIO
Comments	none

Figure 8.1: Mapping: from the user view to the computer view

8.4 Mapping PPI Templates and L-Patterns to PPINOT Metamodel

Figure §8.2 depicts the mapping from the PPI-template to PPINOT metamodel. It shows that there exists a correspondence 1:1 between the fields *identifier*, *name*, *goals*, *responsible*, *informed* and *comments* and the corresponding attributes with the same names in the class *PPI*. The same happens with the field *unit of measure* and the corresponding attribute in class *MeasureDefinition*. Furthermore, the first three options given for the *target* correspond to the class *SingleTarget* and the last one to the class *CustomTarget* in PPINOT metamodel. Furthermore, the information to be fulfilled by the user in these options corresponds to the *lowerBound*, *upperBound* and *restriction* attributes respectively. Regarding the mapping of the PPI definition, there is also a direct correspondence between the options *TimeMeasure*, *CountMeasure*, *ConditionMeasure*, *DataMeasure*, *DerivedMeasure* and *AggregatedMeasure*, and the classes with the same name of PPINOT metamodel. The information that the user provides by fulfilling the L-patterns, is mapped to a number of *condition* classes, together with their *state* and *restriction* attributes, as well as to the *BPElement* class. Associations *aggregates* and *aggregationFunction* as well as the class *AggregationFunction* are used to map the information fulfilled in the aggregated measure L-Pattern. Finally, an association called *uses* and the class *Variable* is used to map the information contained in the derived measure L-pattern. This last part of the mapping is not contained in Figure §8.2 for the sake of simplicity and readability.

Regarding the definition of the scope, Figure §8.3 summarises the mapping between the AP-template and the corresponding part of the PPINOT metamodel. On the one hand, the con-

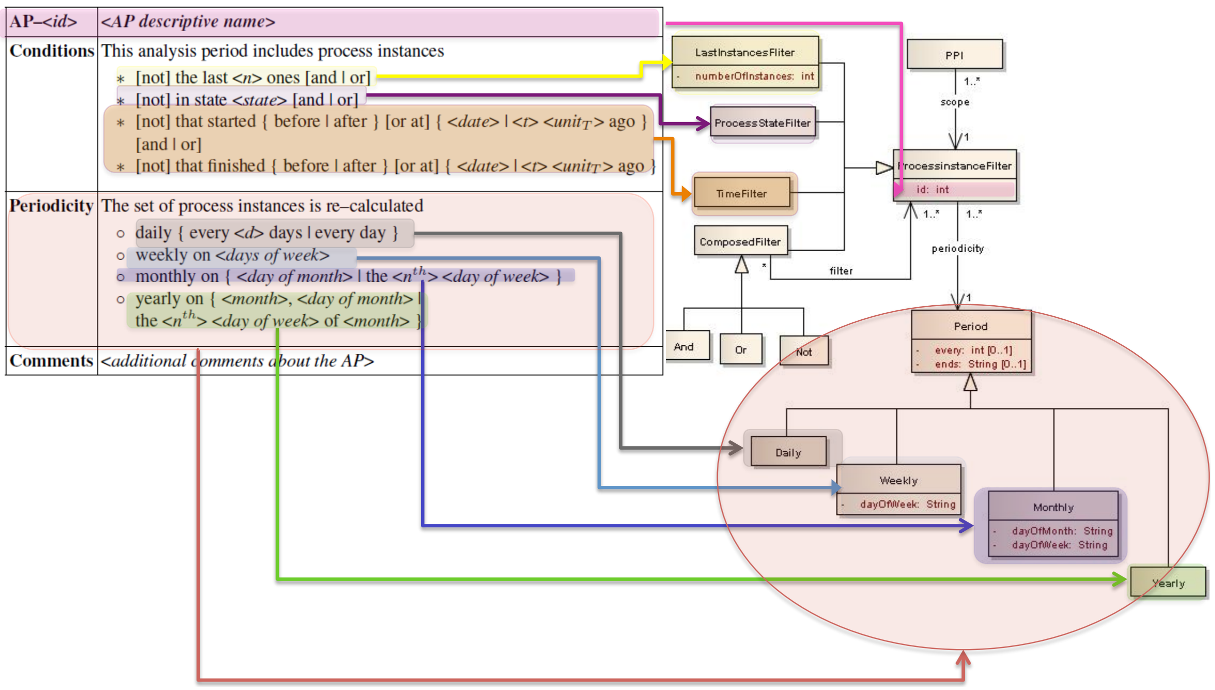


PPI–*<id>*	*<PPI descriptive name>*
Process	*<process id>*
Goals	*<strategic or operational goals the PPI is related to>*
Definition	The PPI is defined as ○ *<TimeMeasure>* ○ *<CountMeasure>* ○ *<ConditionMeasure>* ○ *<DataMeasure>* ○ *<DerivedMeasure>* ○ *<AggregatedMeasure>*
Target	The PPI value must ○ be greater than [or equal to] *<lower bound>* ○ be less than [or equal to] *<upper bound>* ○ be between *<lower bound>* and *<upper bound>* [inclusive] ○ fulfill the following constraint: *<target constraint>*
Unit of measure	*<unit of measure of the PPI>*
Scope	The process instances considered for this PPI are described in the analysis period *<AP–x>*
Source	*<source from which the PPI can be taken>*
Responsible	{ *<role>* | *<department>* | *<organization>* | *<person>* }
Informed	{ *<role>* | *<department>* | *<organization>* | *<person>* }
Comments	*<additional comments about the PPI>*

Figure 8.2: Mapping of PPI template to PPINOT metamodel

ditions in AP-template are translated to the corresponding classes in the metamodel; concretely, the first condition corresponds to the class *LastInstancessFilter*, and the information fulfilled by the user is mapped to the attribute *numberOfInstances*; the second condition corresponds to the class *ProcessStateFilter* and the information fulfilled by the user is mapped to the attribute *processState*; finally, the third and the fourth ones correspond to the class *TimeFilter*. Regarding the information fulfilled by the user, in this case, there are several aspects to be fulfilled, the *berfore* | *after* and *or* | *at* selection is mapped to an attribute called *conditionType* (whose value is enumerated), then, if the user choose a *date* or a time *t* is mapped to the classes *AbsoluteTimeDefinition* and *RelativeTimeDefinition*, respectively. Finally, if the third condition is selected, it is mapped to the value *start* of the attribute *moment*, and if it is the fourth condition the chosen, to he value *end*.Note that all this information of the metamodel is not depicted in Figure §8.3 for the sake of simplicity and readability. On the other hand, with respect to the periodicity, it is mapped to the class *Period*, concretely, each option is mapped to one of its subclasses *Daily*, *Weekly*, *Monthly* and *Yearly*, respectively, and the information fulfilled by the user to the corresponding attributes.

It is noteworthy that every condition starts with an optional *not* and ends (except the last one) with an optional *and* | *or*. This options are mapped to the class *ComposedFilter* and allow to form any combination of the different filters (options) provided by using *and*, *or* and *not*.



AP–<*id*>	*<AP descriptive name>*
Conditions	This analysis period includes process instances ∗ [not] the last <*n*> ones [and | or] ∗ [not] in state <*state*> [and | or] ∗ [not] that started { before | after } [or at] { <*date*> | <*t*> <$unit_T$> ago } [and | or] ∗ [not] that finished { before | after } [or at] { <*date*> | <*t*> <$unit_T$> ago }
Periodicity	The set of process instances is re–calculated ○ daily { every <*d*> days | every day } ○ weekly on <*days of week*> ○ monthly on { <*day of month*> | the <n^{th}> <*day of week*> } ○ yearly on { <*month*>, <*day of month*> | the <n^{th}> <*day of week*> of <*month*> }
Comments	*<additional comments about the AP>*

Figure 8.3: Mapping of AP template to PPINOT metamodel

8.5 Summary

In this chapter we have presented a notation based on templates and L-PATTERNs that can help to define PPIs while keeping the benefits from using natural language and avoiding early formalization of PPIs. Concretely two templates, one for PPIs and one for the definition of the scope have been presented. As previously stated, this notation is easy to learn, promotes reuse, reduces ambiguities and the possibiity of missing information, is understandable to all stakeholders and maintains traceability with the process model. Furthermore we have shown the mapping to our PPINOT metamodel, that will allow a subsequent automated analysis on these PPI definitions. We summarise the problems we overcome with this contribution by means of the Kiviat diagram depicted in Figure §8.4.

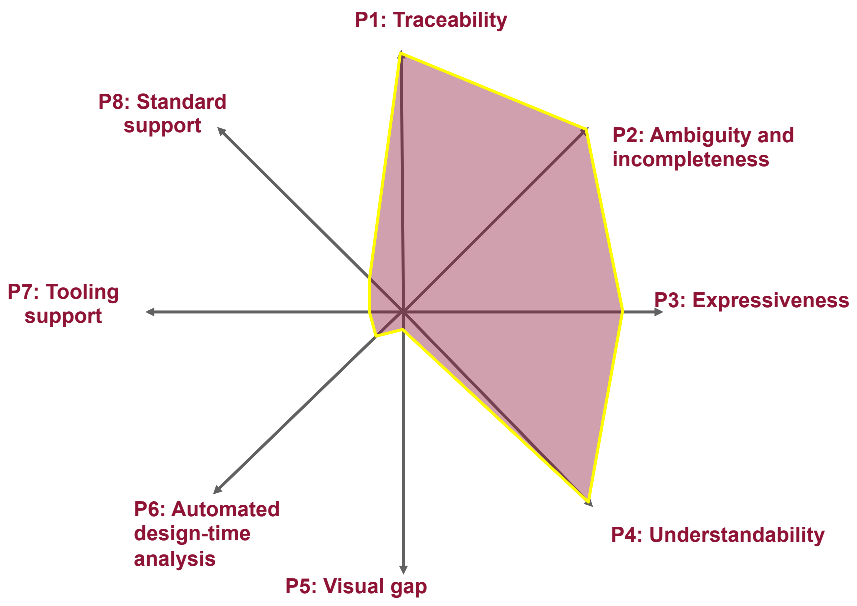


Figure 8.4: Kiviat diagram for the problems overcome by PPI Templates and L-patterns

Automated Analysis of PPIs

Technology does not drive change, it enables change.

Unknown source,

In this chapter we deal with the automated analysis of PPIs. Until now, the effort on this field has been driven to the development of techniques and tools to automatically analyse BPs and PPIs at runtime (specially BAM techniques) and post mortem, i.e. using the information obtained from those process instances already finished (e.g. techniques from the fields of business intelligence, data warehousing and process mining); but little efforts have been conducted to automatically analysed PPIs at design time. We propose here a set of analysis operations, grouped in families, that allow to obtain, at design-time, information implicitly contained in the PPI definitions. This information can assist business process analysts during PPIs definition and business process evolution. Furthermore, we provide a formalisation of the PPINOT metamodel using DL as well as the implementation of the aforementioned operation analysis. Thanks to this formalisation, it is possible to use the power of existing DL reasoning systems to automatically analysed PPIs.

The structure of this chapter is the following one: In Section §9.1, we introduce the main issues addressed in this chapter. Then, Section §9.2.2 and Section §9.3 describe two families of analysis operations, one for obtaining the relationships between PPIs and BP elements, and the other for obtaining the relationships between PPIs themselves, respectively. In Section §9.4 we provide the formalisation of our approach using DL. Finally Section §9.5 summarises the chapter.

9.1 Introduction

The automated analysis of PPIs can be defined as the computer-aided extraction of information from PPI models and instances. This analysis allows to investigate properties of PPI specifications. The analysis of PPIs is performed in terms of analysis operations, which take a set of parameters as input and returns a result as output. In this chapter, we focus on two families of operations, namely: operations that obtain information about the relationship between PPIs and the elements of a business process at design-time, and operations that obtain information about the relationship between PPIs themselves at design-time. Each family of operations includes two or more operations and for each of them, its definition, an example, and possible practical applications are presented.

Furthermore, in this chapter, we also provide a formalisation of our approach by means of DL. The goals of this formalisation are two. First, it helps eliminating ambiguities in the metamodel as well as provides more semantics about some of the concepts and relationships. Second, it facilitates the aforementioned automated analysis of the PPIs model by leveraging reasoner services available for the formalism.

In this thesis, DL are used to formalise the PPINOT metamodel. We could have used some other formalism, but we opted for DL as it serves our purpose by the following reasons. First, the definition of a set of PPIs for a business process fits nicely into the way DLs express their concepts and, hence, it provides a very natural way to describe the problem. Second, powerful reasoning systems for DL like Racer [80], Hermit [81] and Pellet [71] have been developed, allowing thus to build upon such standard DL reasoning services to automatically and efficiently analyse PPIs. Finally, the fact that OWL [58], the industry standard and W3C recommendation for the representation of ontologies, is based on DLs has caused the development of a variety of software tools that facilitates working with DLs such as Protégé [37, 79] and its associated OWL Plugin [73].

Note that we do not state that other fromalisms cannot be used to formalise the PPINOT metamodel; however, a discussion of the appropriateness of other formalisms, although a relevant research topic, are outside the scope of this dissertation.

9.2 Relationship between PPIs and business process elements

The PPINOT metamodel puts a special emphasis on the relationship between PPIs and business process elements. This enables a design-time analysis of the PPIs model together with the business process model that can be materialised into two operations that are described next:

It is possible to obtain information about the way PPIs and business process elements influence each other. The possible operations are described below:

9.2.1 Operations for the relationship *measured by*

A BP element is measured by a PPI when the value of the PPI is calculated by measuring some aspect of the execution of the BP element. For instance, let us say we have a PPI that measures the *average lifetime of an approved RFC* and, hence, it can be defined as the average of the duration between the time instant when the event *Receive RFC* is triggered and the time instant when the end event *Report RFC Approved* is triggered. In this case, the elements measured by the PPI are the event *Receive RFC* and the event *Report RFC Approved* since the value of the PPI is calculated by measuring the moment in which both events are triggered.

This relationship can be straightforwardly inferred from the measure definition of a PPI in the PPINOT metamodel. If the measure definition is a base measure, then the elements measured by the PPI are those to which the condition used by the measure definition applies. If the measure definition is an aggregated measure or a derived measure, then the elements measured are those measured by the base measure that the aggregated or derived measure aggregates or uses, respectively. Two analysis operations can be defined based on this relationship.

BP elements measured by a set of PPIs

This operation takes a set of PPIs and their corresponding BP model as input and returns the set of business process elements that are measured by those PPIs.

$$MeasuredBPElement(PPI[1..*], BP) : BPElement[0..*]$$

This operation is useful when processes are instrumented to take the measures that are necessary to calculate the PPIs. It provides information that helps process analysers and developers to abstract away from other aspects of the PPIs and focus on the BP elements whose execution must be measured. Furthermore, if PPIs change, it can provide useful information about whether new BP elements should be instrumented.

PPIs that measure a given BPElement

This operation takes a set of PPIs, their corresponding BP model and a BP element as input and returns the PPIs that are calculated by measuring the given BP element.

$$MeasureType(PPI[1..*], BP, BPElement) : PPI[0..*]$$

The information provided by this operation is useful to find out about which are the PPIs that get affected if there is a BP element whose execution cannot be measured.

9.2.2 Operations for the relationship *involved in*

A BP element is involved in a PPI when it has an influence in the value of the PPI. For instance, if we take the same example as before (the *average lifetime of an approved RFC*), the elements involved in the PPI are all the elements that by taking more or less time to execute make the average lifetime to be longer or shorter. In this case these elements are the element where the time measure ends (event *Report RFC Approved*) and all the elements between that element and the element where time starts to be counted (event *Receive RFC*), i.e. activity *Analyse in committee*, activity *Analyse RFC*, activity *Elevate decision to committee*, activity *Analyse in committee* and activity *Approve RFC*. Note that event *Receive RFC* is not included since time starts counting when *Receive RFC* is triggered, which means it does not have any influence on the duration of the RFC lifetime.

Unlike relationship *measured by*, the set of elements involved in a PPI cannot always be directly inferred from the PPINOT metamodel since there are many factors that can make a BP element to have an influence in the value of a PPI. For example, the type of an RFC may have an influence on the lifetime of an approved RFC since some activities may take more time if the RFC corresponds to an adaptive change than if it corresponds to a corrective change.

Nevertheless, it is possible to leverage the definition of the PPI and the control flow of the business process to make a design-time estimation of the BP elements that have an influence on the PPI. This estimation can be later refined by means of techniques similar to those described in [**?**] to find out relationships between PPIs such as:

- Knowledge of domain experts can be used to determine possible domain-specific influences of a BP element in a PPI.

Measure type	Elements involved
Time	(1) The element where time starts and ends to be measured unless the state considered are end or start respectively and (2) Every BP element between the element where time starts to be measured and the element where it ends (except for data objects since they do not consume time).
Count	(1) The element that is being counted and (2) the XOR gateways that have taken the execution path to that element.
Condition	(1) The element used in the condition and (2) if the condition involves a data object, the activities that can write in it.
Data	(1) The data object being measured and (2) the activities that can modify the data object.
Aggregated	(1) The elements involved in the measure that it aggregates and (2) if it groups by some value, the data object that provides it.
Derived	The elements involved in the measures used in the mathematical function applied to calculate the value.

Table 9.1: BP elements involved in a PPI

- Data mining techniques can be applied to the data collected during the execution of the process. For instance, an analysis of the executions of a process may conclude that the content of the RFC have an influence on the previous PPI if the likelihood of having a longer average lifetime changes depending on the type of RFC.

Obtaining automatically a design-time estimation of the BP elements that have an influence on a PPI is useful because of the following reasons. First, it is not necessary to have execution data available, which is an advantage because sometimes it is not possible to have execution data, either because it is very costly to obtain or because it refers to business process or PPIs that are being designed and, hence, have not been executed yet. Second, although domain experts can determine possible domain-specific influences of a BP element in a PPI without execution data, this task is error prone, specially when the number of PPIs or elements are high. Therefore, this automated estimation can be used as an input or a complement for domain experts to determine domain-specific influences. Finally, if the design-time estimation is obtained automatically, it helps the modeller of PPIs to quickly analyse different configurations of PPIs in order to find one that covers the most relevant BP elements.

Table §9.1 summarises the elements involved in a PPI. Like relationship *measured by*, the elements involved in a PPI depends on the type of its measure definition as detailed next:

- In a time measure, the elements that have an influence on the PPI are those that can make the measure longer or shorter, i.e. the elements that are executed between the start and the end of the time measure. This includes the elements at the start or at the end if they some of the time of its execution is included in the time measure.
- In a count measure, the elements that have an influence on the PPI are those that take the execution path to the element whose execution is being counted.
- In a condition measure, the elements that have an influence on the PPI are the element used in the condition and if the condition involves a data object, the activities that can write in it.
- In a data measure, the value of a data object used by the process is measured. Therefore, the elements that have an influence on the PPI are the data itself and all of the activities that can modify the data.
- In an aggregated or derived measure, the elements that have an influence on the PPI are those who have an influence on the measures that are aggregated or used by each of them respectively. Furthermore, if the aggregated measure groups results by some value, the data object that provides it have also an influence on the PPI.

Regardless on whether the relationship is inferred at design-time or refined using one of the aforementioned techniques the following analysis operations can be defined.

BP elements involved in a PPI

This operation takes a set of PPIs and the corresponding BP model as input and returns the set of business process elements involved in that PPI, i.e, it allows to answer the question *Given a PPI, which are the elements of the process model that it assesses?*.

$$InvolvedBPElement(PPI[1..*], BP) : BPElement[0..*]$$

Related with this operation other similar operations can be defined that returns:

- The BP elements that are not involved in any PPI:

$$NotInvolvedBPElement(PPI[1..*], BP) : BPElement[0..*]$$

- The BP elements that are involved in all of the PPIs:

$$InvolvedInAllBPElement(PPI[1..*], BP) : BPElement[0..*]$$

The information provided by these operations is very useful when a PPI must be replaced with others (maybe because it is very costly to obtain its value) in order to assure that every element of the business process that was taken into consideration before is taken into consideration in the new case. It is also useful to find out which are the elements in a business process that are not involved with any PPI. This would mean that those elements are not being taken into consideration by the current set of PPIs and, hence, that it may be convenient to introduce or modify a PPI to cover them.

PPIs associated to a BP element

This operation is the inverse of the previous one. It takes a set of BP elements, the BP model that contains those BP elements and the PPIs model defined for that BP as input and returns the set of PPIs that are associated to those BP elements.

$$AssociatedPPI(BPElement, BP, PPI[1..*]) : PPI[0..*]$$

As an example, the PPIs that are used to assess the activity *Analyse RFC* are the following PPIs:

- PPI5 because it directly measures the average duration of that activity.
- PPI6 because it measures the number of RFCs with state *in analysis*, which is the state in which RFCs are while the activity *Analyse RFC* is being executed. Therefore, this number may be influenced by the throughput of this activity amongst other reasons.
- PPI9 because it measures the average lifetime of a RFC and, hence, the duration of activity *Analyse RFC* has an influence on it.

This operation allows to answer the question *given a particular business process element, which are the PPIs associated to them?*. The information obtained in this operation can help in assisting during the evolution of business processes. For instance, if part of the business process evolves and is modified (e.g., an activity is deleted), this analysis allows to identify which PPIs will be affected and may be updated.

9.3 Relationships between PPIs

One of the problems that appear when defining PPIs and setting their target values is that they are often conflicting and improving one of them may worsen other. Therefore, finding relationships between PPIs is very appealing since it helps to detect this conflicts and, hence, to understand better the consequences of trying to optimise one PPI or another. The PPINOT metamodel enables a design-time analysis of these relationships based on the following operations.

9.3.1 PPIs associated to the same, a subset or a superset of the elements of other PPI

These operations are derived from the operation PPIs associated to a BP element, explained in the previous section. They take a PPI, the set of PPIs defined by a business process and the corresponding BP model as inputs and returns the set of PPIs that are associated to either:

- the same elements as the given PPI:

$$PPISameElements(PPI, PPI[1..*], BP) : PPI[0..*]$$

- a subset of the elements of the given PPI:

$$PPISubsetElements(PPI, PPI[1..*], BP) : PPI[0..*]$$

- a superset of the elements of the given PPI:

$$PPISupersetElements(PPI, PPI[1..*], BP) : PPI[0..*]$$

The information provided by these operations can be useful to find out about possible redundancies amongst the PPIs defined for a business process. For instance, many PPIs associated to a superset of the elements of a given PPI could be an indicator that the PPI is providing redundant information since there are many PPIs associated to the same BP elements. Nevertheless, these operations just provides hints and it is always the task of the PPI modeller to decide whether an indicator provides redundant information or not.

9.3.2 Time-related PPIs that include other time-related PPIs

This operation takes a time-related PPI (i.e. a PPI defined by means of a time measure or an aggregation of time measures), the PPIs model and the corresponding BP model as inputs and returns the set of time-related PPIs that are included in it.

$$IncludedTimeMeasures(PPI, PPIs, BP) : set < PPI >$$

A time-related PPI p_1 is included in another time-related PPI p_2 if the fragment of the process measured by p_1 is contained in the fragment of the process measured by p_2. This can happen in three cases:

1. if both of them are defined over time measures and p_2 is contained in p_1.
2. if p_1 is defined over an aggregated measure that aggregates a time measure and p_2 is defined over a time measure and p_2 is contained in p_1.
3. if p_1 and p_2 are defined over aggregated measures that aggregate a time measure and p_2 is contained in p_1.

For instance, a PPI defined as *the duration between the time instant when the event* Receive RFC *is triggered and the time instant when the end event* Report RFC Approved *is triggered* includes a PPI defined as *the duration between the time instant when activity* Analyse RFC *starts and the time instant when activity* Analyse RFC *ends* since the process fragment measured by the latter PPI (activity *Analyse RFC*) is contained in the process fragment measured by the former (the whole process).

The information provided by this operation may be useful to find redundant time-related PPIs. Furthermore, it can be used to set dependencies between PPIs since an increase in the value of a PPI included by other PPI may cause an increase in the value of the latter.

9.3.3 PPIs that depends on other PPI

This operation takes a PPI, the PPIs model and the corresponding BP model as inputs and returns the set of PPIs that depends directly or inversely with the given PPI.

$$DependsDirectlyOn(PPI, PPIs, BP) : set < PPI >$$

$$DependsInverselyOn(PPI, PPIs, BP) : set < PPI >$$

A PPI P_1 depends directly on another PPI P_2 if any of these conditions hold:

- P_1 is defined as an aggregation of the measure that defines P_2
- P_1 is defined as a derived measure that is calculated as a mathematical function of the measure that defines P_2 and the changes in the latter affects the other measure in the same

direction (positively). For instance, if $PercentRequestsApproved = \frac{RequestsApproved}{RequestRegistered} \times 100$, then *PercentRequestsApproved* is calculated positively from *Requests Approved* and is calculated negatively from *Requests Registered*.

- P_1 is a time-related PPI that includes P_2.
- P_1 depends directly on another PPI P_3 and this PPI depends directly on P_2
- P_1 depends inversely on another PPI P_3 and this PPI depends inversely on P_2

Similarly, a PPI P_1 depends inversely on another PPI P_2 if any of these conditions hold:

- P_1 is defined as a derived measure that is calculated as a mathematical function of the measure that defines P_2 and the changes in the latter affects the other measure in the opposite direction (negatively).
- P_1 depends directly on another PPI P_3 and this PPI depends inversely on P_2
- P_1 depends inversely on another PPI P_3 and this PPI depends directly on P_2

The information provided by this operation is useful to build graphs of dependencies between PPIs so that the analyst can visually analyse the dependencies between the PPIs that have been defined. It can also be used in *what-if* analyses to evaluate the impact trying to improve a PPI may have on the others and, hence, to identify potentially conflicting PPIs.

9.4 Formalising the PPINOT Metamodel Using Description Logics

This section introduces a formalisation of the PPINOT Metamodel using DL, whose two main goals are, as stated before, on the one hand to help eliminating ambiguities in the metamodel as well as to provide more semantics about some of the concepts and relationships, and on the other hand, to facilitate an automated analysis of the PPIs model by leveraging reasoner services available for the formalism.

9.4.1 Mapping the PPINOT Metamodel into DL

The mapping of the PPINOT metamodel into DL involves defining a DL knowledge base that includes the classes and relations of the metamodel as axioms of its TBox. In particular, we must create one DL concept for each class of the metamodel (keeping the same names) and set in the knowledge base the hierarchies that appear in the metamodel using the concept inclusion axiom (e.g. $BaseMeasure \sqsubseteq InstanceMeasure \sqsubseteq MeasureDefinition$).

The directed associations in the metamodel are mapped into roles of the TBox whose domain is the DL concept corresponding to the class of the metamodel that acts as source, and whose range is the DL concept corresponding to class that acts as target. The cardinality restrictions of the associations are mapped as concept inclusion axioms to the source class. For instance, the cardinality restriction: "*each Condition appliesTo at most one BPElement*" is mapped into the following axiom: $Condition \sqsubseteq \leq 1 appliesTo.BPElement$. In addition, we have created inverse roles for most of the roles to make the formulation of some analysis operations easier. For example, roles $isUsedBy$, $isAggregatedBy$, $isFromFor$ and $isUsedInCondition$ have been defined as the inverse roles of $isDefinedOver$, $aggregates$, $from$ and $appliesTo$ respectively. Finally, we introduce a new DL role $relatesTo$ as a super-role of the roles that relate measures with conditions (i.e. $from \sqsubseteq relatesTo$, $to \sqsubseteq relatesTo$, $when \sqsubseteq relatesTo$ and so on).

9.4.2 Automated Analysis of PPIs using DL

An advantage of formalising the definition of PPIs using DL is the possibility to automate the analysis operations described in Sections §9.2.2 and §9.3 by means of DL reasoners. DL reasoners are software tools that implement several operations on the ontologies in an efficient manner by using several heuristics and techniques. Some examples of such operations are:

- $satisfiability(C)$: It determines whether a description of the concept C is not contradictory with the KB.
- $subsumes(A,B)$: It determines whether concept A subsumes concept B, i.e., whether description of A is more general than description of B.
- $individuals(C)$: It finds all individuals that are instances of concept C.
- $realization(i)$: It finds all concepts to which the individual i belongs.

On the basis of these operations and the mapping of the metamodel to DL described above, the analysis operations detailed in Sections §9.2.2 and §9.3 can be formulated so that DL reasoners can be used to implement them. Specifically, the approach we follow to formalise these operations in DL is to define the relationship as a new role in the knowledge base and, then to formulate the operations in terms of DL reasoning operations.

Operations for the relationship *measured by*

This relationship is formalised in DL by defining a new role in the KB ($measuredBy$), whose domain is $BPElement$ and whose range is PPI so that $measuredBy(e, p)$ means that a BP element e is *measured by* PPI p.

To give semantics to the relationship in terms of the elements defined in the KB we use the fact that the relationship *measured by* can be inferred from the measure definition of a PPI. The idea is to introduce a new role $measures$ with domain $MeasureDefinition$ and range $BPElement$ that relates measure definitions with BP elements and, then, use this new role in the definition of $measuredBy$ as follows:

$$measures^- \circ definition^- \sqsubseteq measuredBy$$

Finally, role $measures$ is used to relate each measure definition with the BP elements used in the definition. In the KB, this can be formalised using the composition of roles as follows:

$$\begin{gathered}
relatesTo \circ appliesTo \sqsubseteq measures \\
measuresData \circ data \sqsubseteq measures \\
aggregates \circ measures \sqsubseteq measures \\
isGroupedBy \circ data \sqsubseteq measures \\
uses \circ refersTo \circ measures \sqsubseteq measures
\end{gathered}$$

The first two axioms define role $measures$ for base measures, the next two axioms define it for aggregated measures and the last axiom defines it for derived measures. Note that in both aggregated and derived measures role $measures$ is defined recursively based on the measure definitions they aggregate or compose.

Based on role $measuredBy$, the operations for this relationship can be easily formulated as follows:

- The BP elements measured by a set of PPIs P are those elements e that have a role $measuredBy(e, p)$, where $p \in P$. In DL, this can be expressed as the result of $individuals(\exists measuredBy.P)$.
- Similarly, the PPIs that measure a given BPElement e is equivalent to the result of $individuals(\exists measuredBy^-.\{e\})$.

Operations for the relationship *involved in*

Like in relationship *measured by*, the elements involved in a PPI P are determined by its definition, which is expressed in the metamodel by means of a measure definition. Therefore the approach we follow with this relationship is very similar to the one followed with relationship *measured by*.

Specifically, we define two new roles in the KB, namely: role $involvedIn$ whose domain is $BPElement$ and whose range is PPI so that $involvedIn(e, p)$ means that a BPElement e is involved in a PPI p, and role inv whose domain is also $BPElement$, but whose range is $MeasureDefinition$. Finally, the following axiom formalises the relationship between them by stating that the elements involved in a PPI are those involved in its measure definition:

$$inv \circ defines^{-} \sqsubseteq involvedIn$$

Consequently, the definition of role $involvedIn$ can be reduced to the definition of role inv. However, this relationship is more complex than relationship *measured by* and, hence, we cannot define role inv in a generic way, but for each measure definition. In particular, we add an axiom that represent the BP elements involved in a measure definition m for each measure definition m in the KB. This axiom may vary depending on the type of the measure definition (i.e. time, count, condition, data, aggregated or derived) as detailed in Table §9.1. Next we detail each of the possible cases.

Time measure According to Table §9.1 the elements involved for a time measure tm are $\exists inv.\{tm\} \equiv ElemStart_{tm} \sqcup ElemEnd_{tm} \sqcup ElemPath_{tm}$, where:

- $ElemStart_{tm}$ is the element where time starts to be measured unless the state in which the measure starts is an end state. In that case, $ElemStart_{tm}$ is empty: $ElemStart_{tm} \equiv \exists appliesTo^{-}.(\exists from^{-}.\{tm\} \sqcap \neg\exists state.EndState)$
- $ElemEnd_{tm}$ is the element used in the condition that specifies when time ends to be measured unless the state specified by the condition is a start state, in which case it is empty: $ElemEnd_{tm} \equiv \exists appliesTo^{-}.(\exists to^{-}.\{tm\} \sqcap \neg\exists state.StartState)$
- $ElemPath_{tm}$ is the BP elements that succeeds the start and precedes the end: $ElemPath_{tm} \equiv \exists succ.(\exists appliesTo^{-}.(\exists from^{-}.\{tm\})) \sqcap \exists prec.(\exists appliesTo^{-}.(\exists to^{-}.\{tm\}))$

Count measure In this case, the elements involved for a count measure cm are: $\exists inv.\{cm\} \equiv ElemCount_{cm} \sqcup InvolvedXor_{cm}$, where:

- $ElemCount_{cm}$ is the element that is being counted: $ElemCount_{cm} \equiv \exists appliesTo^{-}.(\exists when^{-}.\{cm\})$
- $InvolvedXor_{cm}$ corresponds to every gateway that precedes the element that is being counted: $InvolvedXor_{cm} \equiv XorGateway \sqcap \exists prec.(\exists appliesTo^{-}.(\exists when^{\cdot} - \{cm\}))$

Condition measure The elements involved for a condition measure cm are: $\exists inv.\{cm\} \equiv ElemCond_{cm} \sqcup ElemWriters_{cm}$, where:

- $ElemCond_{cm}$ is the element whose condition is being evaluated:

  $ElemCond_{cm} \equiv \exists appliesTo^{-}.(\exists meets^{-}.\{cm\})$.
- $ElemWriters_{cm}$ correponds to every activity that can modify, i.e., write the data object whose state is being evaluated if there is any: $ElemWriters_{cm} \equiv \exists dataOutput.(DataObject \sqcap ElemCond_{cm})$

Data measure The elements involved in a data measure dm are: $\exists inv.\{dm\} \equiv DataMeasured_{dm} \cup ElemWriters_{dm}$, where:

- $DataMeasured_{dm}$ is the data object that is being measured:

  $DataMeasured_{dm} \equiv \exists data^{-}.(\exists measuresData^{-}.\{dm\})$.
- $ElemWriters_{dm}$ is the activities that could have modified that data object $ElemWriters_{cm} \equiv \exists dataOutput.DataMeasured_{dm}$

Aggregated measure The elements involved in an aggregated measure am are: $\exists inv.\{am\} \equiv \exists inv.\{bm\} \sqcup InvolvedData_{am}$, where:

- $\exists inv.\{bm\}$ is the elements involved in the base measure that the aggregated measure aggregates (i.e. $bm \in \exists aggregates^{-}.\{am\}$).
- $InvolvedData_{am}$ is the data that the aggregated measure groups by if there is any: $InvolvedData_{am} \equiv \exists data^{-}.(\exists isGroupedBy^{-}.\{am\})$

Derived measure Finally, for a derived measure dm, the elements involved will be those elements involved in each of the measures used in the mathematical function applied to calculate the value. Let $d_1,...,d_n$ be the first, ..., nth measure definition used to calculate dm ($\{d_1,\ldots,d_n\} \in \exists isUsedToCalculate.\{dm\}$), then, the elements involved in the derived measure are the union of the elements involved in these measures: $\exists inv.\{dm\} \equiv \exists inv.\{d_1\} \sqcup \cdots \sqcup \exists inv.\{d_n\}$.

These definitions have two limitations regarding time measures and count measures that are worth mentioning. The first one is the definition regarding time measures considers all of the elements between the first occurrence of the element that starts the time measure and the last occurrence of the element that ends the time measure. However, if the time measure is cyclic, it only should include the elements until the first occurrence of the element that ends the time measure. The consequence is that if the time measure is cyclic, then the definition may include more involved elements than it should (i.e. those between the first and the last occurrence of the element that ends the time measure). However, from a practical point of view, this only affects when the time measure is defined in a loop since in other cases the first and the last occurrence of the element that ends the time measure coincides. The second limitation is that in count measures we consider every preceding gateway and we do not exclude those opening gateways that were already closed with another one (that is, for instance, a splitting gateway with its corresponding merge).

Both limitations could be solved using another formalism such as Petri nets to analyse the control flow of the business process and including the result of the analysis in the KB. For instance, Petri nets could be used to obtain the process fragment between the first occurrence of the element that starts the time measure and the first occurrence of the element that ends it. However, the specific mechanism on how to do so is out of the scope of this paper.

Like relationship *measured by*, operations for relationship *involved in* can be formulated based on role $involvedIn$ as follows:

- Operation $InvolvedBPElement$, which returns the BP elements that are involved in a set of PPIs P can be formulated as $individuals(\exists involvedIn.P)$
- Operation $NotInvolvedBPElement$, which returns the BP elements that are not involved in any PPI of the set of PPIs P can be formulated as $individuals(\neg(\exists involvedIn.P))$
- Operation $InvolvedInAllBPElement$, which returns the BP elements that are involved in all of the PPIs included in a set of PPIs P can be formulated as

  $individuals(\exists involvedIn.\{p_1\} \cap \ldots \cap \exists involvedIn.\{p_n\})$, where $p_1,\ldots p_n \in P$.

- Operation $AssociatedPPI$, which returns the PPIs in which a given BP element e is involved can be formulated as $individuals(\exists involvedIn^{-}.\{e\})$.

Operations to identify PPIs associated to the same, a subset or a superset of the elements of other PPI

The last three operations related to the involved in relationship are classified outside the previous subsection because they allow to extract information regarding the relationships between PPIs. As described above, they returns the PPIs associated to the same, a subset or a superset of the elements of a given PPI p must be done in three steps. First, a concept $InvolvedBPElement_q$ that represents all the BP elements that are involved by a PPI q ($InvolvedBPElement_q \equiv \exists involvedIn.\{q\}$) is defined for each PPI q included in the KB. This allows the DL reasoner to classify all these concepts by using DL operation $subsumes$. Then, this classification is used to obtain the concepts that are equivalent or a subset or a superset to the concept $InvolvedBPElement_p$ depending on whether we are defining operation $PPISameElements$, $PPISubsetElements$ or $PPISupersetElements$, respectively. In each case, the result is a set of $InvolvedBPElement_{p_1}, \ldots, InvolvedBPElement_{p_n}$ concepts. The final step is to obtain the PPIs $p_1, \ldots, p_n$ associated to those $InvolvedBPElement$ concepts. These are the resulting PPIs.

Operation Included Time-related PPIs

In order to implement this operation, we define a new role in the KB called $inlcudes$, whose domain is $MeasureDefinition$ and whose range is also $MeasureDefinition$ so that $includes(m1, m2)$ means that a measure definition $m2$ is included in a measure definition $m1$. According to the definition of the inclusion relationships given in Section §9.3.2, we define a set of inference rules[1] that allow to obtain the corresponding inclusion relationships.

Let $m1$, $m2$, $n1$ and $n2$ be four measure definitions, $c1$, $c2$, $c3$ and $c4$ four timeInstantConditions, and $e1$, $e2$, $e3$, and $e4$ four BP elements, then, the three inference rules corresponding to the three possible cases of inclusions are:

[1]Rules are expressed using a syntax close to the Semantic Web Rules Language (SWRL).

$$\begin{aligned}
&from(?m1,?c1), appliesTo(?c1,?e1),\\
&from(?m2,?c2), appliesTo(?c2,?e2), succeeds(?e2,?e1),\\
&to(?m1,?c3), appliesTo(?c3,?e3),\\
&to(?m2,?c4), appliesTo(?c4,?e4), precedes(?e4,?e3) \longrightarrow includes(?m1,?m2)
\end{aligned}$$

$$\begin{aligned}
&aggregates(?m1,?n1), from(?n1,?c1), appliesTo(?c1,?e1),\\
&from(?m2,?c2), appliesTo(?c2,?e2), succeeds(?e2,?e1),\\
&to(?n1,?c3), appliesTo(?c3,?e3),\\
&to(?m2,?c4), appliesTo(?c4,?e4), precedes(?e4,?e3) \longrightarrow includes(?m1,?m2)
\end{aligned}$$

$$\begin{aligned}
&aggregates(?m1,?n1), from(?n1,?c1), appliesTo(?c1,?e1),\\
&aggregates(?m2,?n2), from(?n2,?c2), appliesTo(?c2,?e2),\\
&succeeds(?e2,?e1),\\
&to(?n1,?c3), appliesTo(?c3,?e3),\\
&to(?n2,?c4), appliesTo(?c4,?e4), precedes(?e4,?e3) \longrightarrow includes(?m1,?m2)
\end{aligned}$$

Furthermore, we also define another role, called $inlcudesPPI$, whose domain and range are PPI and that establish the inclusion relationship between two PPIs, *i.e.*, analogously to $includes(m1,m2)$, $includesPPI(p1,p2)$ means that a PPI $p2$ is included in a PPI $p1$. We finally define an inference rule that allows to obtain the inclusion relationship between PPIs:

$$\begin{aligned}
&isDefinedOver(?p1,?m1),\\
&isDefinedOver(?p2,?m2),\\
&\quad includes(?m1,?m2) \longrightarrow includesPPI((?p1,?p2)
\end{aligned}$$

Where $p1$ and $p2$ are two PPIs.

Operation Depends on PPIs

As stated in Section §9.3.3, a PPI P_1 depends directly on another PPI P_2 if one of the five conditions indicated hold. In order to implement this operation, the task is twofold:

1. We define roles $aggregates$, $includes$ and $isCalculatedPositively$ as subroles of $dependsDirectlyOn$

(by using the inclusion axiom: $e.g\ aggregates \sqsubseteq dependsDirectlyOn$). By doing this we cover the first three conditions.

2. To deal with the last two conditions, first, we define the roles $dependsDirectlyOn$ and $dependsInverselyOn$, with $MeasureDefinition$ as domain and ranges, so that $dependsDirectlyOn(m1,m2)$ means that a measure definition $m1$ depends directly on $m2$ and the same for $dependsIndirectlyOn(m1,m2)$ but with the inverse dependency. Then, we also define the role $dependsDirectlyOnPPI$, that is analogous to $dependsDirectlyOn$ but between PPIs. Finally, we define the following inference rules:

$$\begin{aligned}
&dependsInverselyOn(?m1,?m2),\\
&dependsInverselyOn(?m2,?m3) \longrightarrow dependsDirectlyOn(?m1,?m3)\\
&dependsInverselyOn(?m1,?m2),\\
&dependsDirectlyOn(?m2,?m3) \longrightarrow dependsInverselyOn(?m1,?m3)\\
&isDefinedOver(?p1,?m1),\\
&isDefinedOver(?p2,?m2),\\
&dependsDirectlyOn(?m1,?m2) \longrightarrow dependsDirectlyOnPPI(?p1,?p2)
\end{aligned}$$

For the implementation of the inverse dependency, according to the three conditions, we proceed as follows

1. To deal with the first condition, we define the role $isCalculatedNegatively$ as subrole of $dependsInverselyOn$ (by using the inclusion axiom: $isCalculatedNegatively \sqsubseteq dependsInverselyOn$.

2. The rest of conditions are implemented by defining the role $dependsInverselyOnPPI$ (that analogously to $dependsDirectlyOnPPI$, defines the inverse relationship between PPIs) and the following inference rules:

$$\begin{aligned}
&dependsDirectlyOn(?m1,?m2),\\
&dependsInverselyOn(?m2,?m3) \longrightarrow dependsInverselyOn(?m1,?m3)\\
&isDefinedOver(?p1,?m1),\\
&isDefinedOver(?p2,?m2),\\
&dependsInverselyOn(?m1,?m2) \longrightarrow dependsInverselyOnPPI(?p1,?p2)
\end{aligned}$$

9.5 Summary

In this chapter, we have presented the automated analysis of PPIs at design-time by means of analysis operations that allow to investigate properties of their definitions. Concretely we have presented two families of analysis operations: those that provide information about the relationships between PPIs and BP elements; and those that provide information about the relationships between the PPIs themselves. For each analysis operations, we have provided its definition, an example, and possible practical applications. Furthermore, in order to facilitate this automated analysis, we have also proposed a formalisation of the PPINOT metamodel using *Description Logics (DL)*. We have described the mapping of such metamodel to a DL *Knowledge Base (KB)*, as well as the way the different analysis operations are implemented using this formalism. This mapping can also be used to implement other new analysis operations.

We summarise the problems we overcome with this contribution by means of the Kiviat diagram depicted in Figure §9.1.

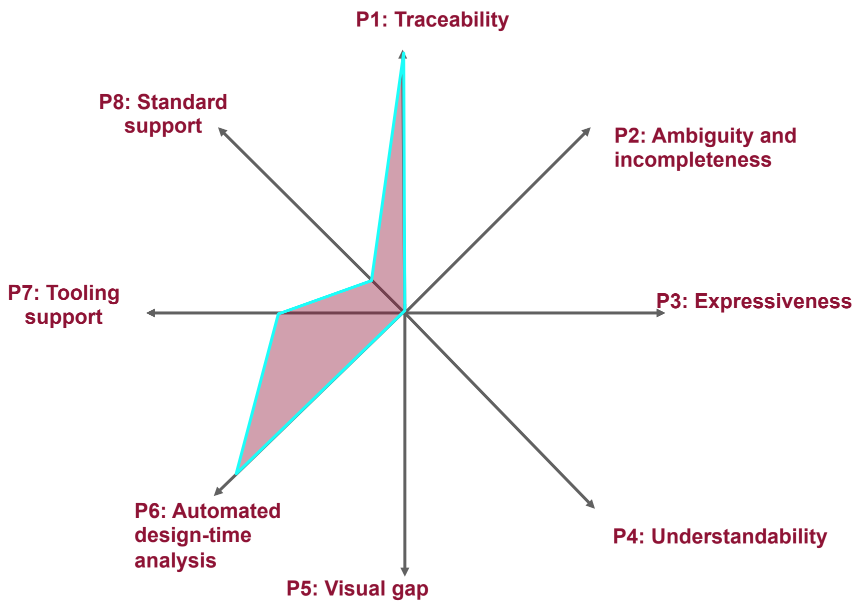


Figure 9.1: Kiviat diagram for the problems overcome by the catalogue of automated analysis operations

Evaluating the Approach with PPINOT Tool Suite

Before software can be reusable it first has to be usable.

Ralph Johnson (1955–),
American Computer Scientist

In order to be able to exploit the results of the work presented in the previous chapters, it is necessary to provide the user with tool support The goal of this chapter is to present PPINOT tool suite, an easy-to-use tool suite that provides support to the phases of design and instrumentation of the PPIM lifecycle, overcoming the problems that existing proposals have. Concretely, it provides both, a graphical and a template-based editor, as well as an analyser component that provides support for the automated analysis presented in this thesis. In addition, a mechanism to extract the information required to calculate PPI values from an open source BPMS has been developed. This tool serves as a proof of concepts of our approach. Furthermore, we present the real scenarios where our approach has been applied in order to show its applicability.

The remainder of this chapter is structured as follows: Section §10.1 introduces the chapter, focusing on the presentation of PPINOT tool. In Section §10.2 we describe the tool and its structure, as well as the way it works, its inputs and outputs, providing some details regarding the implementation. The set of real scenarios where our approach was applied together with an experiment conducted to evaluate its usability are presented in Section §10.3. Finally in Section §10.4, we summarize the main points of the chapter.

10.1 Introduction

In the previous chapters we have presented a novel approach for the definition and design-time analysis of PPIs. In order to make these results available to users, a software tool support was desirable. To address this issue and as a proof of concepts, we present in this chapter PPINOT tool. It is a prototype tool that allows the graphical definition of PPIs together with their corresponding BPs, based on the PPINOT metamodel, and their subsequent automated analysis, previously explained. We can highlight the following PPINOT tool features:

BPMN 2.0 compliant. PPIs can be defined over BP diagrams (BPDs) previously modelled using the de facto standard BPMN 2.0.

Graphical definition of PPIs. PPINOT supports the graphical definition of PPIs using a graph-based graphical notation that is easily understandable by non-technical users, at the same time that it is supported by a metamodel that assures the precise and complete definition of PPIs.

Automated analysis of PPIs. The aforementioned metamodel support allows to automatically formalise PPI definitions using Description Logics, enabling, inter alia, to obtain information about the way PPIs and BP elements influence each other. Concretely, two kinds of analysis operations are supported in the current version of PPINOT: (I) *BPElements involved*, that allows to answer the question *Given a PPI P, Which are the process model's elements involved?*, this information is very useful in many scenarios, like for instance when a PPI must be replaced with others (maybe because it is very costly to obtain its value) and it is necessary to assure that every element of the BP that was measured before is measured in the new case; and (II) *PPIs associated to BPElement*, that allows to answer the question *Given a BPElement E, Which are the PPIs associated or applied to them?*, this information can assist during the evolution of BPs, e.g. if a part of the BP has evolved and is modified, for instance if an activity is deleted, this analysis allows to identify which PPIs will be affected and should be updated.

Template-based definition of PPIs. PPIs can be defined by fulfilling templates written in structured natural language, where the user only has to properly introduce the missing information, assisted by linguistic patterns.

PPI definition mapping Graphical definition of PPIs can be mapped to their corresponding templates in natural language.

PPI values computation Taking as starting point any the aforementioned PPI definitions, PPINOT also provides the possibility to extract the information required to calculate PPI values from Activiti, an open source BP management platform, and to create reports with these values.

10.2 PPINOT Tool Suite: Definition and Structure

The structure of PPINOT is depicted in the component model of Figure §10.1. One part of this tool, the PPINOT Graphical Editor and the PPINOT Analyser, has been implemented as an extension of the ORYX platform. Concretely we have provided a new stencil set with the shapes and connectors of the PPI graphical notation (PPINOT Graphical Editor), that extends the existing one of BPMN. In addition, a new plugin called *PPINOT analyser* has been developed. It supports the formalisation of PPI graphical definitions to DL, and the subsequent analysis. Furthermore, it has been designed as a reusable component so that it can be integrated into other environments than Oryx without any change. In the following we describe the way this part of PPINOT works.

A business process diagram (BPD) is defined in *Oryx* [20]. Then, the set of PPIs is defined over such BPD using the *PPINOT Oryx Stencilset* (as depicted in Figure §10.2). An xml file containing all this information (BPD + PPIs) is obtained from Oryx (through the *PPINOT Service*) and mapped to OWL using the PPINOT ontology described in [22]. This is done by the *XML2OWL Mapper* and produces the OWL file BP +PPIs Ontology, which is the target file of the DL reasoner (in this case *HermiT*) used by the *PPI Analyser*, so the proper DL operations are executed on the PPI definitions of this OWL file to infer the information required. Finally, an OWL file containing the information required (elements involved in a PPI definition or PPIs associated to a given BP element) is automatically generated (PPIAnalysisOntology).

Furthermore, an editor for the definition of PPIs using templates has been developed. This is the PPINOT Templates Editor Component. In this case, the tool provides a template of PPI where, depending on the selection that the user performs in the different fields, the corresponding linguistic patterns are shown, and within these patterns, the user must fulfill the blanks according to her objective. In addition, this component is also able to read the XML file produced by the PPINOT Graphical Editor in order to show the corresponding PPI template.

Finally, the XML that contains the information of the PPIs and the associated BP, is the input for the PPINOT Instrumenter, that allows to gather from Activiti the required information to compute their values. Finally, these values are shown by the PPINOT Reporter.

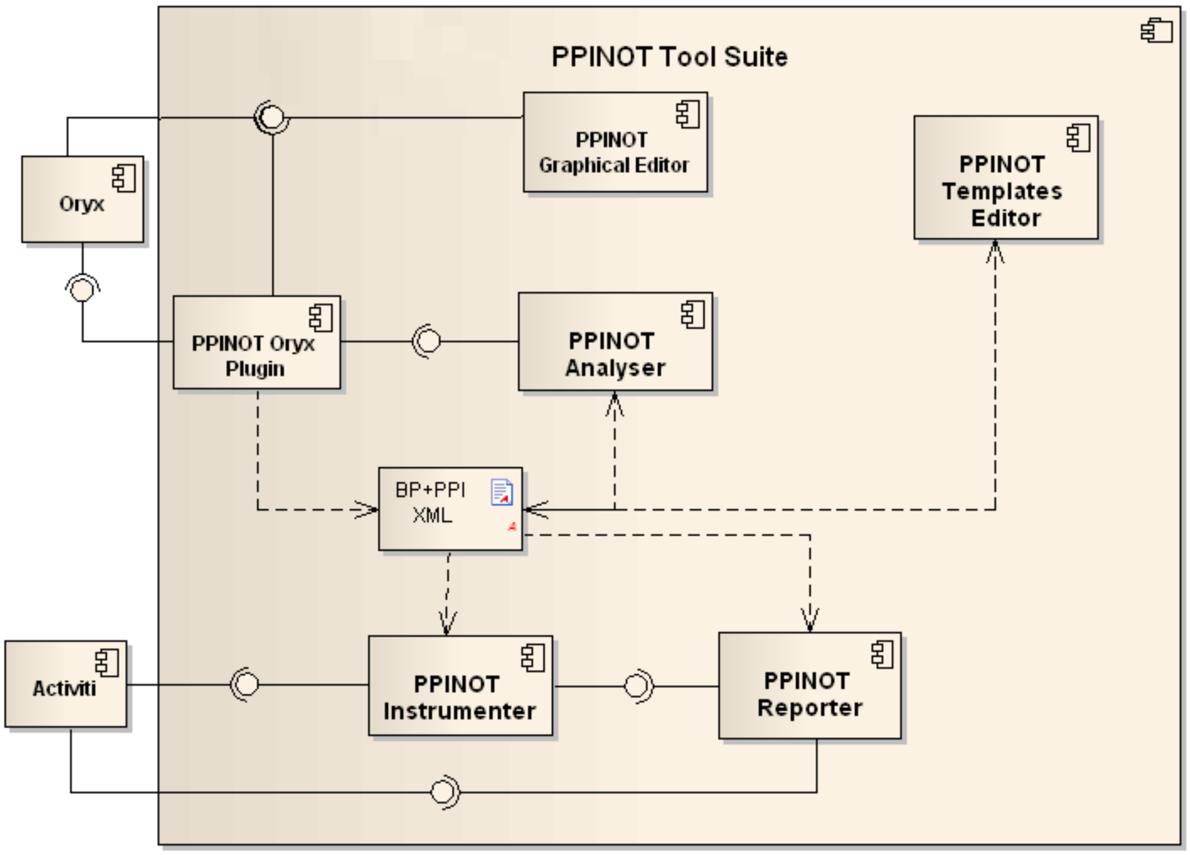


Figure 10.1: PPINOT component model

This tool suite can be tested following the instructions described at http://www.isa.us.es/PPINOT.

In the following, we present some details (those that we consider most interesting) related to some of the components that constitute PPINOT tool Suite.

10.2.1 PPINOT Graphical Editor

We integrated PPINOT graphical notation (already presented in Chapter §7) into the web-based editor ORYX as a result of a collaboration stay with the BPT group at the HPI Potsdam. We developed an extension allowing the user to depict PPIs over the BPMN diagrams using our PPINOT Notation.

Oryx was born in Hasso Plattner Institute in 2006 as an open source and extensible platform for business process management. Since Sept 30, 2011, the Oryx Online system was discontinued and people from academia were redirected to the BPM academic initiative solutions. One of this solutions is the open source one called Signavio Core Components, specifically designed

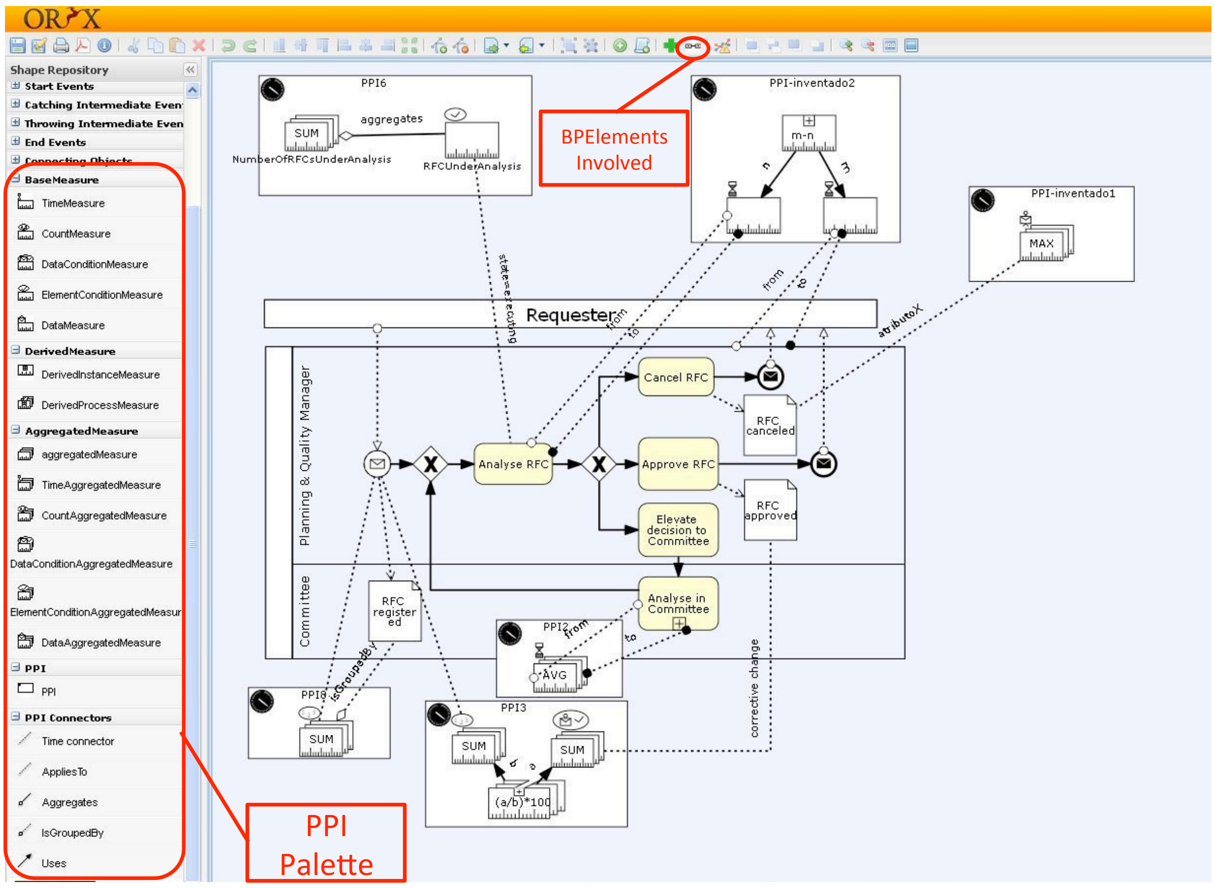


Figure 10.2: PPINOT screenshot

to be easily extendable. In order to support the modelling of PPIs using our graphical notation together with BPMN diagrams we developed an extension of the BPMN stencil set (provided in these Signavio Core Components) by adding the new shapes and connectors previously explained. This is done by implementing a JSON file and a set of SVG and pictures files (one of each them per defined stencil). For every stencil, a set of properties (attributes) and rules (e.g. connection, morphing, cardinality) is defined.

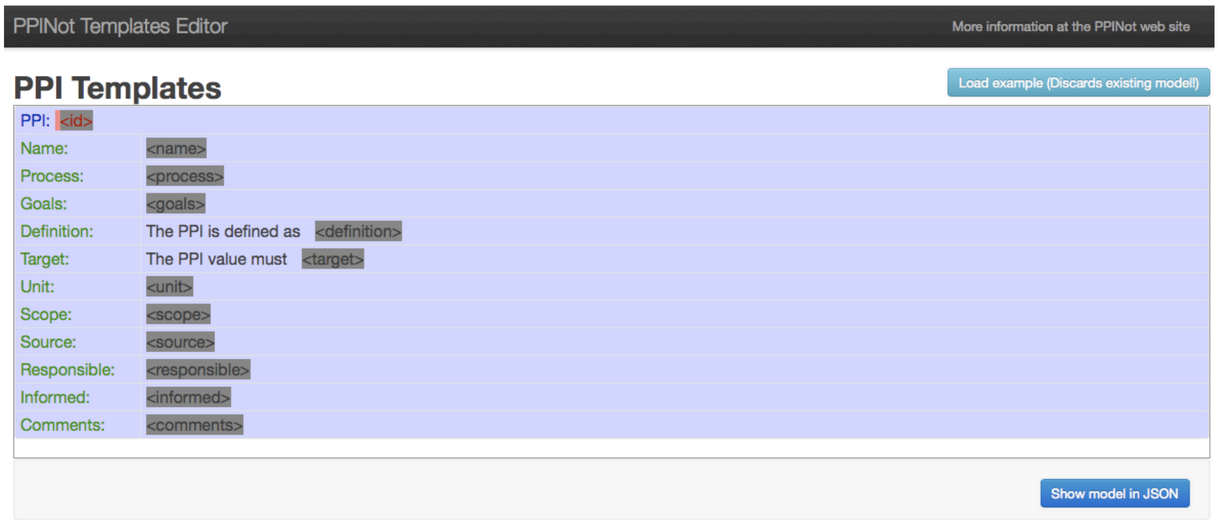
This tool can be accessed via the website `http://labs.isa.us.es:8081/backend/poem/repository`. It runs in everyone's browser[1]).

10.2.2 PPINOT Template Editor

We have developed a PPI template editor using the *Concrete* Editor (`http://concrete-editor.org`). In order to support our PPI templates, its metamodel written in a Json style was provided as input for the *Concrete Editor* and produced the template editor shown in Figure§10.3. Once

[1]Firefox browser must be used. This is a limitation imposed by ORYX web editor

the PPI template is defined, it can be saved as a json file, that can be translated to the an XML in order to be analysed or used by the instrumenter to compute PPI values.


PPINot Templates Editor
More information at the PPINot web site
PPI Templates
Load example (Discards existing model!)
Show model in JSON

PPI: <id>

Name:	<name>
Process:	<process>
Goals:	<goals>
Definition:	The PPI is defined as <definition>
Target:	The PPI value must <target>
Unit:	<unit>
Scope:	<scope>
Source:	<source>
Responsible:	<responsible>
Informed:	<informed>
Comments:	<comments>

Figure 10.3: PPINOT screenshot

10.2.3 PPI Analyser

This component has been developed as several plugins for Oryx. In its current version, the whole set of operation is not implemented yet, but a subset of it. Concretely the operations related to the relationships between PPIs and BPs. The user select the elements that act as input from the set presented by the plugin and thanks to several libraries, the proper answer is provided.

We plan to implement the rest of operations, as well as to make this implementation independent from Oryx, so that it can be used in any environment, taking as starting point the XML file that contains the information related to the BP and the PPIs defined for it.

10.3 Real Application Scenarios

In order to show the applicability of our proposal, in this section we present an experiment conducted to evaluate the understandability of PPINOT graphical notation, and a set of case studies where this notation have been applied with satisfactory results.

10.3.1 Experiment

With the objective of demonstrating the usability of our proposal, in the following we show the experiment we conducted where we investigate if the graphical notation is easy to understand and easy to use.

First we collected a set of three process models from practice of different levels of difficulty. We took the reference for these models from[87], and they were translated and slightly modified. Then we defined a set of six PPIs for every process model (the material used for the experiment can be seen in Appendix §C). Then, following the process described in[52], we construct a questionnaire that measured the following variables:

- Theory: Students made a self-assessment of theoretical knowledge in business process field and the BPMN notation on a five point ordinal scale,
- Practice: Students made a self-assessment of practical experience in business process field and the BPMN notation on a five point ordinal scale,
- BPMN previous knowledge test: A test with 17 questions was proposed to the students. First 5 questions were related to general BPMN concepts; then the students were asked to model some situations using BPMN in 4 questions; finally, two process models were provided and four questions related to each of them were asked.
- Perceived: For each model students made an assessment of the perceived difficulty of the model,
- Score: For each model and for each PPI, students answered a closed question. In half of the questions we gave the subject a PPI modelled and she had to select the correct definition from three possible proposed answers, and in the other half, the subject was provided a description in natural language and she had to choose one of the three PPI's models proposed; from the answers we calculated SCORE as the sum of correct answers to serve as an operationalization of understandability.
- Time: For each question, students annotated the time consumed in answering (specifying hour, minute and second).

The questionnaire was filled out in class settings at two different masters (MSC) of the university of Seville by 68 students in total. At the time of the experiment, students were following or completing courses of BPMN. The motivation for the students was that they were informed that the questionnaire would be a good exam preparation.

Results

- Analysis of comprehensibility of PPINOT graphical notation

  An average of 82.72 correct answers was obtained. That is a sign of the good understandability of the notation. In particular, for the type of questions where a PPI model had to be selected (type one from now), the average of correct answers was 76.84, while for the type of questions where a PPI definition had to be selected (type two from now), the average of correct answers was 88,60. From these results it can be deduced the fact that natural language is more ambiguous and leads to errors, when using it, it is necessary to define very precisely what has to be graphically represented (this is quite similar to the eternal problem of understanding customer preferences during the requirements elicitation).

- Impact of individual variables such as BPMN previous knowledge or difficulty of the process model

  Figure §10.4 shows that there exist some differences in the number of correct answers depending on the previous level of BPMN knowledge, although they are not very significant to stablish the imposibility of understanding PPINOT notation without a high knowledge of BPMN. (Low level: 6 students, medium level: 22, high level: 35, 5 students did not fulfilled the BPMN test)

  Regarding the influence of the complexity of the process on the number of correct answers, and thus on the understandability of the notation, Figure §10.5 reflects the results obtained. The percentage of correct answers decreases as the complexity of the process model increases, although the average of percentage of correct answers for the medium to complex process model case is still high (71,81%)

10.3.2 Case studies

Our PPINOT graphical notation presented in this paper has been or is being used in several real scenarios in order to model their internal PPIs.

IT Department of the Andalusian Health Service

One of the scenarios where our proposal has been applied, and actually the initial motivating scenario for our research, is the IT Department of the Andalusian Health Service. The main factor that made this organisation to be interested in our proposal was the existing partial view of the different departments/roles in charge of, on the one hand, the modelling and execution of such a process, and on the other hand the definition and consecution of goals and its associated indicators. To address this issue, from the set of processes defined for the IT Service

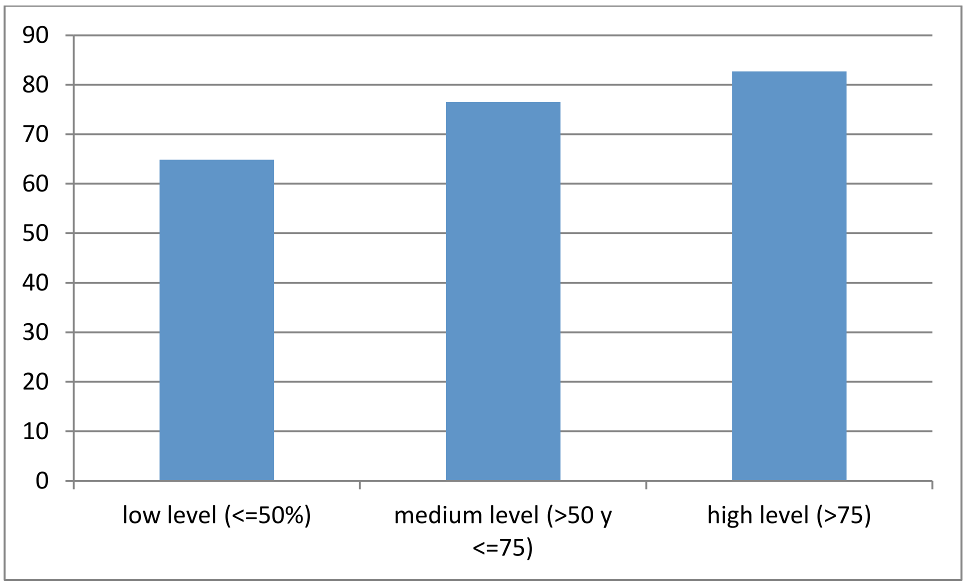


Figure 10.4: Correct answers in % according to the previous BPMN knowledge

Management (ITSM) of this organisation, we focused on the Request For Change (RFC) management process (the one introduced in Section §5.2 as motivating scenario). Concretely, we were provided with two separate documents, the process model on he one hand, and a document containing the definition of the associated PPIs in natural language (see Table §5.1) on the other hand: nobody had the comprehensive view of both worlds and it was really complicated to maintain the coherence across them, since changes in one document were not reflected in the other and vice-versa, thus leading to inconsistencies between them. We refined the RFC management process model and graphically modelled its associated PPIs using PPINOT tool. We needed to refine our PPINOT metamodel several times until being able to define the whole set of PPIs.

Another limitation they have to deal with is the high number of suppliers they have. This makes it more difficult to obtain the appropriate information required for the definition of PPIs. To cope this limitation, we provided them our PPI-templates and L-patterns. In this way, the different suppliers can gather the required information simply fulfilling the blanks, and using natural language, taking advantage of the underpinning metamodel. We are still waiting the results from this experience, in order to refine t PPI-templates and L-patterns if ot were necessary.

Information and Communication Service of the University of Seville

The second scenario takes place in the Information and Communication Service of the University of Seville. In this case, a process orientation was to be adopted in this organisation, and, apart from their business process modelling, they also required a definition and modelling

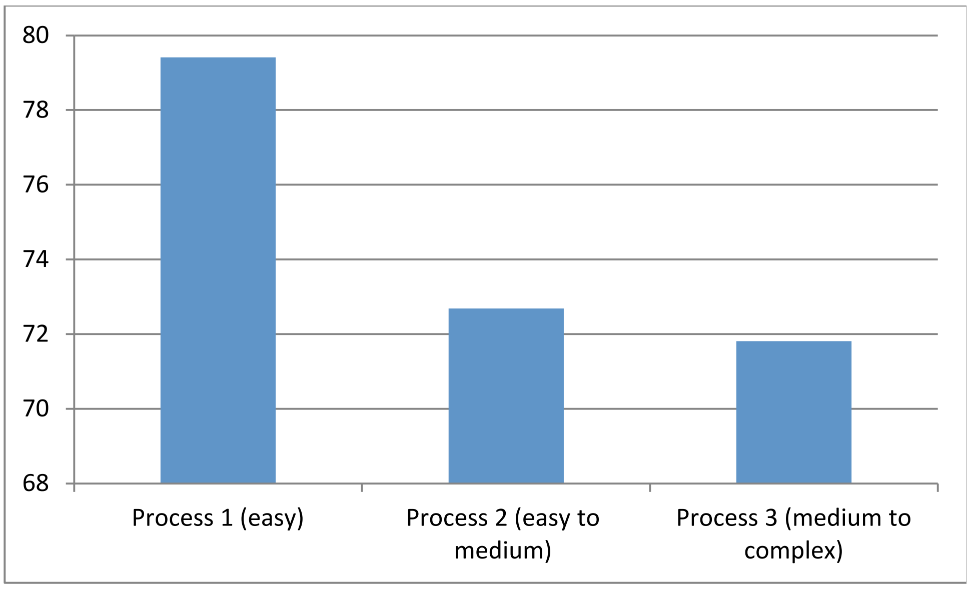


Figure 10.5: Correct answers in % according to the difficulty of the process model

of PPIs. A set of four processes have been modelled (user management process, process or the personal information management, process of the management of incidents sending emails, process of the management of incidents receiving emails) and their corresponding PPIs are being modelled using PPINOT Graphical Notation (aproximately 5 PPIs per process). Furthermore, we plan to refine within the context of this scenario our PPI templates.

ITIL

The Information Technology Infrastructure Library (ITIL) is a set of good practices for IT service management that focuses on aligning IT services with the needs of the business. It describes procedures (or processes), tasks and checklists to allow the organisation to establish a baseline form which it can plan, implement, and measure. In this context, for two of these processes, Incidents Management and Changes Management, their PPIs have been modelled using PPINOT. Concretely, for the Changes Management process, a set of 10 `metrics` are defined in ITIl; they have been refined and translated to PPINOT graphical Notation, obtaining 15 measures that define 8 PPIs. For the case of the Incidents Management process, from the set of 10 `metrics` defined in ITIL, 16 PPIs defined by 24 measures.

Academic case studies

Finally, this tool suite has been used in two universities: Universidad de Sevilla and Hasso Plattner Institute (Univeristät Potsdam).

First, it has been used in two courses of two masters (MSc) of the University of Seville, the Information Technology and Communication Management (ITCM) MSc, and the Software Engineering and Technology MSc. In the first one, a set of 40 students were asked to modell at least two PPIs for a real Business Process (modelled in BPMN) using PPINOT. In the second one, it was a set of 25 students who used our graphical editor for their classes. We used the feedback we obtained during these teaching activities in order to refine our approach, to make it affordable for final users.

Second, PPINOT Graphical Editor and Analyser are being used by the Business Process Technology research group and their students in the Hasso Plattner Institute of Potsdam University. They are being used in both, teaching and research activities.

10.4 Summary

In this chapter we have presented PPINOT tool suite, an easy-to-use tool suite for the definition and automated analysis of PPIs. PPINOT satisfies the necessity of a tool that, on the one hand, fills the visual gap between BPs and their corresponding PPIs by allowing the modelling of such PPIs together with the corresponding BP, while also allowing a definition of such PPIs in a structured natural language by means of templates and patterns; and on the other hand, automates the error-prone and tedious task of analyse PPIs. Furthermore, a mechanism to extract the PPI values from an open source BPMS has been developed. Another aspect covered in this chapter is the evaluation of our approach based on the real scenarios where it has been applied. We have also presented the results obtained in an experiment conducted in order to measure its usability.

We summarise the problems we overcome with this contribution by means of the Kiviat diagram depicted in Figure §10.6.

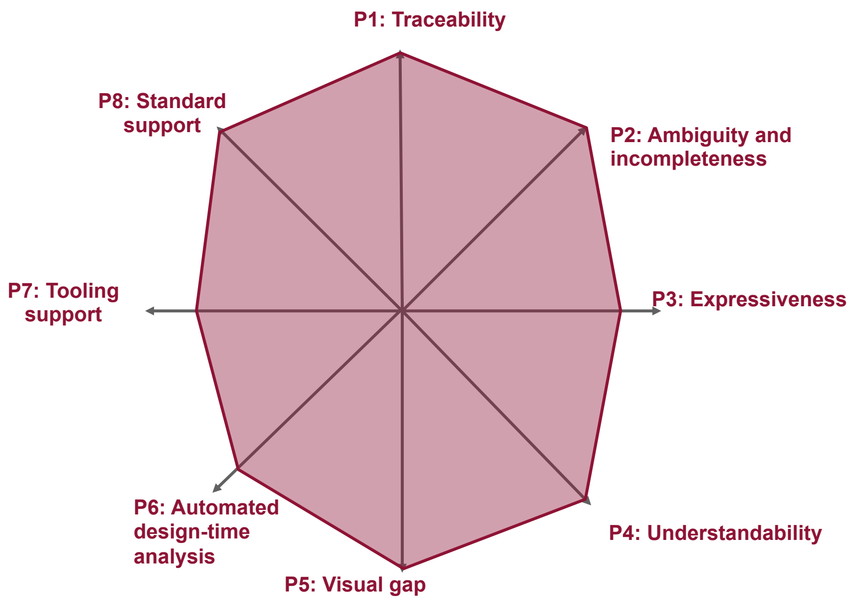


Figure 10.6: Kiviat diagram for the problems overcome by PPINOT Tool Suite

Part IV

Final Remarks

Conclusions and Future Work

All progress is precarious, and the solution of one problem brings us face to face with another problem.

Martin Luther King Jr. (1929 – 1968), US civil rights leader and clergyman

11.1 Conclusions

In this dissertation we have shown that:

We provide solutions to improve both the design and the instrumentation phase of the PPIM lifecycle.

In this dissertation we have presented a set of techniques and tools to define, represent and automatically analyse, at design-time, PPIs. These contributions are the result of an extensive analysis of a variety of PPIs defined by different organisations, a careful study of the related literature and the application of the knowledge gained in previous experiences with the automated analysis in other areas like feature models and SLAs. Our main results are a metamodel for the definition of PPIs, PPINOT metamodel; a graphical notation and a set of templates and L-PATTERNs based on this metamodel that make available to non-technical users the definition of PPIs at the right level of abstraction they require; two families of analysis operations, that leverages the benefits of DL reasoners to extract information at design-time from the definition of PPIs; and a software tool suite that provides support to the previous techniques.

With these contributions we overcome most of the problems identified at the beginning of this dissertation (Chapter §5). We summarise this information in Figure §11.1 by providing the whole picture of the Kiviat diagrams presented in the previous chapters.

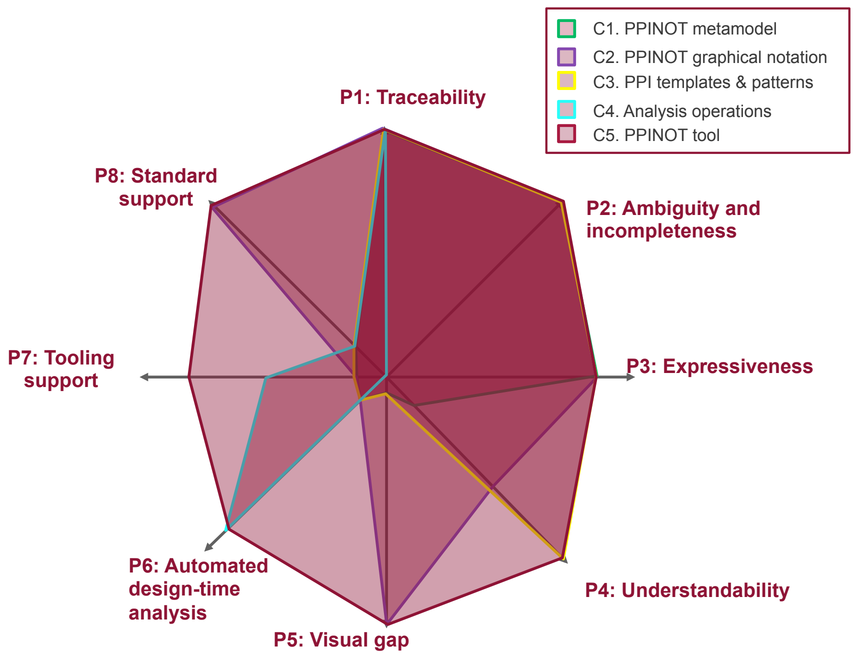


Figure 11.1: Kiviat diagram for the problems overcome by the set of contributions presented in this dissertation

To show the feasibility of our approach, we have presented its application to several real scenarios, the results of an experiment conducted and the interest of several institutions in our PPINOT tool. These results support the success of our dissertation in providing a solid background for the definition, representation and automated support for the design-time analysis of PPIs, contributing to the progress of the discipline and leading it to a new level of maturity.

11.2 Support for Results

Some of the results shown in this thesis have been already published in scientific forums. Figure §11.2 summarises these publications, grouping them by two dimensions: type and topic. Five types are defined: book, journal, conferences, tool-demos and workshops. Furthermore some types of publications have a quality level associated, JCR for journals, and CORE and MAS (Microsoft Academic Search) rankings for conferences. Regarding the topic dimension, three lines are depicted, the first one for those publications related to the metamodel and the automated analysis approach, the second one for those elated to the graphical notation and the tool suite, and the third one for those related to the templates approach.

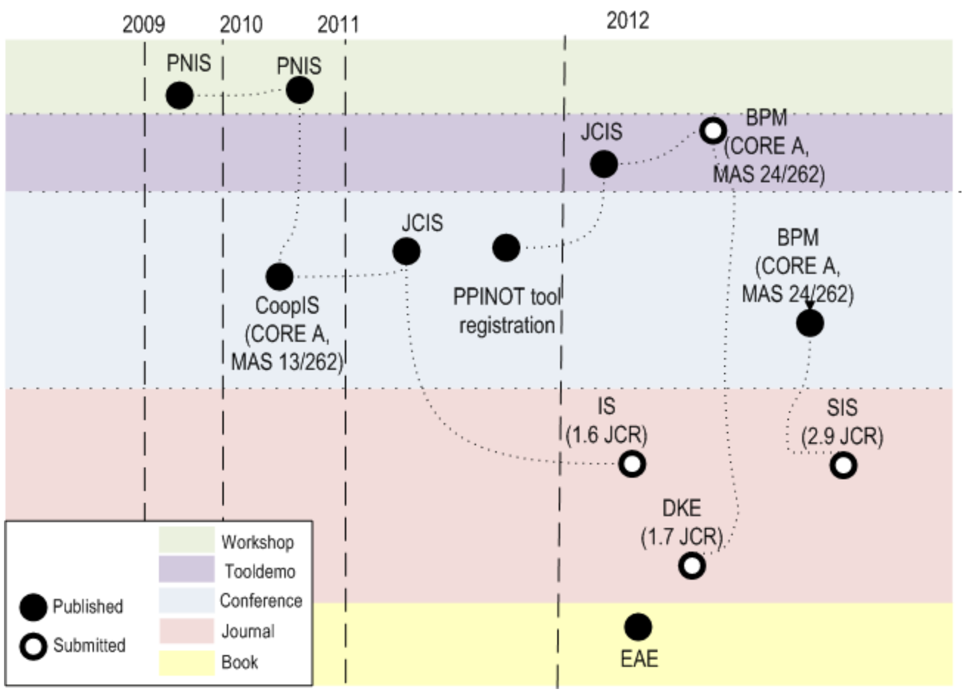


Figure 11.2: Publications grouped by type and topic

11.3 Discussion, limitations and extensions

We next discuss some of the decisions that we have made in this dissertation highlighting its main limitations and possible extensions.

- **PPINOT metamodel expressiveness.** The metamodel presented in this dissertation was designed to allow the definition of all the possible PPIs. Nevertheless, we left without consideration those PPIs related to resources. This was initially motivated by the lack of support of resource definition in BPMN. However, due to the need identified in the related works and the real scenarios, and based on the knowledge acquired in this context in our research group, we claim the definition of such kind of PPIs can benefit from the techniques for the PPI definition and automated analysis presented in this dissertation.

  Extension: Extend the PPINOT metamodel and graphical notation and the analysis operations as well as the supporting tool to deal with resource-aware PPIs.

- **Performance of DL as the formal foundation.** The formalisation of PPINOT metamodel is carried out by using DL. However, the usage of this formalism in real life tools can bring the disadvantage of a low performance, since this is the main reason why DL reasoners are criticised.

Extension: Look for another kind of formalism for the metamodel that overcome this problem without loosing the advantage of allowing an automated analysis at design time.

- **Tool support.** The contributions presented in this paper have been implemented into PPINOT tool, supporting the definition and automated design-time analysis of PPIs. Nevertheless this tool presents some limitations to be overcome:

  – the automatic generation of PPI templates from their corresponding graphical representations and viceversa.

  – the syntactical validation of PPI models.

  – the implementation of the rest of analysis operations that are not supported yet.

  – the improvement of the user web interface for the analysis feature.

  Extension: Extend the current Tool suite with the aforementioned features.

Figure §11.3 illustrates by means of the Kiviat diagrams we used along the whole dissertation, how these extensions would completely overcome those problems already commented.

11.4 Other future work

In addition to the aforementioned extensions to our work, we also identify a number of open research topics to be explored in our future research, namely:

- **Green PPIs.** The sustainability of organisations' business activities is gaining increasing importance. In this context, energy-related metrics called Green Performance Indicators (GPIs) are being defined to monitor the energy consumption level of such activities. Extending our PPINOT metamodel to allow the definition of this GPIs in the context of process-oriented organisations will certainly be part of our research work.

- **Automated Analysis post-mortem.** In this dissertation we have presented several families of operations to automatically analysed PPI definitions at design time. This information could be improved by using knowledge obtained during the BP execution. In this

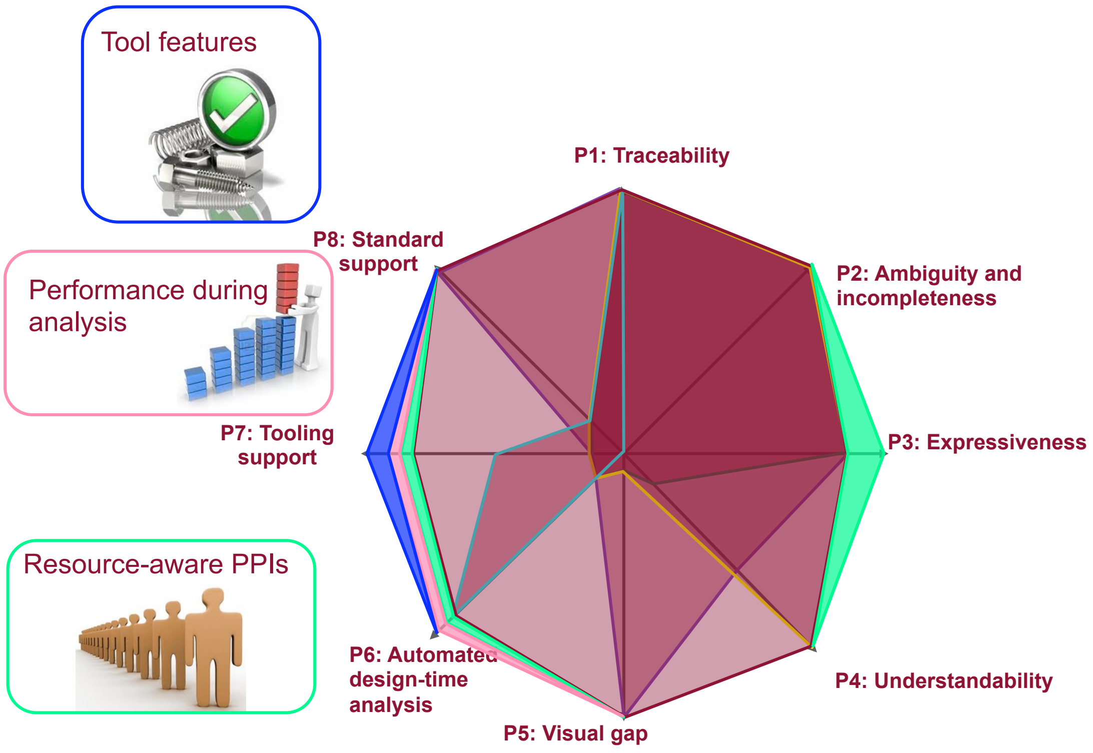


Figure 11.3: Kiviat diagram for the problems overcome by the catalogue of automated analysis operations

way, we could leverage the features of our approach and apply it to a post-mortem automated analysis, taking also advantage of the existing solutions using process mining and other business intelligence techniques.

- **Configurable PPIs.** Some business processes recur in similar form from one company to another, and it does not make sense that every time a company engages in modelling and re-designing one of these BPs, it did so "from scratch". In order to deal with this variability configurable BPs are used. "A configurable process model is a model that captures multiple variants of a business process in a consolidated manner" [83]. This context could highly benefit from our approach in order to provide PPI definitions and automated analysis operations for these configurable BPs. To do so, an extension of our techniques and tools to define *configurable PPIs* is required.

- **PPIs in the context of web services and their SLAs.** The negotiation of non-functional properties between web service provider and consumer can be agreed a priori by specifying SLAs. WS-agreement is a protocol for the specification of these SLAs. Nevertheless, it does not provide complete specifications of these agreements, but templates with blanks to be fulfilled. Within these templates there exists a section where metrics are defined.

For the case of complex web services, such as Amazon Web Services (S3, EC2, etc.), this metrics section becomes quite complex to be specified. We believe that the use of our approach in this context for the definition of this metrics section could be very appropriate.

- **Integration of PPINOT Tool into a BPMS.** The integration of the PPI definition and automated design-time analysis support into a BPMS will certainly bring great benefits, among which we can highlight the comprehensive support of the PPI lifecycle inside the BPM lifecycle.

Part V

Appendices

A

DL In a nutshell

A.1 Description Logics

DL are logics that serve primarily for formal description of *concepts*, *roles* (relations between the concepts) and *individuals* (instances of the concepts)[1]. Semantically, they are found on predicate logic, but their language is formed so that it would be enough for practical modeling purposes and also so that the logic would have good computational properties such as decidability.

Knowledge representation systems based on DL consist of two components: TBox and ABox. The TBox describes *terminology*, i.e., the ontology in the form of *concepts* and *property* definitions and their relationships, while the ABox contains *assertions* about individuals using the terms from the ontology.

"The basic form of declaration in a TBox is a concept *definition*, that is, the definition of a new concept in terms of other previously defined concepts" [61]. For example, a $DerivedMultiInstanceMeasu$ can be defined as a $DerivedMeasure$ that is a MultiInstanceMeasure by writing this declaration:

$$DerivedMultiInstanceMeasure \equiv DerivedMeasure \sqcap MultiInstanceMeasure$$

"Such a declaration is usually interpreted as a logical equivalence, which amounts to providing both sufficient and necessary conditions for classifying an individual as a DerivedMultiInstanceMeasure. In DL knowledge bases, therefore, a terminology is constituted by a set of concept definitions of the above form.

However, there are some important common assumptions usually made about DL terminologies:

- only one definition for a concept name is allowed;

[1]Sometimes OWL terms *classes*, *properties* and *objects* will be used to refer to DL terms *concepts*, *roles* and *individuals*, respectively.

- definitions are acyclic in the sense that concepts are neither defined in terms of themselves nor in terms of other concepts that indirectly refer to them.

This kind of restriction implies that every defined concept can be expanded in a unique way into a complex expression containing only atomic concepts by replacing every defined concept with the right-hand side of its definition.

In particular, the basic task in constructing a terminology is *classification*, that consists in placing a new concept expression in the proper place in a taxonomic hierarchy of concepts.

The ABox contains extensional knowledge about the domain of interest, that is, assertions about individuals, usually called membership assertions" [61]. For example,

$$DerivedMeasure \sqcap MultiInstanceMeasure(PERCENTAGE-OF-RFCS-APPROVED)$$

states that the individual PERCENTAGE-OF-RFCS-APPROVED is a MultiInstanceMeasure of type DerivedMeasure. Given the above definition of DerivedMultiInstanceMeasure, one can derive from this assertion that PERCENTAGE-OF-RFCS-APPROVED is an instance of the concept DerivedMultiInstanceMeasure. Similarly,

$$isCalculatedUsing(PERCENTAGE-OF-RFCS-APPROVED, RFCs-APPROVED)$$

specifies that PERCENTAGE-OF-RFCS-APPROVED is calculated using RFCs-APPROVED. Assertions of the first kind are also called concept assertions, while assertions of the second kind are also called role assertions.

"The basic reasoning task in an ABox is *instance checking*, which verifies whether a given individual is an instance of (belongs to) a specified concept. Although other reasoning services are usually considered and employed, they can be defined in terms of instance checking. Among them we find *knowledge base consistency*, which amounts to verifying whether every concept in the knowledge base admits at least one individual; *realization*, which finds the most specific concept an individual object is an instance of; and *retrieval*, which finds the individuals in the knowledge base that are instances of a given concept. These can all be accomplished by means of instance checking" [61]. Hence, a distinguished feature in DL is the emphasis on reasoning as a central service, allowing thus to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base.

A.1.1 Description Languages

"Elementary descriptions are *atomic concepts* and *atomic roles*. Complex descriptions can be built from them inductively with *concept constructors*" [2]. There are many varieties of Description Languages, and they are distinguished by the constructors they provide. We focus on the family of $\mathcal{AL}$-*languages*. The language $\mathcal{AL}$ (attributive language) allows for atomic negation, concept intersection, universal restrictions and limited existential quantification. More expressive languages can be obtained by adding further constructors and axioms to $\mathcal{AL}$. In the following we list the possible extensions:

$\mathcal{F}$: Functional properties

$\mathcal{E}$: Full existential qualification

$\mathcal{U}$: Concept union

$\mathcal{C}$: Complex concept negation

$\mathcal{H}$: Role hierarchy

$\mathcal{R}$: Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness

$\mathcal{O}$: Nominals

$\mathcal{I}$: Inverse properties

$\mathcal{N}$: Cardinality restrictions

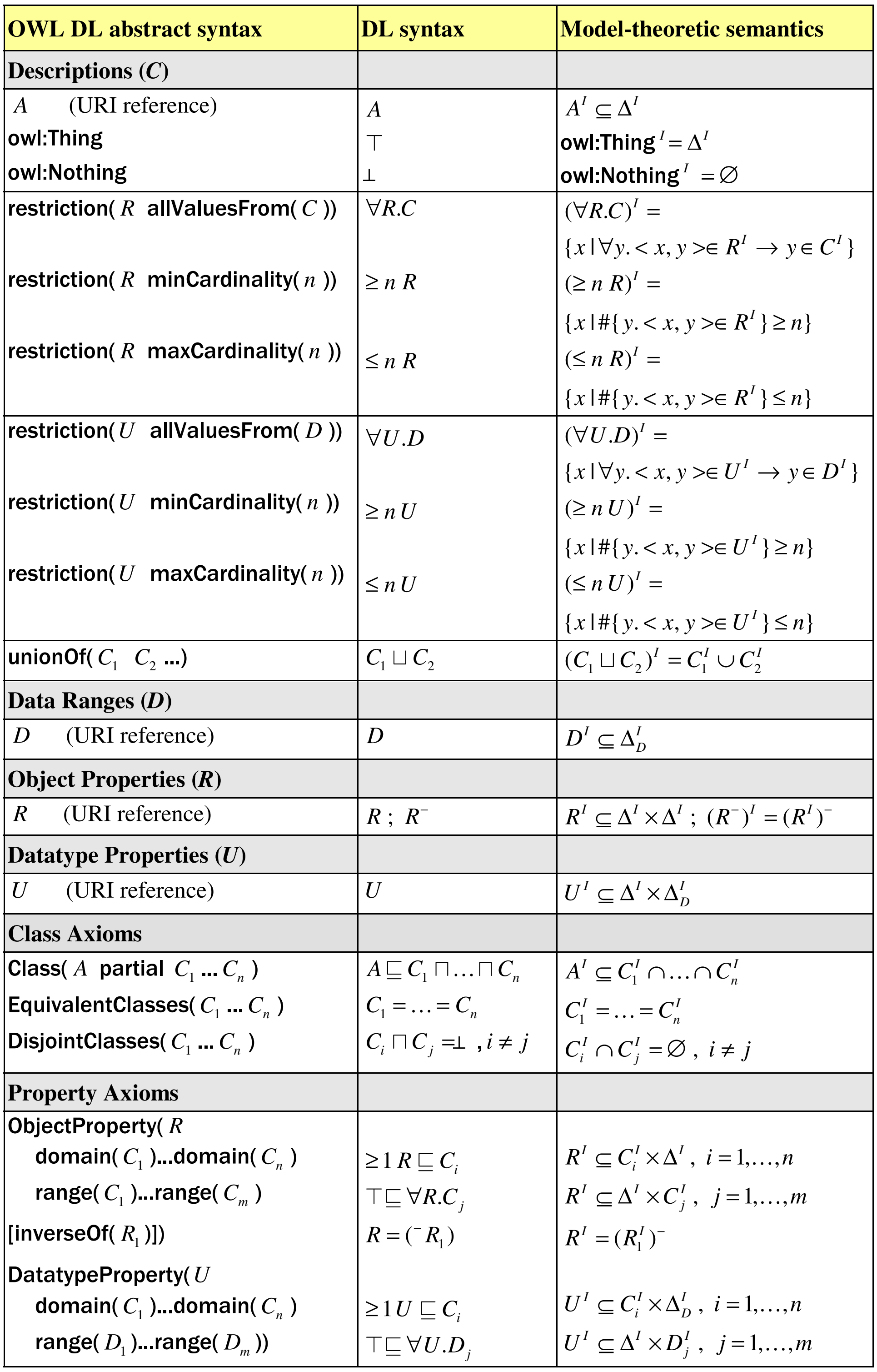
$\mathcal{Q}$: Qualified cardinality restrictions

$^{(\mathcal{D})}$: Use of datatype properties, data values or data types.

Of interest for us is the abbreviation $\mathcal{S}$, that stands for $\mathcal{ALC}$ logic with transitive role. This is the basis for the logics of OWL.

A.2 OWL

The Web Ontology Language (OWL) is a knowledge representing scheme designed specifically for use on the semantic web; it exploits existing web standards (XML and RDF), adding the familiar ontological primitives of object and frame based systems, and the formal rigor of



OWL DL abstract syntax	**DL syntax**	**Model-theoretic semantics**
Descriptions (*C*)		
A (URI reference)	A	$A^I \subseteq \Delta^I$
owl:Thing	$\top$	owl:Thing$^I = \Delta^I$
owl:Nothing	$\bot$	owl:Nothing$^I = \varnothing$
restriction(R allValuesFrom(C))	$\forall R.C$	$(\forall R.C)^I = \{x \mid \forall y. <x,y> \in R^I \rightarrow y \in C^I\}$
restriction(R minCardinality(n))	$\geq n\ R$	$(\geq n\ R)^I = \{x \mid \#\{y. <x,y> \in R^I\} \geq n\}$
restriction(R maxCardinality(n))	$\leq n\ R$	$(\leq n\ R)^I = \{x \mid \#\{y. <x,y> \in R^I\} \leq n\}$
restriction(U allValuesFrom(D))	$\forall U.D$	$(\forall U.D)^I = \{x \mid \forall y. <x,y> \in U^I \rightarrow y \in D^I\}$
restriction(U minCardinality(n))	$\geq n\ U$	$(\geq n\ U)^I = \{x \mid \#\{y. <x,y> \in U^I\} \geq n\}$
restriction(U maxCardinality(n))	$\leq n\ U$	$(\leq n\ U)^I = \{x \mid \#\{y. <x,y> \in U^I\} \leq n\}$
unionOf(C_1 C_2 ...)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
Data Ranges (*D*)		
D (URI reference)	D	$D^I \subseteq \Delta_D^I$
Object Properties (*R*)		
R (URI reference)	R ; R^-	$R^I \subseteq \Delta^I \times \Delta^I$; $(R^-)^I = (R^I)^-$
Datatype Properties (*U*)		
U (URI reference)	U	$U^I \subseteq \Delta^I \times \Delta_D^I$
Class Axioms		
Class(A partial C_1 ... C_n)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$	$A^I \subseteq C_1^I \cap \ldots \cap C_n^I$
EquivalentClasses(C_1 ... C_n)	$C_1 = \ldots = C_n$	$C_1^I = \ldots = C_n^I$
DisjointClasses(C_1 ... C_n)	$C_i \sqcap C_j = \bot$, $i \neq j$	$C_i^I \cap C_j^I = \varnothing$, $i \neq j$
Property Axioms		
ObjectProperty(R		
domain(C_1)...domain(C_n)	$\geq 1\ R \sqsubseteq C_i$	$R^I \subseteq C_i^I \times \Delta^I$, $i = 1,\ldots,n$
range(C_1)...range(C_m)	$\top \sqsubseteq \forall R.C_j$	$R^I \subseteq \Delta^I \times C_j^I$, $j = 1,\ldots,m$
[inverseOf(R_1)])	$R = (^- R_1)$	$R^I = (R_1^I)^-$
DatatypeProperty(U		
domain(C_1)...domain(C_n)	$\geq 1\ U \sqsubseteq C_i$	$U^I \subseteq C_i^I \times \Delta_D^I$, $i = 1,\ldots,n$
range(D_1)...range(D_m))	$\top \sqsubseteq \forall U.D_j$	$U^I \subseteq \Delta^I \times D_j^I$, $j = 1,\ldots,m$

Figure A.1: DL summary

DL. As exemplified in Table §A.1 and Table §A.2 [2] [3], OWL consists a rich set of knowledge

[2]In both tables and in Chapter §9 a syntax commonly used for DLs [3] is utilised

representation constructs that can be used to formally specify PPI-domain knowledge, which in turn can be exploited by DL reasoners for purposes of inferencing, i.e., deductively inferring new facts from knowledge that is explicitly available [6]. As stated in [6], the logical basis of the language means that reasoning services can be provided in order to make OWL described resources more accessible to automated processes thereby allowing one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base. From a formal point of view, “OWL can be seen to be equivalent to a very expressive DL, with an OWL ontology corresponding to a DL terminology (TBox) whereas instance data pertaining to the ontology making up the assertions (ABox)”.

[3] note that this tables are no complete, but contain only those elements useful for us during the development of this thesis work

Axiom	**DL Syntax**	**Example**
Sub class	$C_1 \sqsubseteq C_2$	$BaseMeasure \sqsubseteq InstanceMeasure \sqsubseteq MeasureDefinition$
Equivalent class	$C_1 \equiv C_2$	$MeasureDefinition \equiv InstanceMeasure \sqcup ProcessMeasure$
Disjoint with	$C_1 \sqsubseteq \neg C_2$	$InstanceMeasure \sqsubseteq \neg ProcessMeasure$
Same Individual	$x_1 \equiv x_2$	
Different from	$x_1 \sqsubseteq \neg x_2$	$PPI1 \sqsubseteq \neg PPI2$
Sub property	$P_1 \sqsubseteq P_2$	$indirectlyAggregates \sqsubseteq aggregates$
Equivalent property	$P_1 \equiv P_2$	
Inverse	$P_1 \equiv P_2^-$	$isDefinedOver \equiv isUsedBy^-$
Transitive property	$P^+ \sqsubseteq P$	$indirectlyAggregates^+ \sqsubseteq indirectlyAggregates$
Functional property	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1 isDefinedOver$
Inverse functional property	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1 isUsedBy^-$

Table A.1: OWL DL (class and property) axioms and facts

Constructor	**DL Syntax**	**Example**
Intersection	$C_1 \sqcap \cdots \sqcap C_n$	$MeasureDefinition \sqcap DataMeasure$
Union	$C_1 \sqcup \cdots \sqcup C_n$	$TimeMeasure \sqcup DataMeasure$
Complement	$\neg C$	$\neg DerivedMeasure$
One of	$x_1 \sqcup \cdots \sqcup x_n$	$PPI1 \sqcup PPI3 \sqcup PPI8$
All values from	$\forall P.C$	$\forall isDefinedOver.MeasureDefinition$
Some values	$\exists P.C$	$\exists isUsedBy.PPI \sqsubseteq aggregates$
Has value	$P.x$	$isUsedBy.PPI1$
Max cardinality	$\leq nP$	$\leq 1 appliesTo$
Min cardinality	$\geq nP$	$\geq 1 isCalculated$

Table A.2: OWL class constructors

PPI Definitions for the RFC Management Process

B.1 PPI Specification according to PPINOT Metamodel

In the following we present the specification of the 9 PPIs contained in Table §5.1 using a HUTN-like[1] notation.

```
PPI{
   identifier: PPI1
   name: RFCs cancelled from RFCs registered
   relatedTo: RFCManagement
   goals: Improve customer satisfaction
   definition: DerivedMultiInstanceMeasure{
      function: (a/b)*100
      uses: a.refersTo: AggregatedMeasure {
         scale: float
         unitOfMeasure: RFC
         aggregationFunction: sum
         aggregates: CountMeasure{
            when: TimeInstantCondition{
               changesToState: cancelled
               appliesTo: dataObject RFC
            }
         }
      }
      uses: b.refersTo: AggregatedMeasure {
         scale: float
         unitOfMeasure: RFC
         aggregationFunction: sum
         aggregates: CountMeasure{
            when: TimeInstantCondition{
               changesToState: registered
               appliesTo: dataObject RFC
            }
         }
      }
   }
   target.upperBound: 4
   scope: ComposedFilter{
      And[
         ProcessStateFilter.processState: finished
```

[1]HUTN is the Human-Usable Textual Notation defined by the OMG in [66].

```
            Timefilter.periodicity: Weekly.dayOfWeek: Friday
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
}

PPI{
    identifier: PPI2
    name: Average time of committee decision
    relatedTo: RFCManagement
    goals: Reduce RFC time-to-response
    definition:AggregatedMeasure{
        scale: int
        unitOfMeasure: day
        aggregationFunction: average
        aggregates: TimeMeasure{
            from: TimeInstantCondition{
                changesToState: active
                appliesTo: activity Analyse in Committee
            }
            to: TimeInstantCondition{
                changesToState: completed
                appliesTo: activity Analyse in Committee
            }
        }
    }
    target.upperBound: one day
    scope: ComposedFilter{
        And[
            ProcessStateFilter.processState: finished
            Timefilter.periodicity: Weekly.dayOfWeek: Friday
        ]
    }
    responsible: Planning and Quality Manager
    informed:CIO
}

PPI{
    identifier: PPI3
    name: Corrective RFCs from approved RFCs
    relatedTo: RFCManagement
    goals: Improve customer satisfaction
    definition: DerivedMultiInstanceMeasure{
        function: (a/b)*100
        uses: a.refersTo: AggregatedMeasure {
            scale: int
            unitOfMeasure: RFC
            aggregationFunction: sum
            aggregates: ConditionMeasure{
                meets: DataPropertyCondition{
                    restriction: type of change = corrective
                    statesConsidered: approved
                    appliesTo: dataObject RFC
                }
            }
        }
        uses: b.refersTo: AggregatedMeasure {
            scale: int
            unitOfMeasure: RFC
            aggregationFunction: sum
            aggregates: CounttMeasure{
                when: TimeInstantCondition{
                    chagesToState: triggered
                    appliesTo: event report RFC approved
                }
            }
        }
    }
    target: SimpleTarget.upperBound: 2
    scope: ComposedFilter{
        And[
            ProcessStateFilter.processState: finished
            Timefilter.periodicity: Weekly.dayOfWeek: Friday
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
    comments: values up to 5 \% are reasonable
}

PPI{
    identifier: PPI4
    name: Perfective RFCs from approved RFCs
    relatedTo: RFCManagement
    goals: Improve customer satisfaction
    definition: DerivedMultiInstanceMeasure{
        function: (a/b)*100
        uses: a.refersTo: AggregatedMeasure {
            scale: int
            unitOfMeasure: RFC
            aggregationFunction: sum
            aggregates: ConditionMeasure{
                meets: DataPropertyCondition{
                    restriction: type of change = perfective
                    statesConsidered: registered
                    appliesTo: dataObject RFC
                }
            }
        }
        uses: b.refersTo: AggregatedMeasure {
            scale: int
            unitOfMeasure: RFC
            aggregationFunction: sum
            aggregates: CounttMeasure{
                when: TimeInstantCondition{
                    chagesToState: triggered
                    appliesTo: event report RFC approved
                }
            }
        }
    }
    target: SimpleTarget.upperBound: 4
    scope: ComposedFilter{
        And[
            ProcessStateFilter.processState: finished
            Timefilter.periodicity: Weekly.dayOfWeek: Friday
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
}

PPI{
    identifier: PPI5
    name: Average time of RFC analysis
    relatedTo: RFCManagement
    goals: Reduce RFC time−to−response
    definition:AggregatedMeasure{
        scale: int
        unitOfMeasure: day
        aggregationFunction: average
        aggregates: TimeMeasure{
            from: TimeInstantCondition{
                changesToState: active
                appliesTo: activity Analyse RFC
            }
            to: TimeInstantCondition{
                changesToState: completed
                appliesTo: activity Analyse RFC
            }
        }
    }
    target: simpleTarget.upperBound: two days
    scope: ComposedFilter{
    And[
            LastInstancesFilter.numberOrInstances: 100
            Timefilter.periodicity: Weekly.dayOfWeek: friday
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
}

PPI{
    identifier: PPI6
    name: Number of RFCs in analysis
    relatedTo: RFCManagement
    goals: Improve customer satisfaction. Reduce RFC time−to−response
    definition: AggregatedMeasure{
        scale: int
        unitOfMeasure: RFC
        aggregationFunction: sum
        aggregates: ConditionMeasure{
            meets: StateCondition{
                state: active
                appliesTo: activity analyse RFC
            }
        }
    }
    target: 2 RFCs
    scope: ComposedFilter{
    And[
            ProcessStateFilter.processState: active
            Timefilter.periodicity: Weekly.dayOfWeek: friday
        ]
    }

    responsible: Planning and Quality Manager
    informed: CIO
}

PPI{
    identifier: PPI7
    name: Number of RFCs per type of change
    relatedTo: RFCManagement
    goals:
    definition: AggregatedMeasure{
        scale: map
        unitOfMeasure: RFC
        aggregationFunction: sum
        aggregates: CountMeasure{
            when: TimeInstantCondition{
                changesToState: registered
                appliesTo: dataObject RFC
            }
        }
        isGroupedBy: type of change
    }
    target: ComposedTarget[
        corrective -20 RFCs
        evolutive -30 RFCs
        perfective -20 RFCs
    ]
    scope: ComposedFilter{
        And[
            ProcessStateFilter.processState: finished
            Timefilter.periodicity: Monthly.dayOfMonth: 25
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
    comments: the ideal situation is that corrective RFCs tend to zero
}

PPI{
    identifier: PPI8
    name: Number of RFCs per project
    relatedTo: RFCManagement
    goals:
    AggregatedMeasure{
        scale: map
        unitOfMeasure: RFC
        aggregationFunction: sum
        aggregates: CountMeasure{
            when: TimeInstantCondition{
                changesToState: registered
                appliesTo: dataObject RFC
            }
        }
        isGroupedBy: project
    }
    target: ComposedTarget[
        RR.HH−50 RFCs
        Diraya−60 RFCs
        Pharma−1 RFCs
    ]
    scope: ComposedFilter{
        And[
            ProcessStateFilter.processState: finished
            Timefilter.periodicity: Monthly.dayOfMonth: 25
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
}

PPI{
    identifier: PPI9
    name: Average lifetime of an RFC
    relatedTo: RFCManagement
    goals: Reduce RFC time−to−response
    definition:AggregatedMeasure{
        scale: float
        unitOfMeasure: day
        aggregationFunction: average
        samplingFrequency:
        aggregates: TimeMeasure{
            from: TimeInstantCondition{
                changesToState: triggered
                appliesTo: event Receive RFC
            }
            to: TimeInstanceCondition{
                changesToState: completed
                appliesTo:pool RFC management
            }
        }
    }
    target: 3
    scope: ComposedFilter{
        And[
            ProcessStateFilter.processState: finished
            Timefilter.periodicity: Monthly.dayOfMonth: 25
        ]
    }
    responsible: Planning and Quality Manager
    informed: CIO
}
```

B.2 PPINOT Graphical Representation

In this section we show the graphical representation of the 9 PPIs contained in Table §5.1 using the PPINOT graphical notation described in Chapter §7. Concretely, Figure §B.1 depicts the PPINOT graphical representation for PPIs from 1 to 4, and Figure §B.2 for PPIs from 5 to 8.

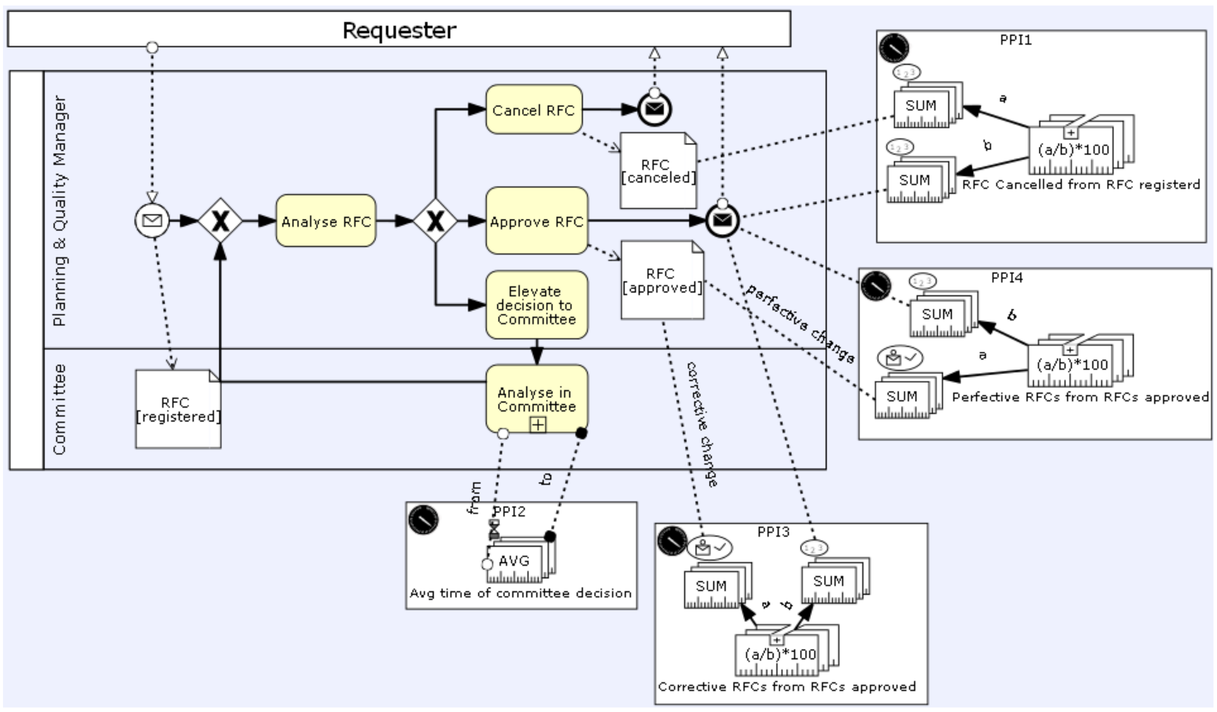


Figure B.1: PPINOT Graphical Representation of PPIs 1 to 4

B.3 PPI and Scope Templates

In this section we show the definition of the 9 PPIs contained in Table §5.1 using the PPI templates described in Chapter §7.

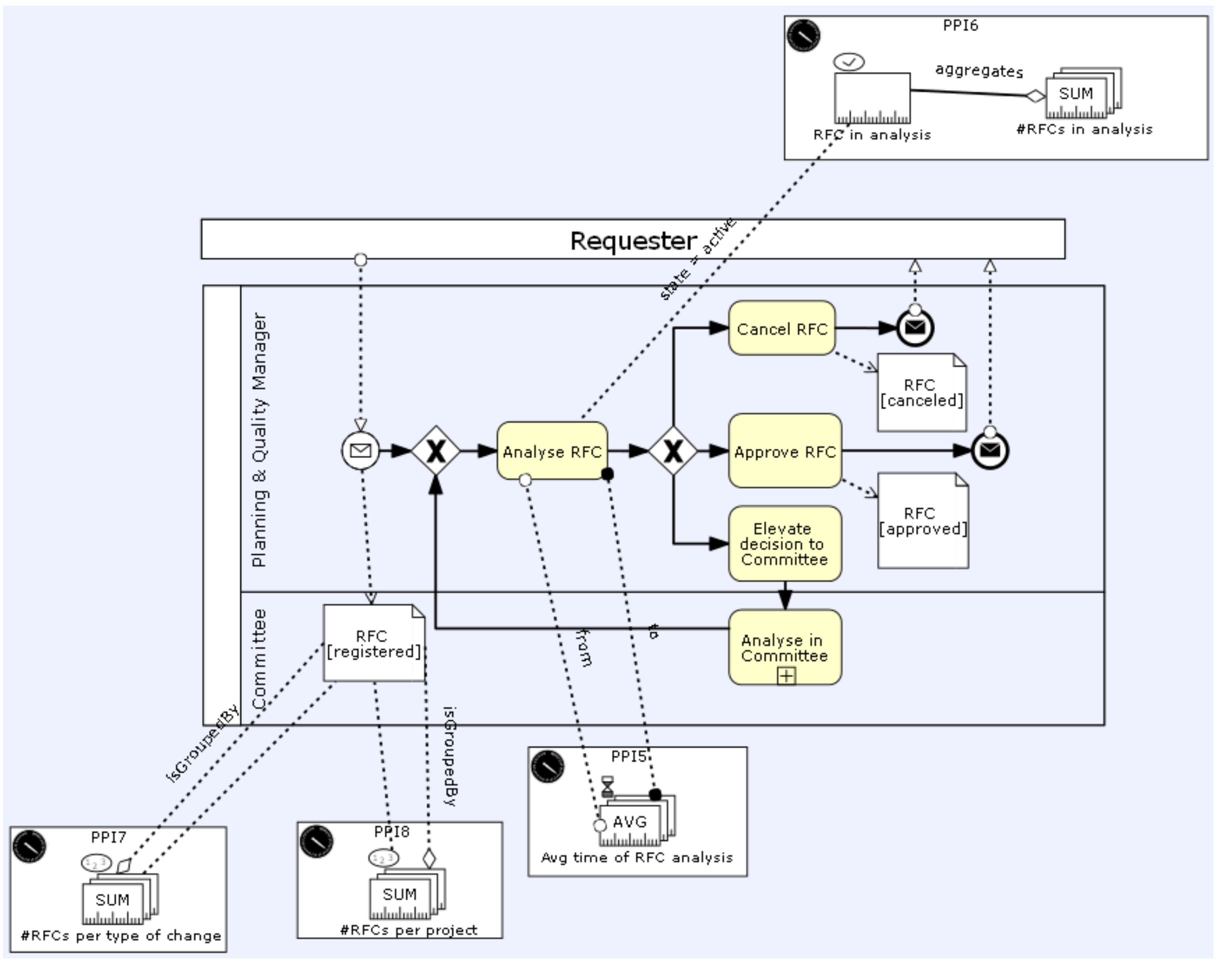


Figure B.2: PPINOT Graphical Representation of PPIs 5 to 8

PPI–001	**RFCs cancelled from RFCs registered**
Process	Request for change (RFC)
Goals	• BG–002: Improve customer satisfaction
Definition	The PPI is defined as *the mathematical function (a/b)*100 where a is the sum of the number of times dataObject* RFC *changes to state cancelled and b is the sum of the number of times dataObject* RFC *changes to state registered*
Unit of measure	none
Target	The PPI value must be less than or equal to *four %*
Scope	The process instances considered for this PPI are described in the scope S-1
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	

Table B.1: PPI template for PPI1 from Table §5.1

S-1	**Finished instances**
Conditions	This scope includes process instances in state finished
Periodicity	weekly on Friday

Table B.2: S-1 scope definition

PPI–002	**Average time of committee decision**
Process	Request for change (RFC)
Goals	• BG–014: Reduce RFC time–to–response
Definition	The PPI is defined as *the average of the duration between the time instants when the* Analyse in Committee *activity changes to state active and when the* Analyse in Committee *activity changes to state completed*
Unit of measure	day
Target	The PPI value must be less than or equal to *one working day*
Scope	The process instances considered for this PPI are described in the scope S-1
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	

Table B.3: PPI template for PPI2 from Table §5.1

PPI–003	**Corrective RFCs from RFCs approved**
Process	Request for change (RFC)
Goals	● BG–002: Improve customer satisfaction
Definition	The PPI is defined as *the mathematical function (a/b)*100 where a is the sum of the fulfillment of restriction* type of change = corrective *over the dataObject* RFC *in state approved and b is the sum of the number of times event* Report RFC approved *is triggered*
Unit of measure	none
Target	The PPI value must be less than or equal to *two %*
Scope	The process instances considered for this PPI are described in the scope S-1
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	*values up to 5 % are reasonable*

Table B.4: PPI template for PPI3 from Table §5.1

PPI–004	**Perfective RFCs from RFCs approved**
Process	Request for change (RFC)
Goals	• BG–002: Improve customer satisfaction
Definition	The PPI is defined as *the mathematical function (a/b)*100 where a is the number of times dataObject RFC in state approved satisfies: type of change = perfective and b is the sum of the number of times dataObject RFC changes to state approved*
Unit of measure	none
Target	The PPI value must be less than or equal to *four %*
Scope	The process instances considered for this PPI are described in the scope S-1
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	

Table B.5: PPI template for PPI4 from Table §5.1

PPI–005	**Average time of RFC analysis**
Process	Request for change (RFC)
Goals	• BG–014: Reduce RFC time–to–response
Definition	The PPI is defined as *the average of the duration between time instants when the Analyse RFC activity changes to state active and when the Analyse RFC activity changes to state completed*
Unit of measure	day
Target	The PPI value must be less than or equal to *two working days*
Scope	The process instances considered for this PPI are described in the scope S-2
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	*Most RFCs are created after 12:00.*

Table B.6: PPI template for PPI5 from Table §5.1

S-2	**Last 100 instances**
Conditions	This scope includes process instances, the last 100 ones
Periodicity	weekly on Friday

Table B.7: S-2 scope definition

PPI–006	**Number of RFCs in analysis**
Process	Request for change (RFC)
Goals	• BG–002: Improve customer satisfaction • BG–014: Reduce RFC time–to–response
Definition	The PPI is defined as *the number of times activity Analyse RFC is currently in state active*
Unit of measure	RFC
Target	The PPI value must be less than or equal to *two RFCs*
Scope	The process instances considered for this PPI are described in the scope S-3
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	

Table B.8: PPI template for PPI6 from Table §5.1

S-3	**Active instances**
Conditions	This scope includes process instances in state active
Periodicity	weekly on Friday

Table B.9: S-3 scope definition

PPI–007	**Number of RFCs per type of change**
Process	Request for change (RFC)
Goals	
Definition	The PPI is defined as *the sum of the number of times dataObject RFC changes to state registered and is grouped by type of change of RFC*
Unit of measure	RFC
Target	The PPI value must fulfill the following consraint: • for type of change = corrective must be less than or equal to *twenty RFCs* • for type of change = evolutive must be less than or equal to *thirty RFCs* • for type of change = perfective must be less than or equal to *twenty RFCs*
Scope	The process instances considered for this PPI are described in the scope S-4
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	*the ideal situation is that corrective RFCs tend to zero*

Table B.10: PPI template for PPI7 from Table §5.1

S-3	**Finished instances every month**
Conditions	This scope includes process instances in state finished
Periodicity	monthly on 25th

Table B.11: S-4 scope definition

PPI–008	**Number of RFCs per project**
Process **Goals**	Request for change (RFC)
Definition	The PPI is defined as *the sum of the number of times dataObject RFC changes to state registered and is grouped by project of RFC*
Unit of measure	RFC
Target	The PPI value must fulfill the following consraint: ● for project = RR.HH must be less than or equal to *fifty RFCs* ● for project = Diraya must be less than or equal to *sixty RFCs* ● for project = Pharma must be less than or equal to *one RFCs*
Scope	The process instances considered for this PPI are described in the scope S-4
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	

Table B.12: PPI template for PPI8 from Table §5.1

PPI–009	**Average lifetime of an RFC**
Process	Request for change (RFC)
Goals	• BG–014: Reduce RFC time–to–response
Definition	The PPI is defined as *the average of the duration between time instants when the* Receive RFC *event is triggered and when the* RFC management *pool changes to state completed*
Unit of measure	day
Target	The PPI value must be less than or equal to *three working days*
Scope	The process instances considered for this PPI are described in the scope S-4
Source	Event logs
Responsible	*Planning and quality manager*
Informed	*CIO*
Comments	*Most RFCs are created after 12:00.*

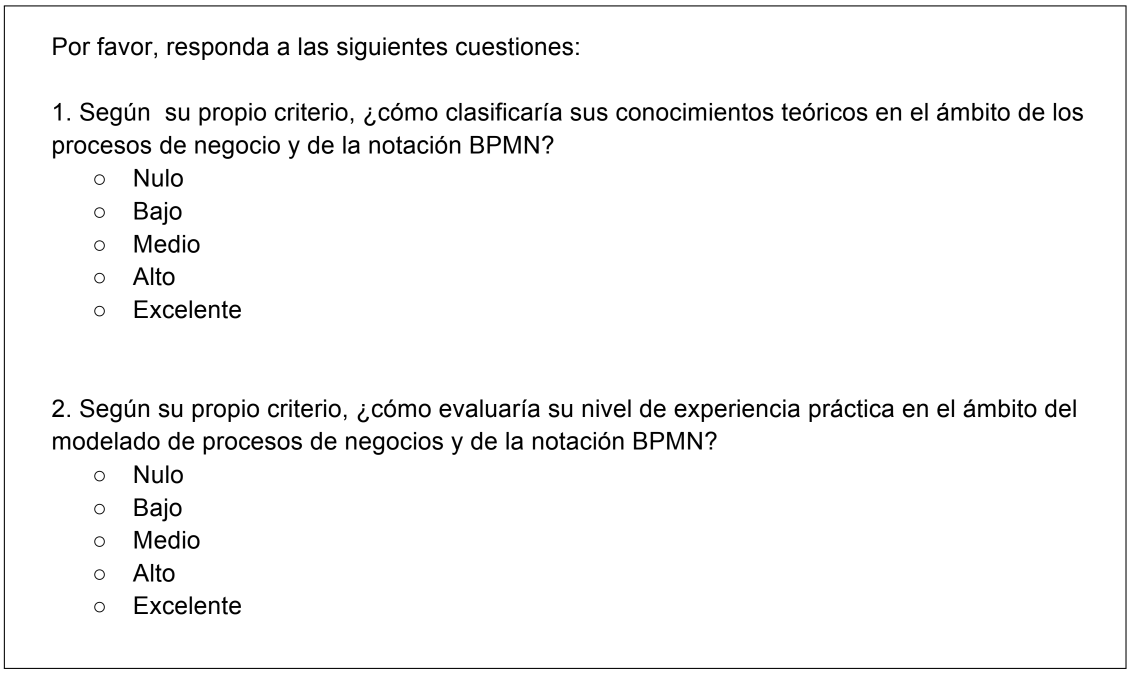
Table B.13: PPI template for PPI9 from Table §5.1

C Experiment Material

This appendix contains the original material provided to the participants of our experiment described in Section §10.3.1.

C.1 Theory and Practice Self-Assessment

In this section we show the self-assesment the experiment participants had to fulfill.



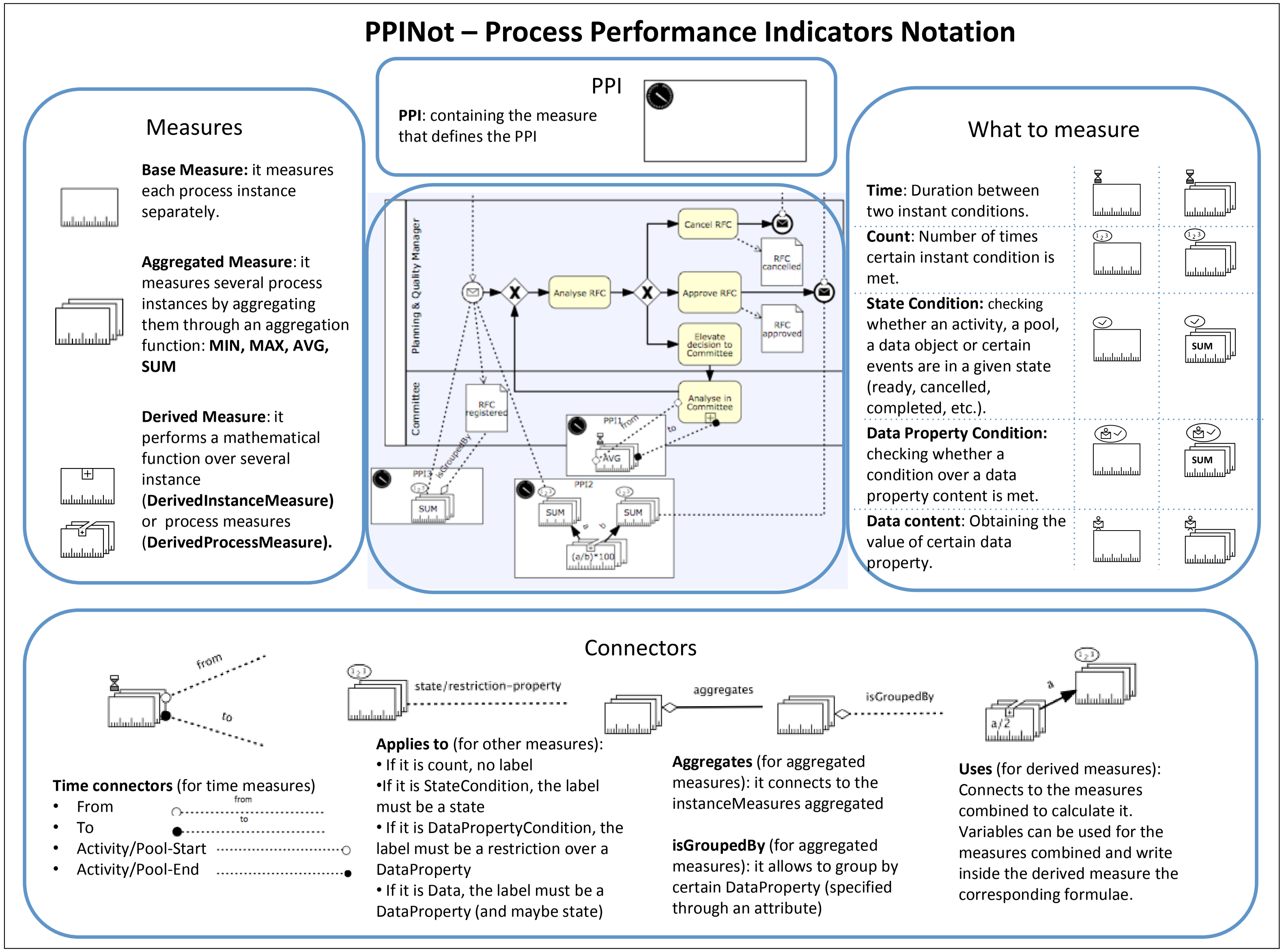
Por favor, responda a las siguientes cuestiones:

1. Según su propio criterio, ¿cómo clasificaría sus conocimientos teóricos en el ámbito de los procesos de negocio y de la notación BPMN?
    - ○ Nulo
    - ○ Bajo
    - ○ Medio
    - ○ Alto
    - ○ Excelente

2. Según su propio criterio, ¿cómo evaluaría su nivel de experiencia práctica en el ámbito del modelado de procesos de negocios y de la notación BPMN?
    - ○ Nulo
    - ○ Bajo
    - ○ Medio
    - ○ Alto
    - ○ Excelente

C.2 PPINOT Graphical Notation Poster

Figure §C.1 shows the poster of our PPINOT Graphical notation that was provided to the experiment participants for them to take it as reference during the PPI questionnaire.



PPINot – Process Performance Indicators Notation

Measures

Base Measure: it measures each process instance separately.

Aggregated Measure: it measures several process instances by aggregating them through an aggregation function: **MIN, MAX, AVG, SUM**

Derived Measure: it performs a mathematical function over several instance (**DerivedInstanceMeasure)** or process measures (**DerivedProcessMeasure).**

PPI

PPI: containing the measure that defines the PPI

What to measure

Time: Duration between two instant conditions.

Count: Number of times certain instant condition is met.

State Condition: checking whether an activity, a pool, a data object or certain events are in a given state (ready, cancelled, completed, etc.).

Data Property Condition: checking whether a condition over a data property content is met.

Data content: Obtaining the value of certain data property.

Connectors

Time connectors (for time measures)

- From
- To
- Activity/Pool-Start
- Activity/Pool-End

Applies to (for other measures):
• If it is count, no label
•If it is StateCondition, the label must be a state
• If it is DataPropertyCondition, the label must be a restriction over a DataProperty
• If it is Data, the label must be a DataProperty (and maybe state)

Aggregates (for aggregated measures): it connects to the instanceMeasures aggregated

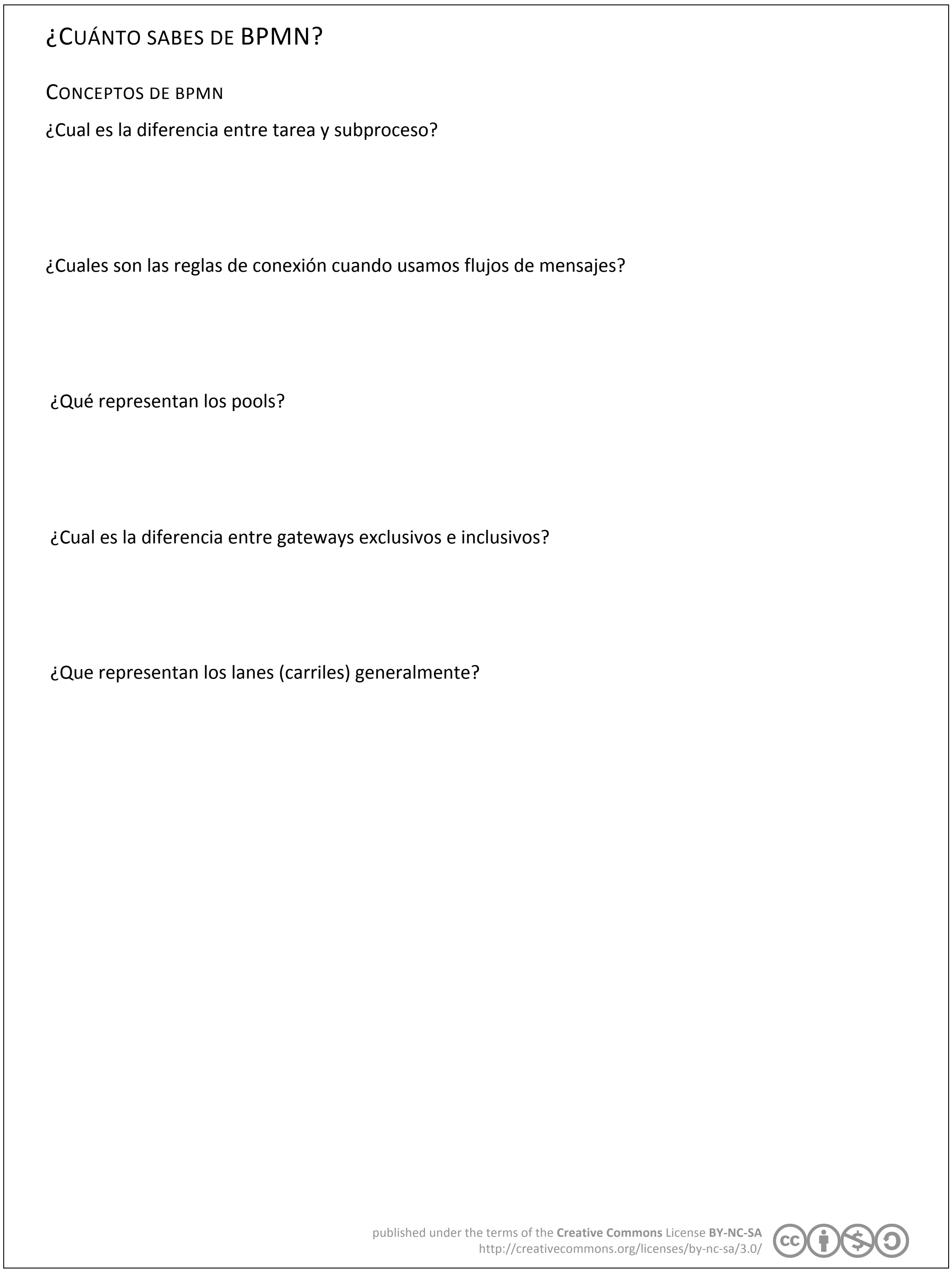
isGroupedBy (for aggregated measures): it allows to group by certain DataProperty (specified through an attribute)

Uses (for derived measures): Connects to the measures combined to calculate it. Variables can be used for the measures combined and write inside the derived measure the corresponding formulae.

Figure C.1: PPINOT graphical Notation Poster

C.3 BPMN Questionnaire

The BPMN questionnaire provide to the experiment participants consists of the set of questions shown in the following.



¿CUÁNTO SABES DE BPMN?

CONCEPTOS DE BPMN

¿Cual es la diferencia entre tarea y subproceso?

¿Cuales son las reglas de conexión cuando usamos flujos de mensajes?

¿Qué representan los pools?

¿Cual es la diferencia entre gateways exclusivos e inclusivos?

¿Que representan los lanes (carriles) generalmente?

SITUACIONES DE MODELADO EN BPMN

Dibuje una tarea con un timeout y el flujo de tareas de salida como consecuencia del timeout

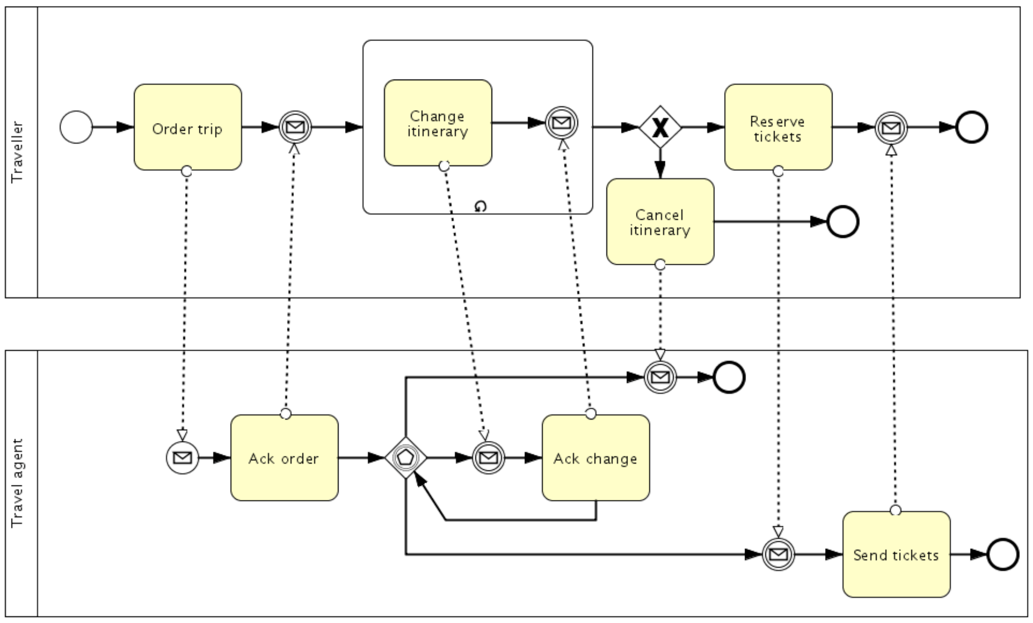
Dibuje dos maneras en que los datos puede salir de una tarea y entrar (ser recibidos) en otra.

Dibuje una espera en un proceso en BPMN

Dibuje la sincronización de dos flujos paralelos

Comprensión de BPMN

Responda las preguntas en función del siguiente diagrama:



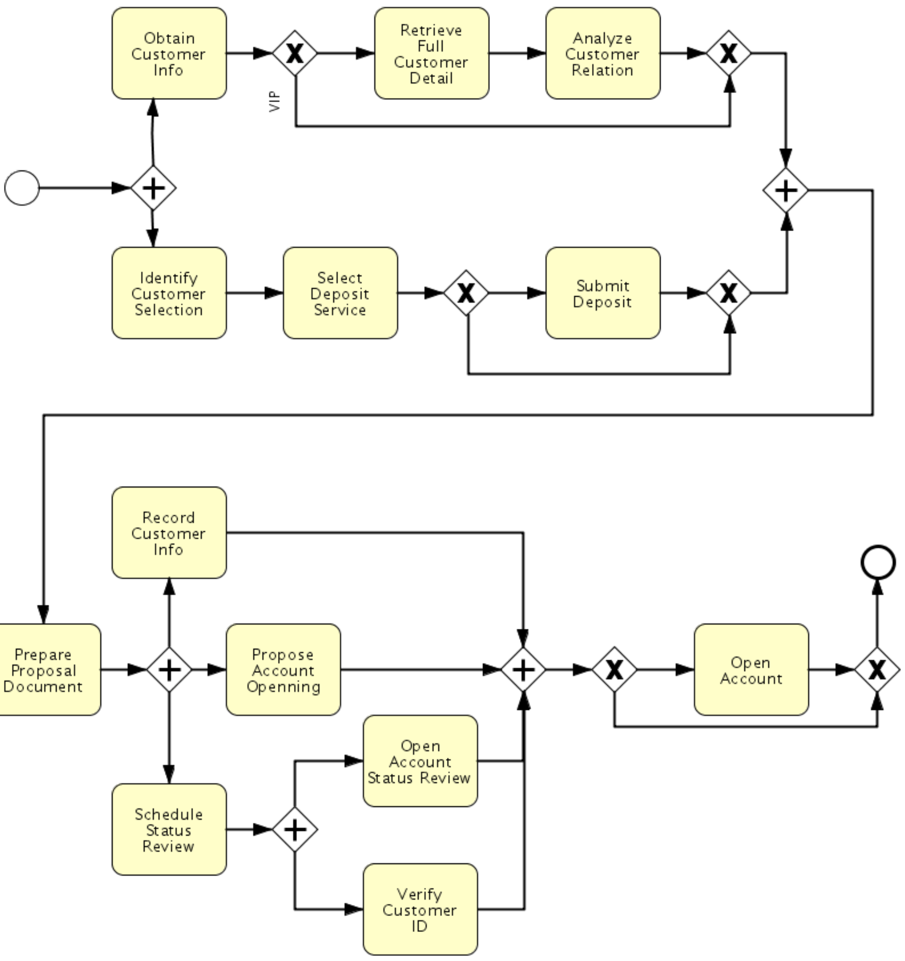
¿Cuántas veces puede ejecutarse la tarea “Ack change”?

¿Cuántas veces puede ejecutarse la tarea “Reserve tickets”?

¿Quién decide cuándo hacer la reserva o la cancelación?

¿Qué significado tiene el gateway basado en eventos situado en el pool inferior?

Responda las preguntas en función del siguiente diagrama:



¿Ocurre siempre “Verify Customer ID” después de “Record Customer Info”?

¿La tarea “Schedule Status Review” se ejecuta en todos los casos?

¿Cuántas tareas se ejecutan como poco? ¿Y como máximo?

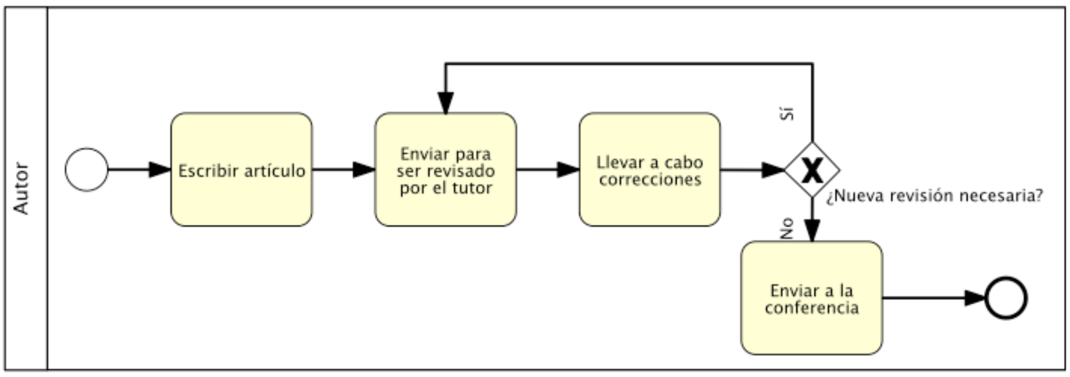
¿Cuántas tareas pueden ocurrir entre “Schedule Status Review” y “Open Account”?

C.4 PPI QUESTIONNAIRE

In this section, we present all the material relative to the PPI questionnaire (three process models, and six questions per process related to PPIs).

Proceso 1: Desarrollo de un artículo de investigación

Observe el siguiente modelo BPMN hasta entender el proceso de negocio que representa.

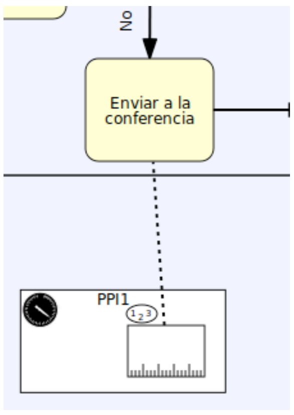

Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
¿Nueva revisión necesaria?
No
Enviar a la conferencia

Dado el proceso de negocio anterior, procedemos a definir una serie de indicadores (PPIs).
Marque la respuesta correcta (sólo una) en cada pregunta:

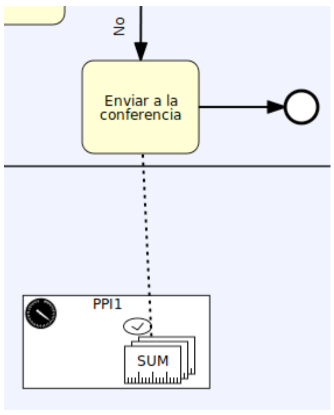
Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

1. **Necesitamos medir el número de artículos enviados a conferencia en el último año (suponemos "analysisPeriod = último año" en todos los casos). ¿Cuál de los siguientes casos representa tal PPI?**

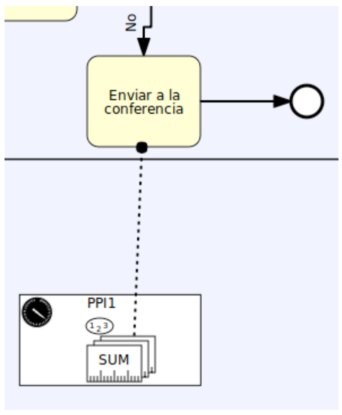
a.



b.



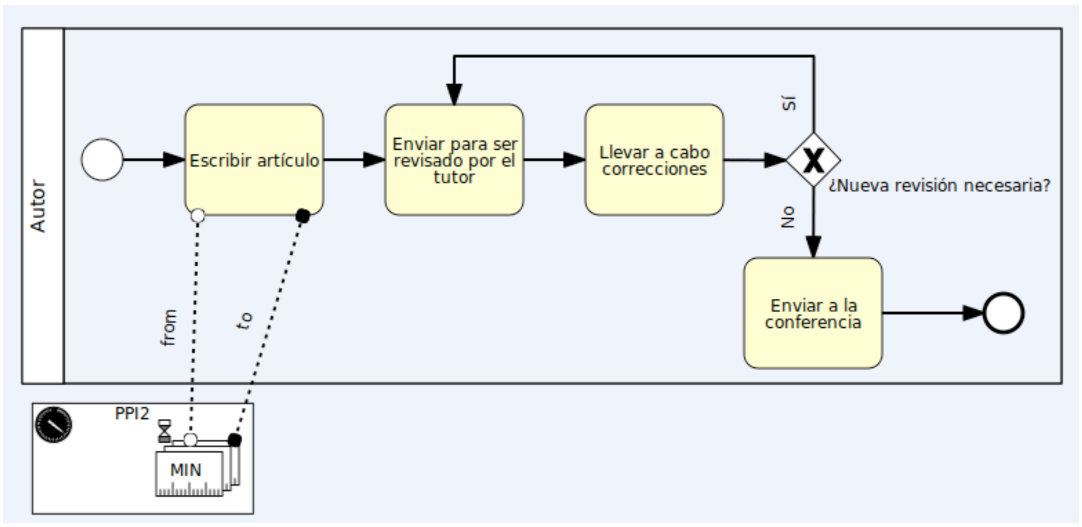
c.



Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

2. **¿Cuál de las tres descripciones de PPI corresponde a la siguiente representación gráfica?**


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
¿Nueva revisión necesaria?
No
Enviar a la conferencia
from
to
PPI2
MIN

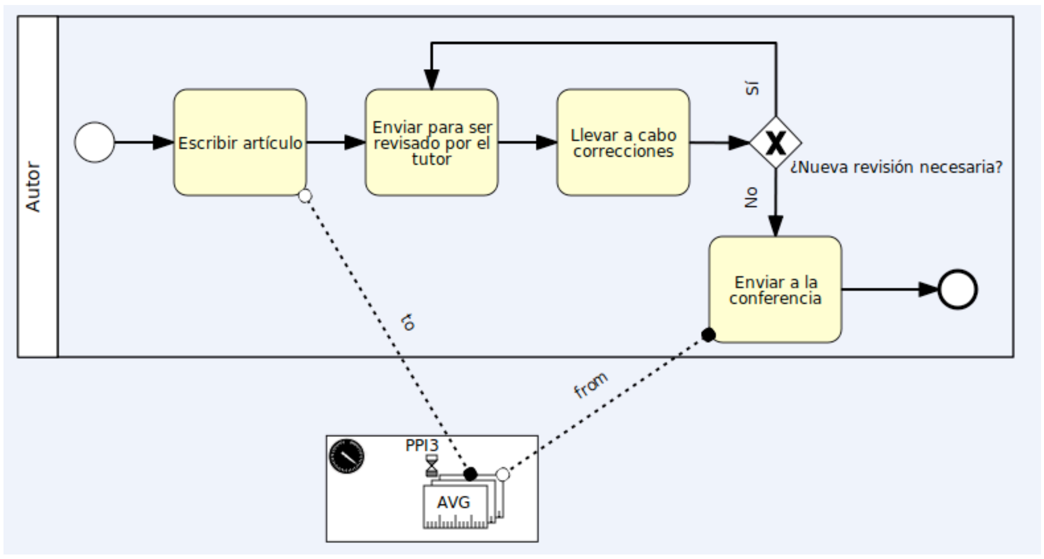
a. Tiempo mínimo de duración del proceso.
b. Número de artículos escritos en el menor tiempo posible.
c. Tiempo mínimo empleado por el autor en escribir un artículo.

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

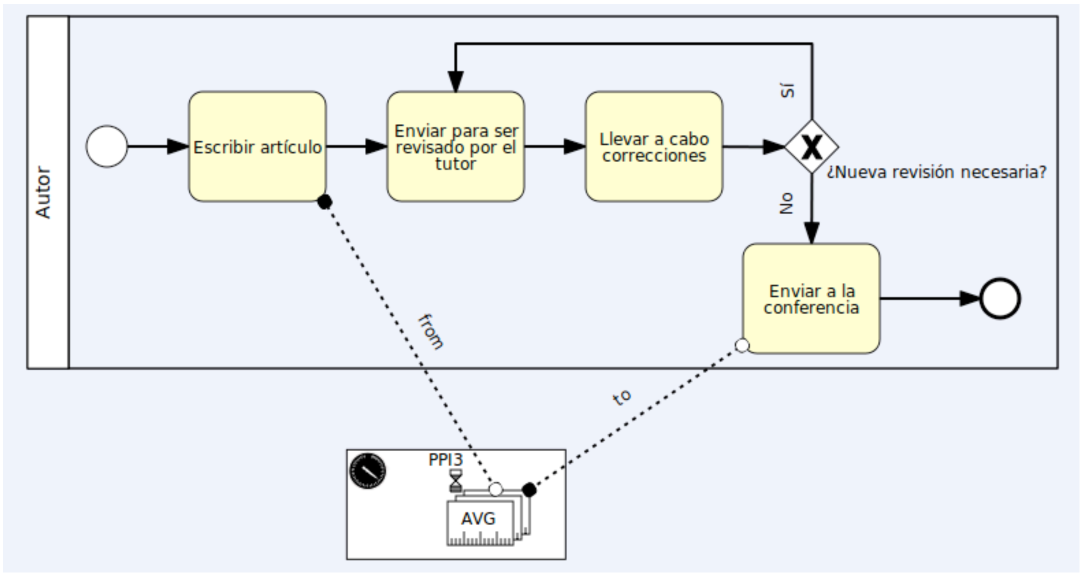
Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

3. **Cuál de las siguientes figuras representa gráficamente el siguiente PPI: “Tiempo medio transcurrido desde que el autor finaliza el proceso de escritura del artículo, hasta que se inicia el envío del mismo”.**

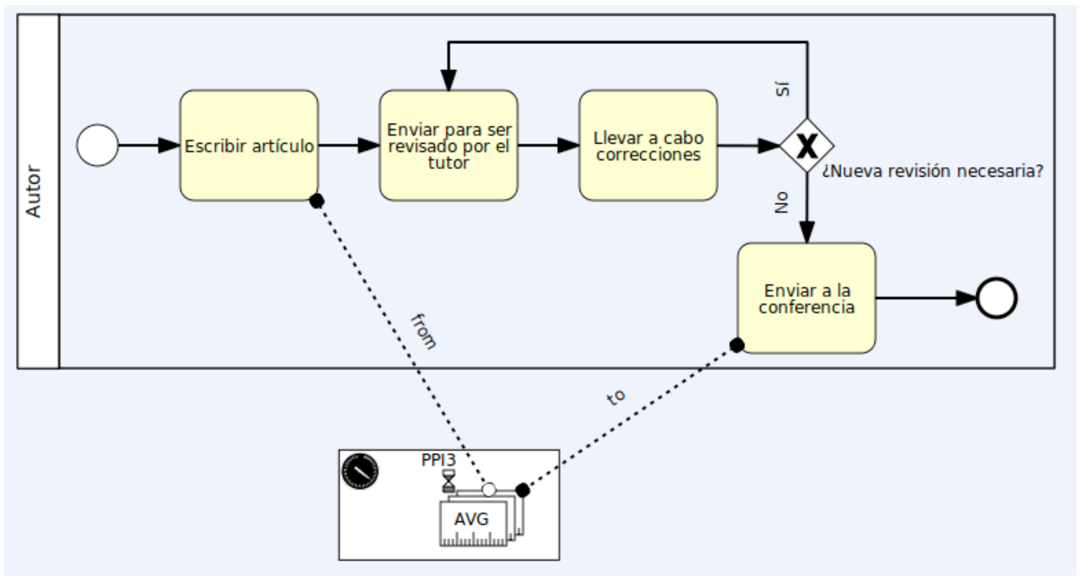
a.


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
to
from
PPI3
AVG

b.


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
from
to
PPI3
AVG

c.

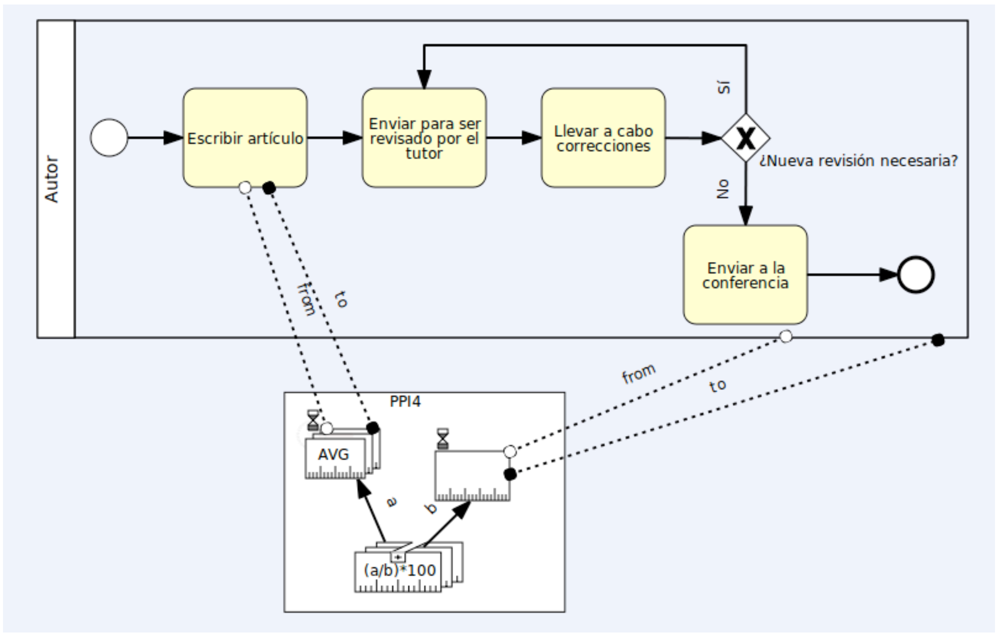

Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
from
to
PPI3
AVG

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

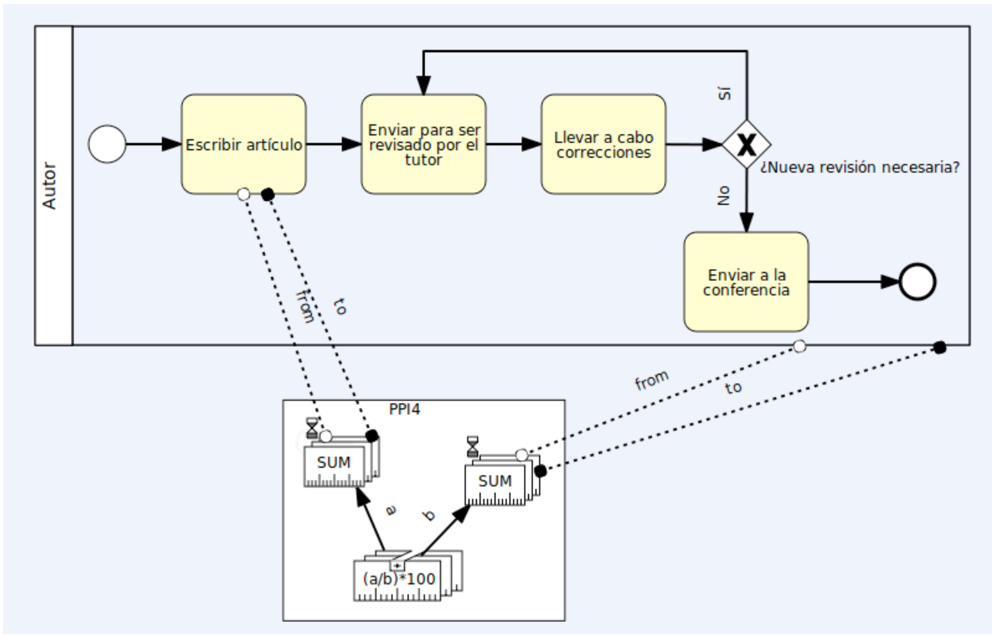
Por favor, escriba la hora de inicio (indicando h/m/s): ____________________________

4. **Señale la figura que representa el PPI: "porcentaje del tiempo de duración del proceso empleado por el autor en escribir el artículo".**

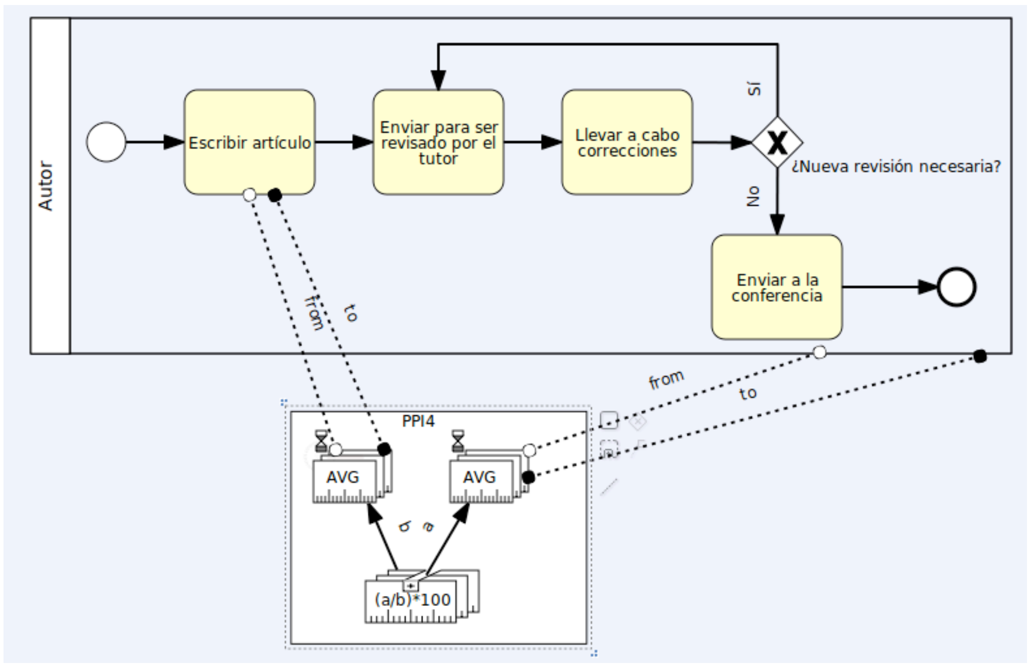
a.


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
from
to
PPI4
AVG
a
b
(a/b)*100
from
to

b.


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
from
to
PPI4
SUM
SUM
a
b
(a/b)*100
from
to

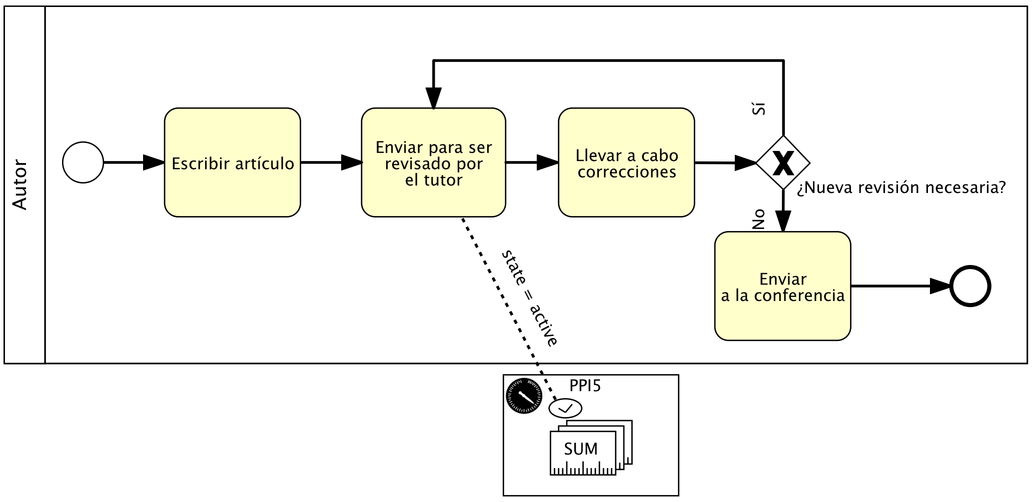
c.


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
from
to
from
to
PPI4
AVG
AVG
b
a
(a/b)*100

Por favor, escriba la hora de fin (indicando h/m/s): ______________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ____________________________

5. **Dada la siguiente representación gráfica de PPI, señale su descripción correcta:**

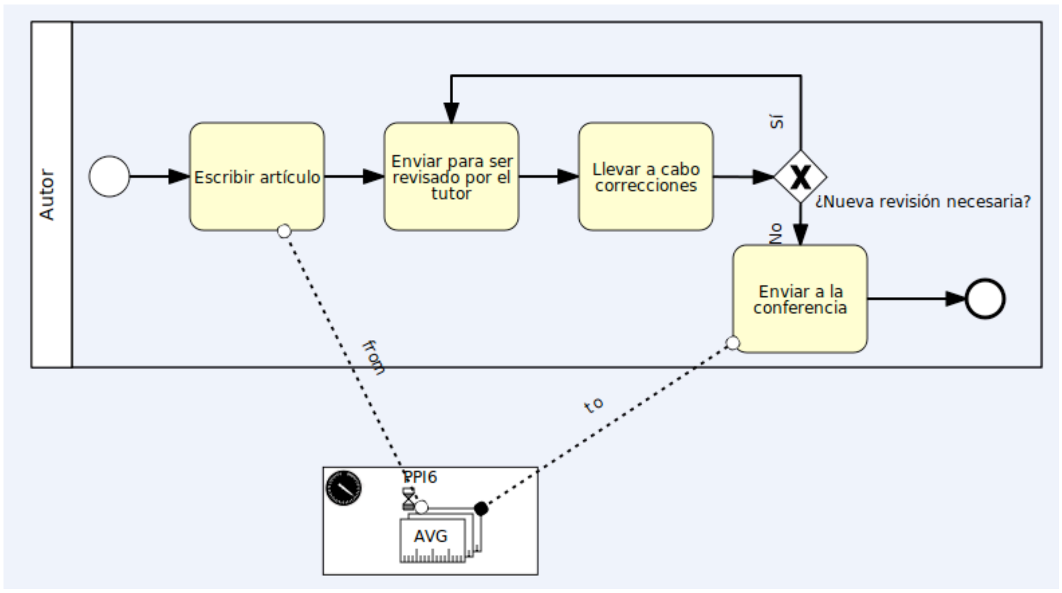

Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
state = active
PPI5
SUM

a. Número de artículos revisados por el tutor.
b. Número de artículos que están siendo enviados para ser revisados.
c. Tiempo que tarda el tutor en revisar el artículo.

Por favor, escriba la hora de fin (indicando h/m/s): ____________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

6. **¿Cómo definiría el PPI representado en la figura que se muestra a continuación? Elija la respuesta correcta.**


Autor
Escribir artículo
Enviar para ser revisado por el tutor
Llevar a cabo correcciones
Sí
No
¿Nueva revisión necesaria?
Enviar a la conferencia
from
to
PPI6
AVG

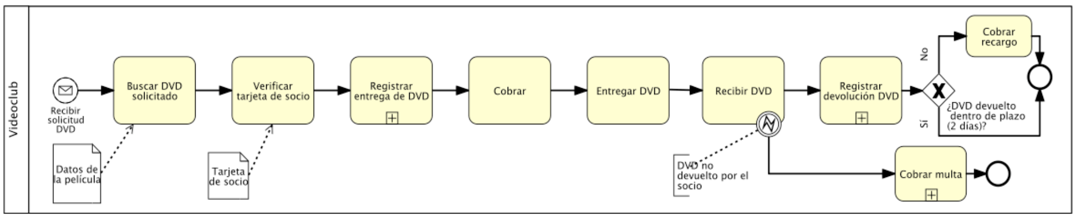
a. Tiempo medio transcurrido desde que el autor comienza a escribir el artículo hasta que termina de realizar todas las correcciones.
b. Tiempo medio transcurrido desde que el autor comienza a escribir el artículo hasta que ha realizado el envío a la conferencia.
c. Tiempo medio de duración del proceso "desarrollo de un artículo" completo.

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

7. **Según su propio criterio, ¿cómo calificaría la dificultad de este modelo de proceso?**
- ○ Muy fácil
- ○ Fácil
- ○ Normal
- ○ Bastante complejo
- ○ Muy complejo

Proceso 2: Alquiler de película en un videoclub

Observe el siguiente modelo BPMN hasta entender el proceso de negocio que representa

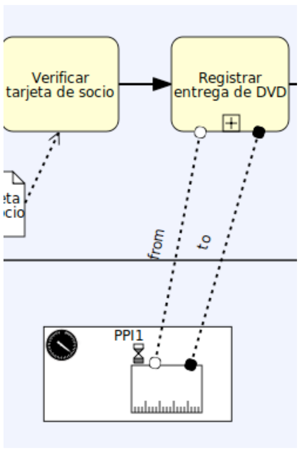


Dado el proceso de negocio anterior, procedemos a definir una serie de indicadores (PPIs).
Marque la respuesta correcta (sólo una) en cada pregunta:

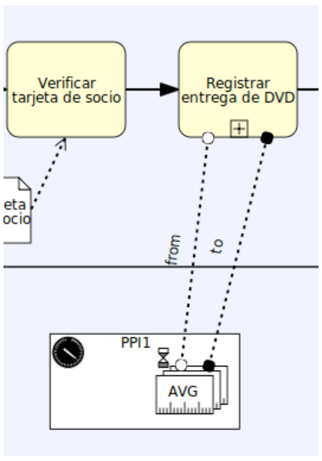
Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

1. **Señale la figura que mejor representa el PPI: "Tiempo que tardó el videoclub en registrar la entrega del DVD la última vez que fui".**

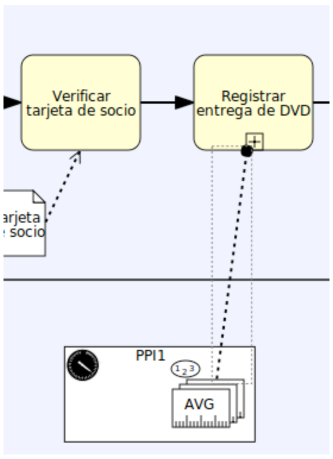
a.



b.



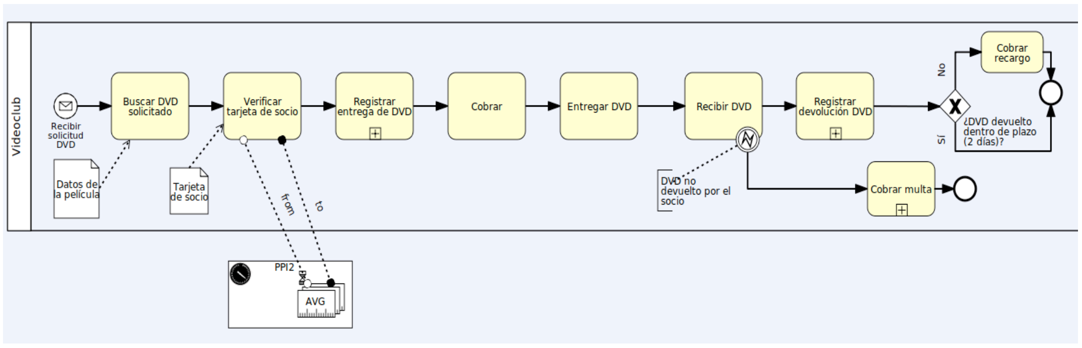
c.



Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

2. **Dada la siguiente representación gráfica de PPI, señale su descripción correcta:**



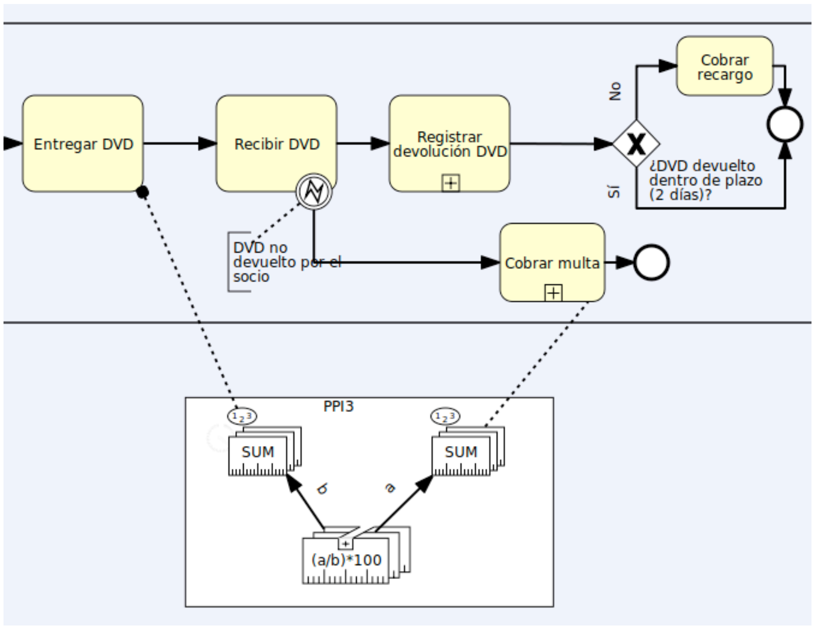
a. Tiempo total empleado por el videoclub en verificar la tarjeta de socio.
b. Tiempo medio empleado por el videoclub en verificar la tarjeta de socio.
c. Tiempo medio empleado por el videoclub en verificar la tarjeta de socio y registrar la entrega del DVD.

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

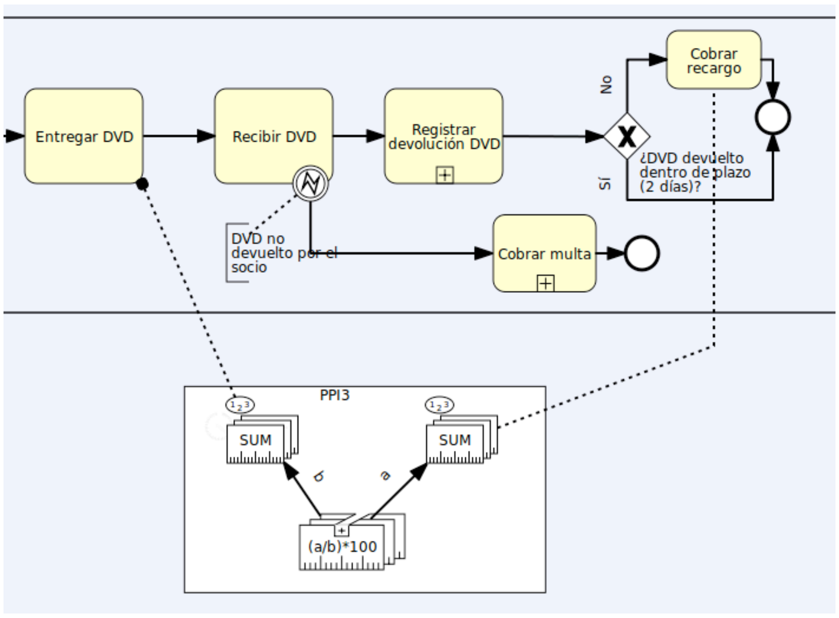
Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

3. ¿Cuál de las siguientes representaciones gráficas se corresponde con el PPI "Porcentaje de DVDs devueltos fuera de plazo con respecto al total entregados a socios"?

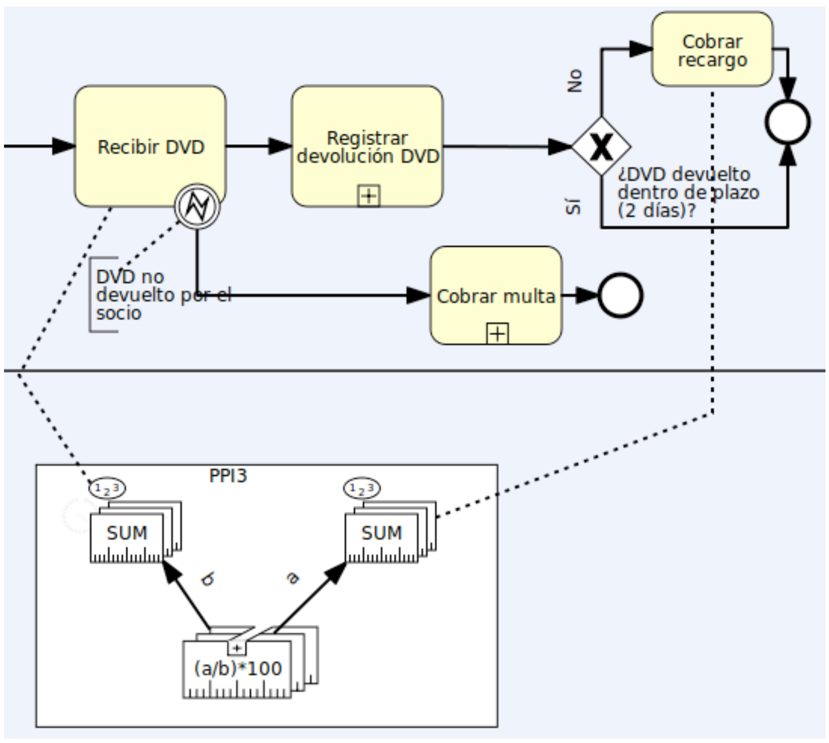
a.



b.



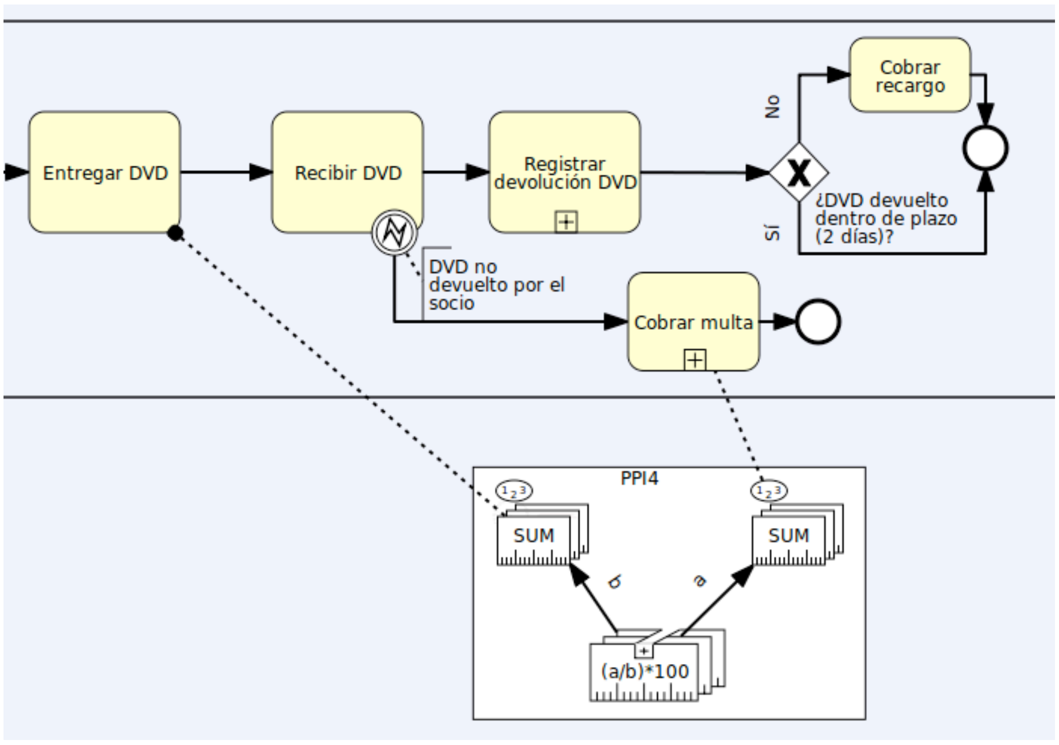
c.



Por favor, escriba la hora de fin (indicando h/m/s): ____________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

4. ¿Cómo definiría el PPI representado en la figura que se muestra a continuación? Elija la respuesta correcta.



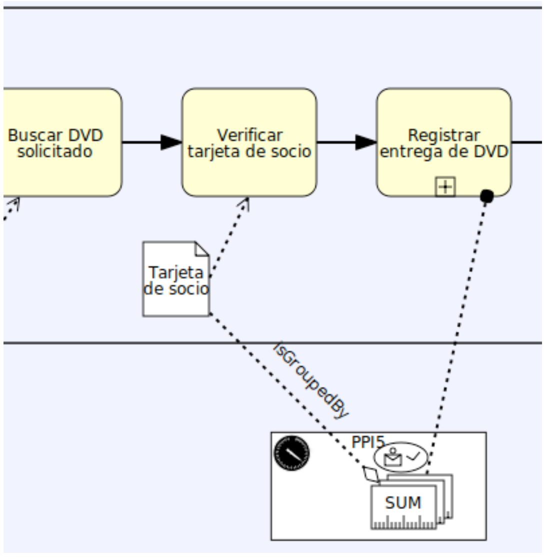
a. Porcentaje de DVDs devueltos fuera de plazo con respecto al total entregados a socios.
b. Porcentaje de DVDs recibidos con respecto al total entregados a socios.
c. Porcentaje de DVDs no devueltos con respecto al total entregados a socios.

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

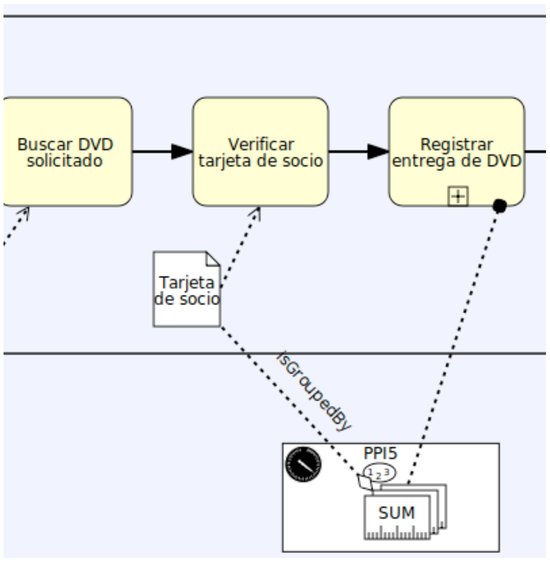
Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

5. **¿Cuál de los siguientes casos representa el PPI “Número de DVDs alquilados por socio”?**

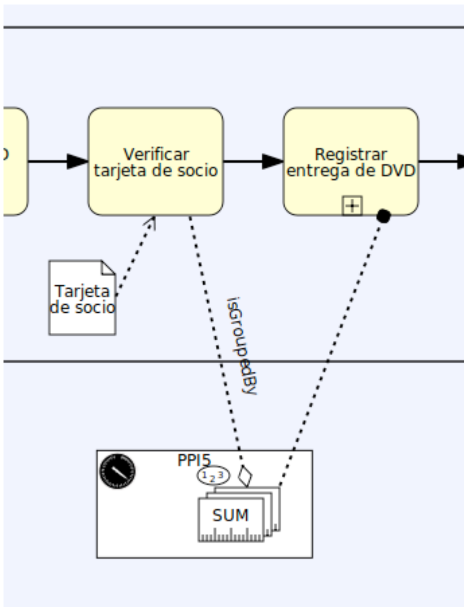
a.



b.



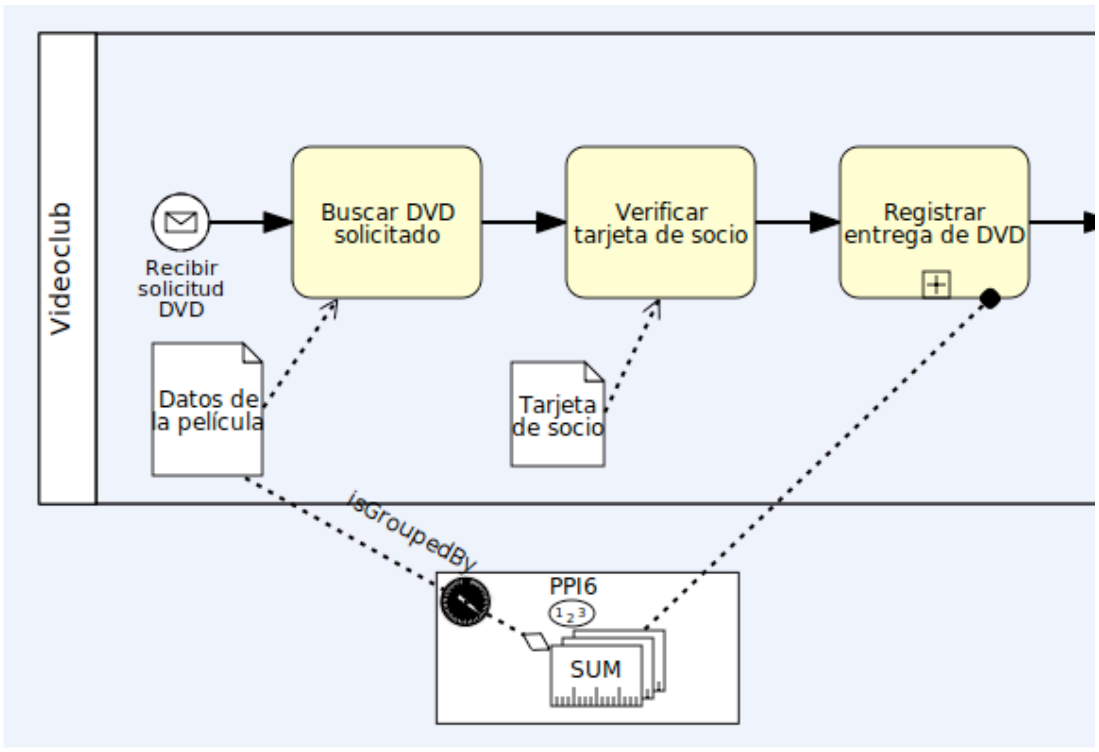
c.



Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

6. ¿Cuál de las tres descripciones de PPI corresponde a la siguiente representación gráfica?



a. Número de películas para mayores de 18 años solicitadas.
b. Número de películas para mayores de 18 años devueltas.
c. Número de películas entregadas, agrupadas por edad recomendada.

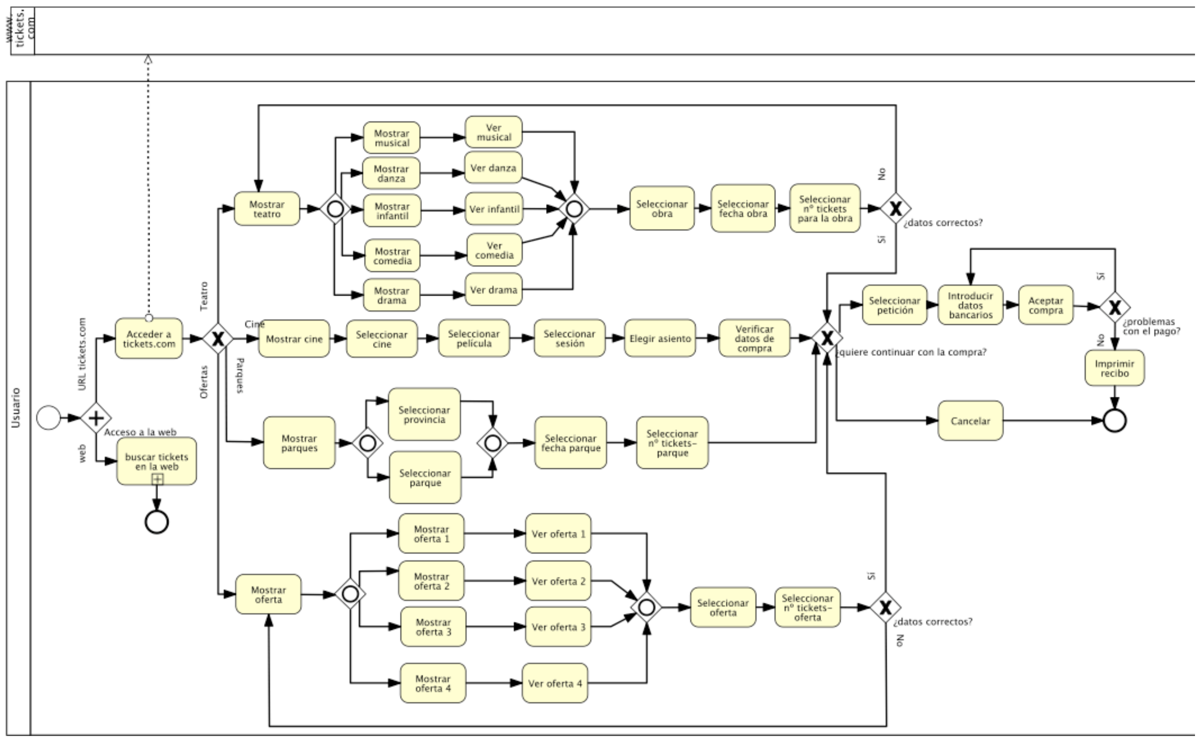
Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

---

7. **Según su propio criterio, ¿cómo calificaría la dificultad de este modelo de proceso?**
- ○ Muy fácil
- ○ Fácil
- ○ Normal
- ○ Bastante complejo
- ○ Muy complejo

Proceso 3: Compra de tickets online

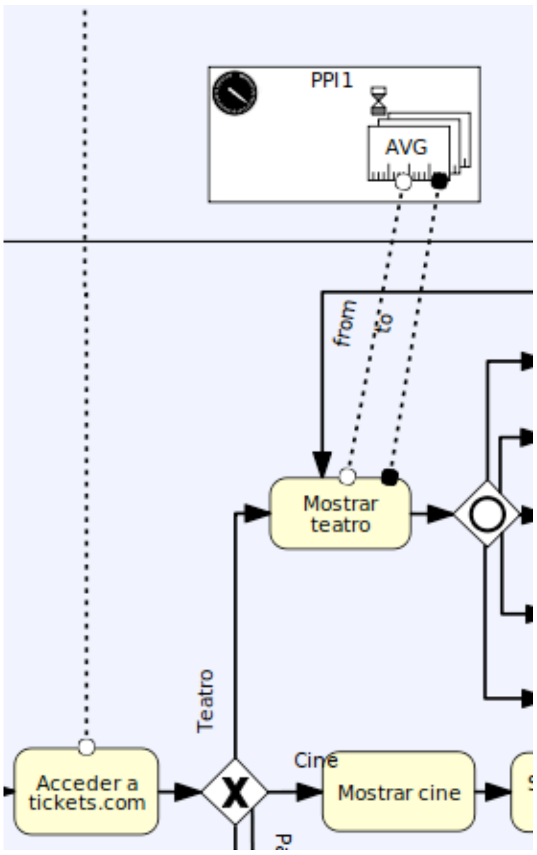
Observe el siguiente modelo BPMN hasta entender el proceso de negocio que representa



Dado el proceso de negocio anterior, procedemos a definir una serie de indicadores (PPIs).
Marque la respuesta correcta (sólo una) en cada pregunta:

Por favor, escriba la hora de inicio (indicando h/m/s): ____________________________

1. ¿Qué estamos midiendo en el proceso de acuerdo con la siguiente figura? Elija la respuesta correcta.



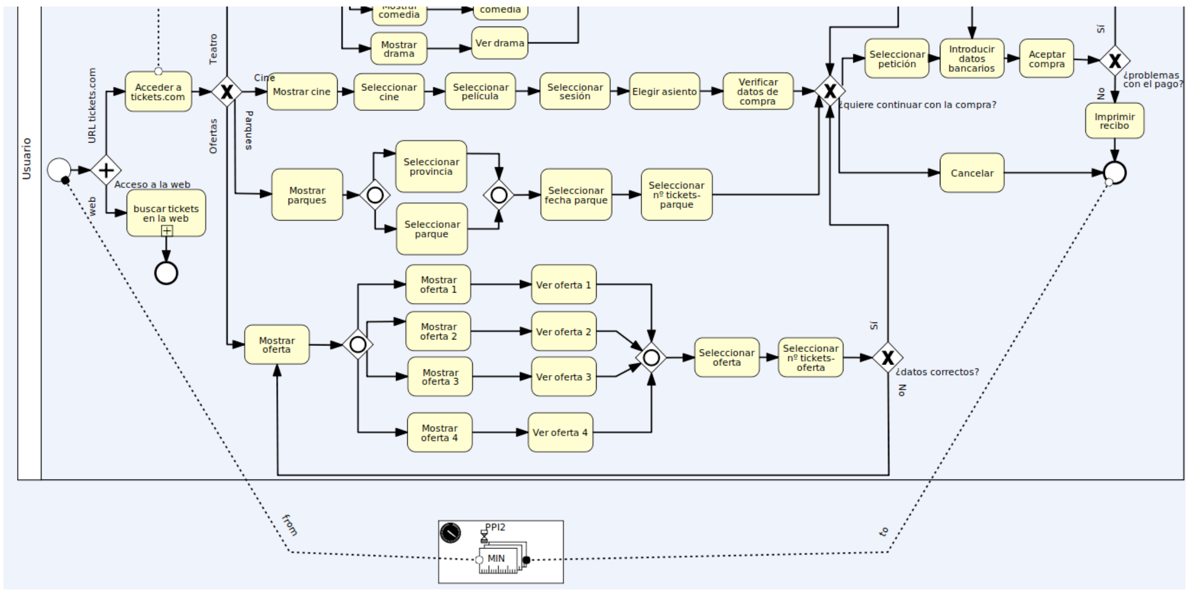
a. Tiempo medio que se tarda en mostrar los teatros.
b. Número de veces aproximado que muestra los teatros.
c. Tiempo medio que tarda en mostrar los cines.

Por favor, escriba la hora de fin (indicando h/m/s): ____________________________

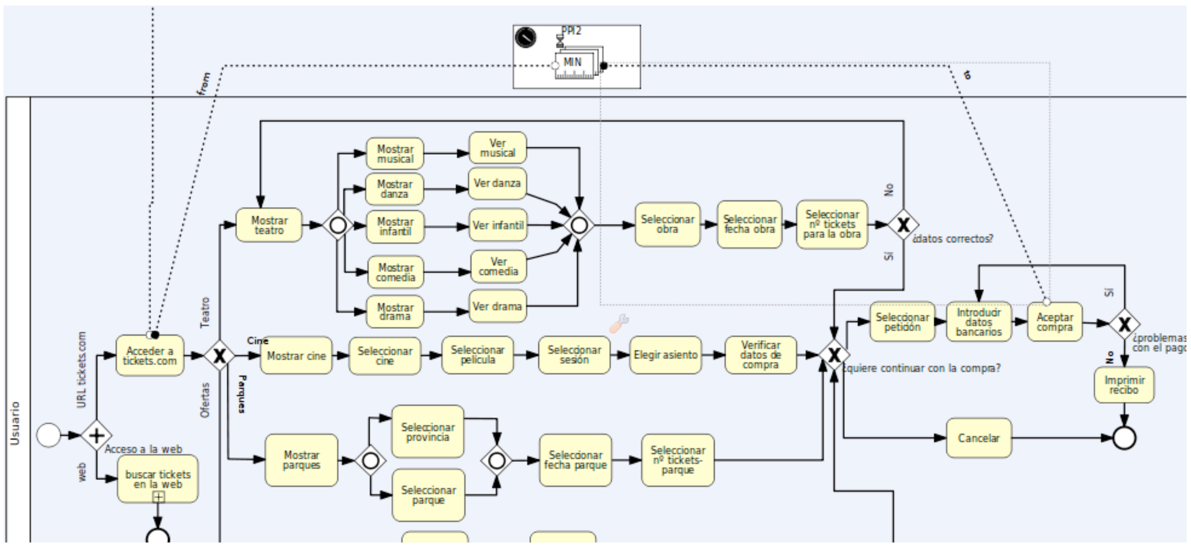
Por favor, escriba la hora de inicio (indicando h/m/s): ________________________

2. ¿Cuál de las siguientes figuras representa al PPI "Tiempo mínimo transcurrido desde que el usuario accede al sistema hasta que finaliza la compra"?

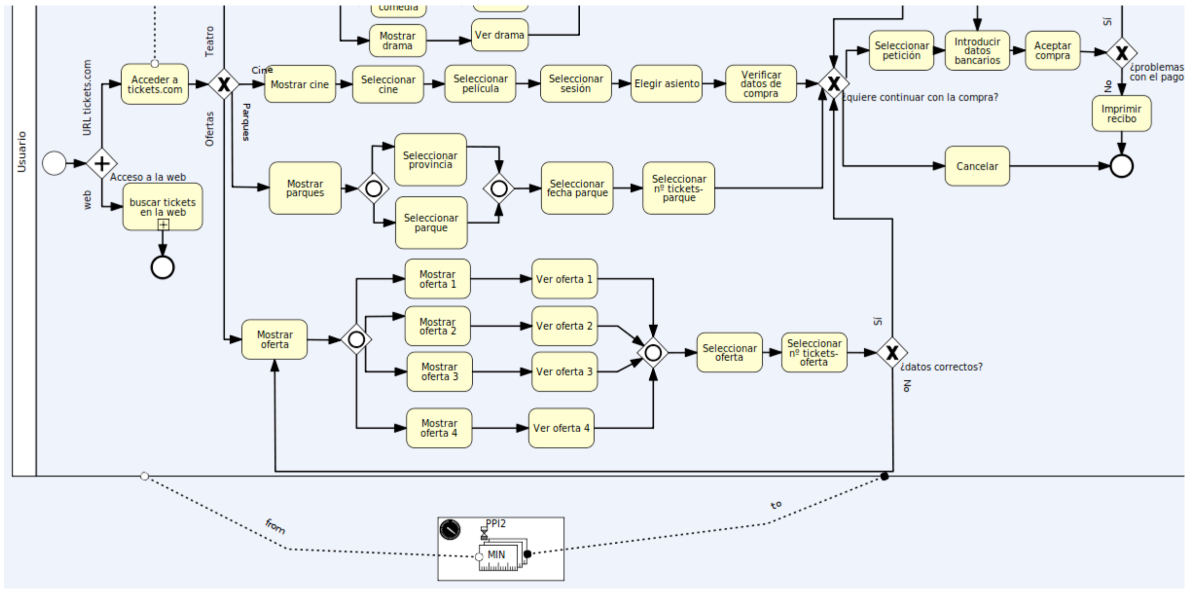
a.



b.



c.

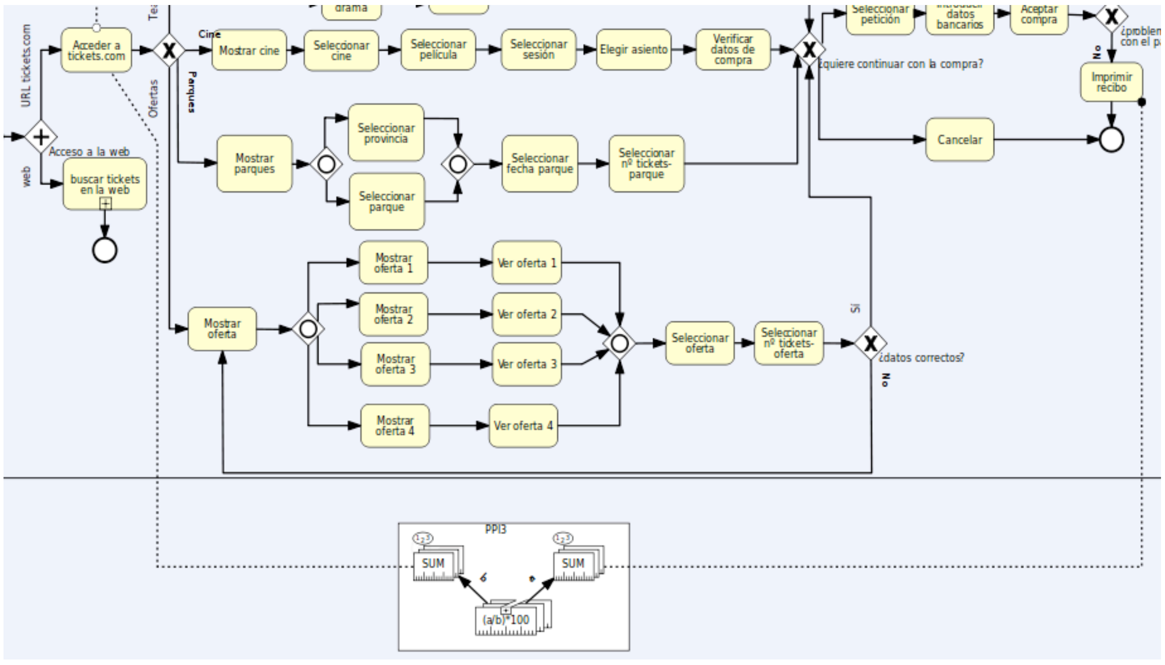


Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

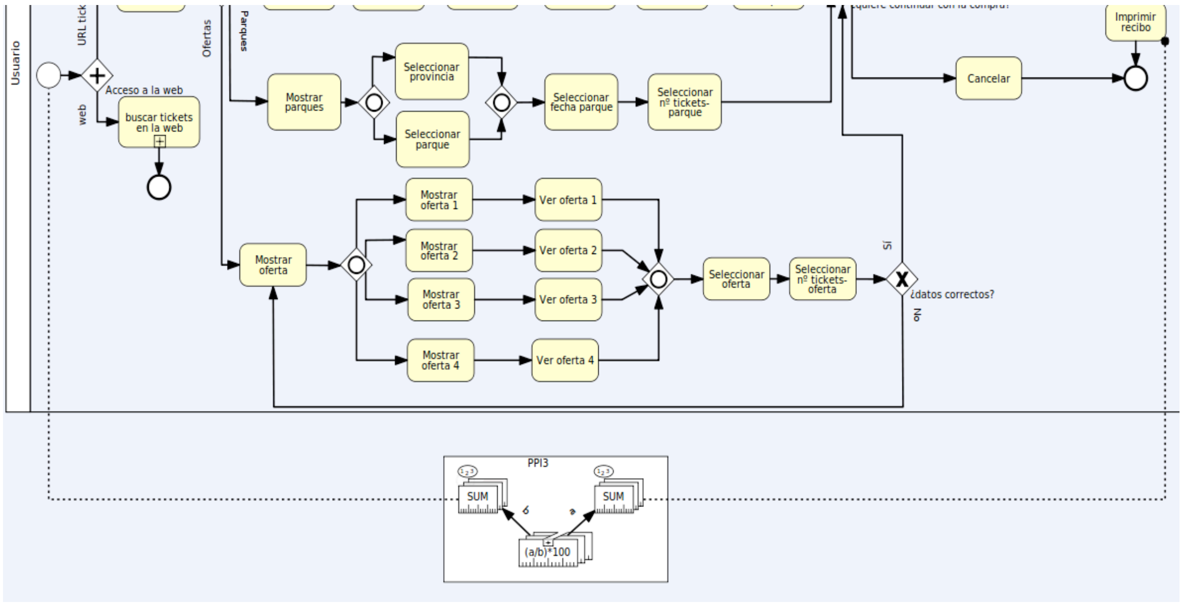
Por favor, escriba la hora de inicio (indicando h/m/s): ______________________

3. Señale la figura que mejor representa el PPI: “Porcentaje de compras realizadas con respecto al total de accesos a tickets.com”.

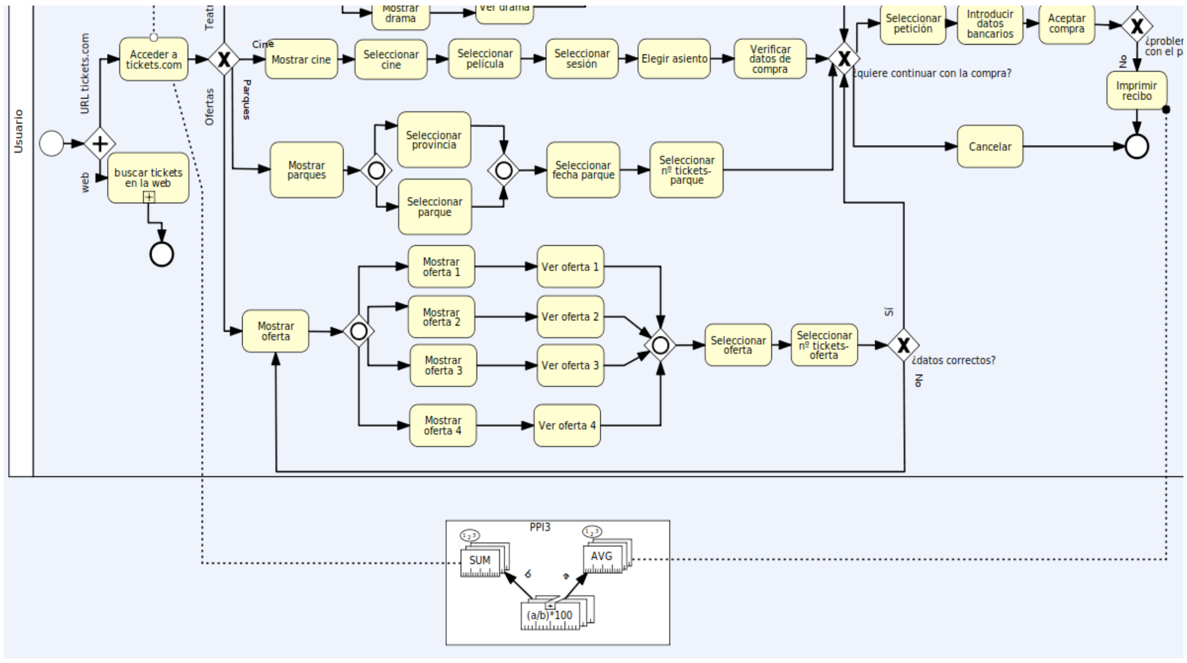
a.



b.



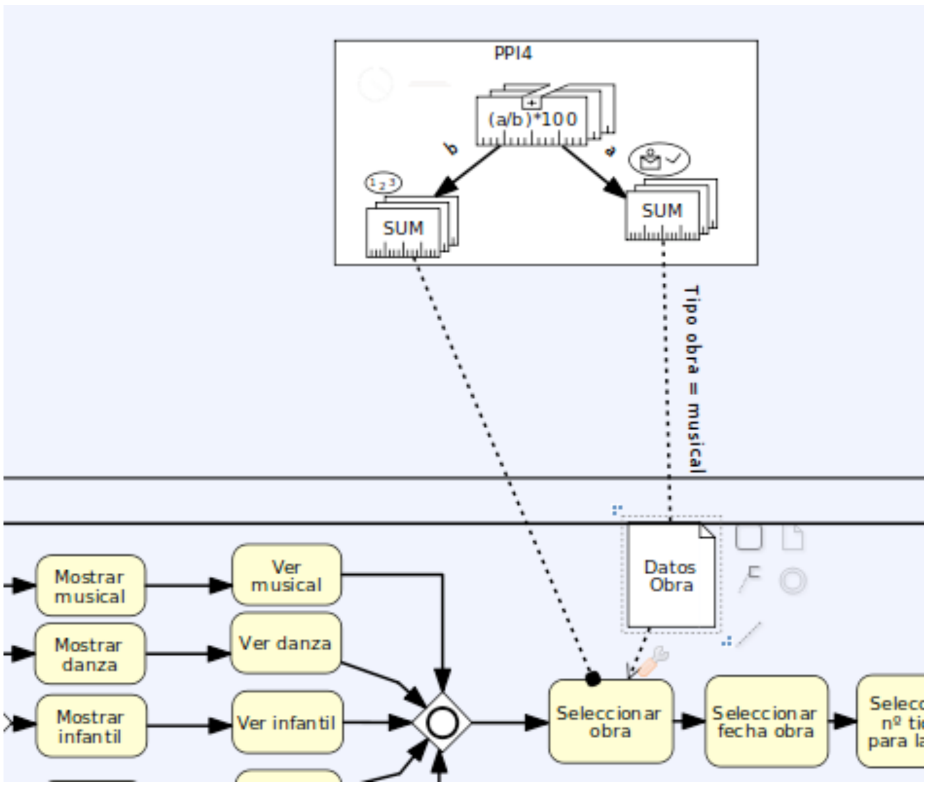
c.



Por favor, escriba la hora de fin (indicando h/m/s): ________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

4. ¿Cuál de las tres descripciones de PPI corresponde a la siguiente representación gráfica?

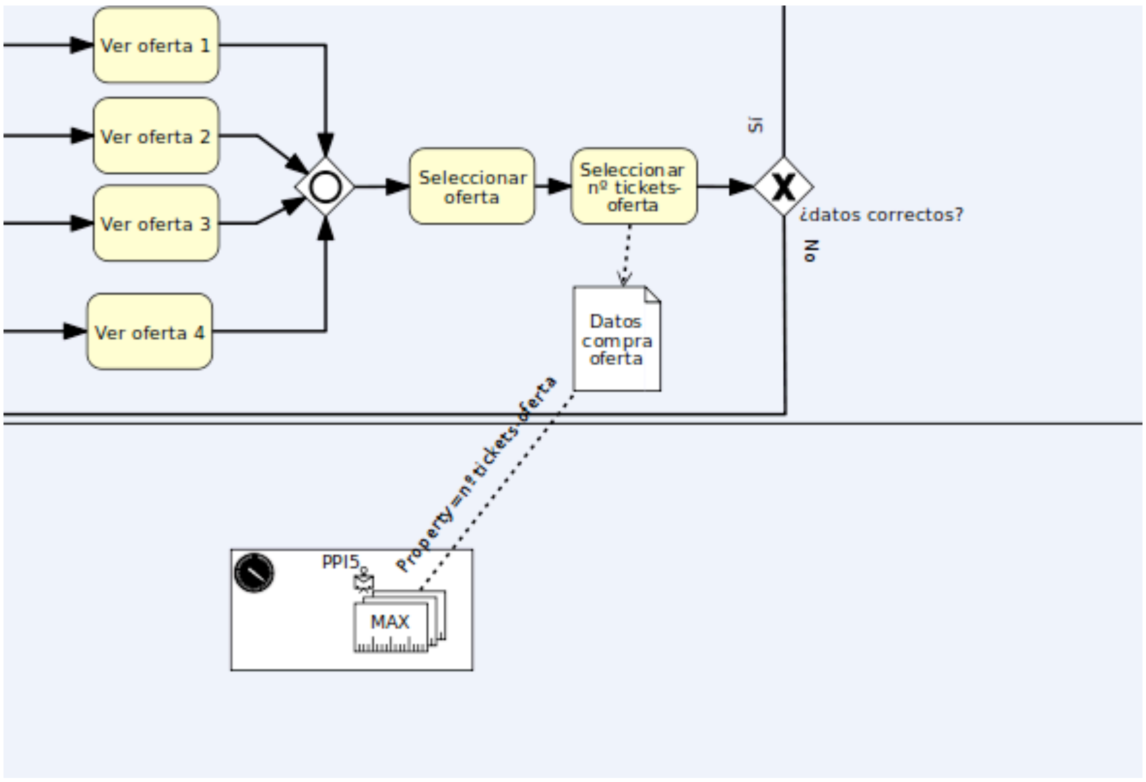


a. Porcentaje de musicales con respecto al número de obras de teatro vistas.
b. Porcentaje de musicales con respecto al número de obras de teatro seleccionadas.
c. Porcentaje de musicales con respecto al número de obras de teatro mostradas.

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

Por favor, escriba la hora de inicio (indicando h/m/s): ___________________________

5. ¿Cuál de las tres descripciones de PPI corresponde a la siguiente representación gráfica?



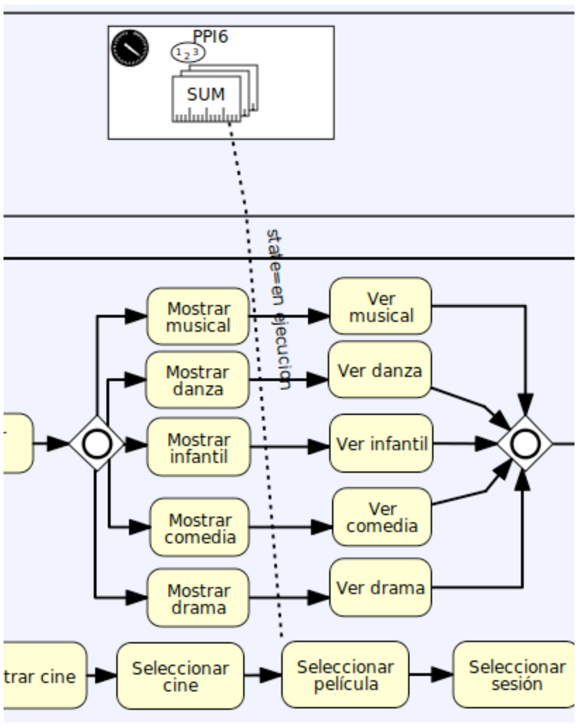
a. Número máximo de ofertas mostradas por tickets.com.
b. Número máximo de tickets de la oferta seleccionados en un acceso.
c. Número máximo de tickets de oferta que se pueden comprar.

Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

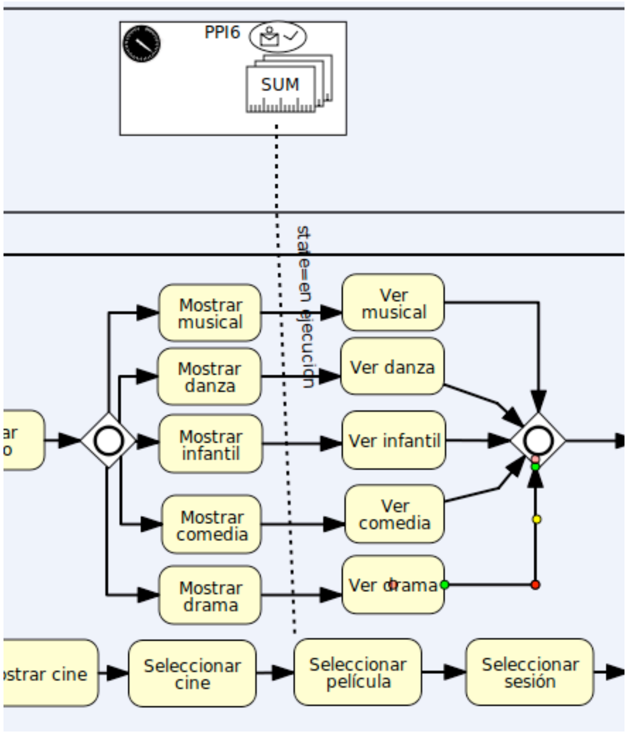
Por favor, escriba la hora de inicio (indicando h/m/s): ____________________________

6. **¿Cuál de los siguientes casos representa el PPI "Número de usuarios que están seleccionando una película de cine en este momento"?**

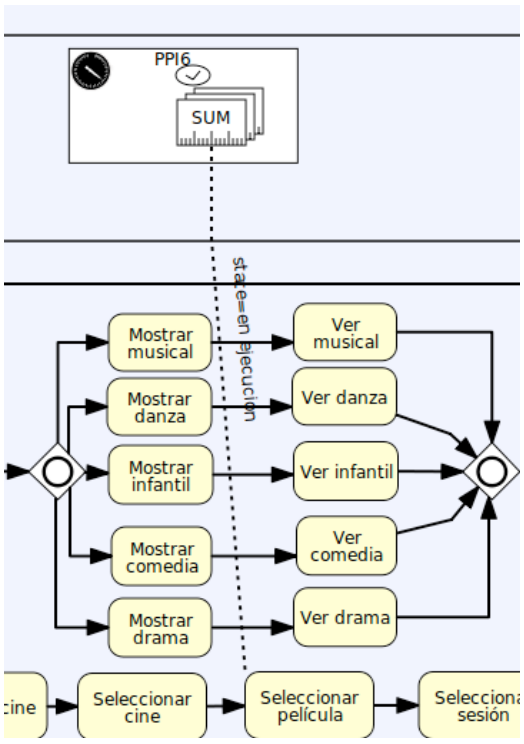
a.



b.



c.



Por favor, escriba la hora de fin (indicando h/m/s): ___________________________

7. **Según su propio criterio, ¿cómo calificaría la dificultad de este modelo de proceso?**

- ○ Muy fácil
- ○ Fácil
- ○ Normal
- ○ Bastante complejo
- ○ Muy complejo

Acronyms

BAM	Business Activity Monitoring.
BP	Business Process.
BPD	Business Process Diagram.
BPEL	Business Process Execution Language.
BPM	Business Process Management.
BPMN	Business Process Model and Notation.
BPMS	Business Process Management System.
CPM	Corporate Performance Management.
DL	Description Logics.
EPC	Event-driven Process Chain.
GUI	Graphical User Interface.
IT	Information Technology.
KB	Knowledge Base.
KPI	Key Performance Indicator.
L-PATTERN	Linguistic Pattern.
MDA	Model Driven Architecture.

PPI	Process Performance Indicator.
PPIM	PPI Management.
PPM	Process Performance Management.
SLA	Service Level Agreement.
SOA	Servie Oriented Architecture.
UML	Unified Modeling Language.

Bibliography

[1] G. D. Alexander Grosskopf and M. Weske. *The Process. Busisness Process Modeling using BPMN*. Megan-kiffer Press, Mar 2009.

[2] F. Baader and W. Nutt. Basic description logics. In Baader et al. [3], pages 43–95. ISBN 0-521-78176-0.

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003. Cambridge University Press. ISBN 0-521-78176-0.

[4] D. Barone, F. Stella, and C. Batini. Dependency discovery in data quality. In Pernici [72], pages 53–67. ISBN 978-3-642-13093-9.

[5] D. Benavides, S. Segura, and A. R. Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, 2010.

[6] M. Bhatt, W. Rahayu, S. P. Soni, and Carlo. Ontology driven semantic profiling and retrieval in medical information systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):317 – 331, 2009. ISSN 1570-8268. doi: 10.1016/j.websem.2009.05.004. URL http://www.sciencedirect.com/science/article/pii/S1570826809000201. <ce:title>Semantic Web challenge 2008</ce:title>.

[7] P. Brewer and T. Speh. Using the balance scorecard to measure supply chain performance. *Journal of Business Logistics*, 21:75–93, 2000.

[8] J. A. Brimson and J. Antos. *Activity-based management: for service industries, government entities, and nonprofit organizations*. Wiley Self-Teaching Guides Series. Wiley, 1994. ISBN 9780471013518. URL http://books.google.es/books?id=5aJCjodQZZEC.

[9] F. Buytendijk, B. Wood, and L. Geishecker. Mapping the road to corporate performance management. Technical Report 30, Gartner, 2004.

[10] C. Cabanillas, M. Resinas, and A. R. Cortés. Ral: A high-level user-oriented resource assignment language for business processes. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops (1)*, volume 99 of *Lecture Notes in Business Information Processing*, pages 50–61. Springer, 2011. ISBN 978-3-642-28107-5.

[11] C. Cabanillas, M. Resinas, and A. R. Cortés. Defining and analysing resource assignments in business processes with ral. In G. Kappel, Z. Maamar, and H. R. M. Nezhad, editors, *ICSOC*, volume 7084 of *Lecture Notes in Computer Science*, pages 477–486. Springer, 2011. ISBN 978-3-642-25534-2.

[12] M. Castellanos, F. Casati, M.-C. Shan, and U. Dayal. ibom: a platform for intelligent business operation management. In *Proceedings. 21st International Conference on Data Engineering, 2005.*, pages 1084– 1095. Hewlett-Packard Laboratories, 2005. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1410218&isnumber=30564`.

[13] F. Chan. Performance measurement in a supply chain. *International Journal of Advanced Manufacturing Technology*, 21:534–548, 2003.

[14] G. Chase, A. Rosenberg, R. Omar, J. Taylor, and M. Rosing. *Applying Real-World BPM in an SAP Environment*. SAP Press. Galileo Press, Incorporated, 2011. ISBN 9781592293438. URL `http://books.google.es/books?id=4dqERAAACAAJ`.

[15] D. Chen, B. Vallespir, and G. Doumeingts. Grai integrated methodology and its mapping onto generic enterprise reference architecture and methology. *Computers in Industry*, 33: 387–394, 1997.

[16] C. Costello and O. Molloy. Building a process performance model for business activity monitoring. In W. Wojtkowski, G. Wojtkowski, M. Lang, K. Conboy, and C. Barry, editors, *Information Systems Development*, pages 237–248. Springer US, 2009. ISBN 978-0-387-68772-8.

[17] T. H. Davenport. *Process innovation: reengineering work through information technology*. Harvard Business School Press, Boston, MA, USA, 1993. ISBN 0-87584-366-2.

[18] R. Davis and E. Brab´ander. Aris design platform. Springer, 2007.

[19] G. Decker. *Design and Analysis of Process Choreographies*. PhD thesis, University of Potsdam, 2009.

[20] G. Decker, H. Overdick, and M. Weske. Oryx - an open modeling platform for the bpm community. In *BPM*, pages 382–385, 2008.

[21] A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés. Challenges to support a ppi management lifecycle. In *III Taller de Procesos de Negocio e Ingeniería de Servicios (PNIS 2010). Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, volume 4, pages 8–13, 2010.

[22] A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés. Defining process performance indicators: An ontological approach. In *Proceedings of the 18th International Conference on Cooperative Information Systems (CoopIS). OTM 2010, Part I*, pages 555–572, October, 2010.

[23] D. M. T. F. (DMTF). Common information model (cim) metrics model, June 2003. URL `http://www.dmtf.org/standards/documents/CIM/DSP0141.pdf`.

[24] G. T. Doran. There's a s.m.a.r.t. way to write management's goals and objectives. *Management Review*, 70(11):35–36, 1981.

[25] G. Doumeingts, B. Vallespir, and D. Chen. *Handbook on Architectures of Information Systems*, chapter Decisional Modelling Using the GRAI Grid, pages 313–338. Springer-Verlag, 1998.

[26] H. Dresner. Business activity monitoring: Bam architecture, 2003.

[27] A. Durán, B. Bernárdez, A. Ruiz-Cortés, and M. Toro. A Requirements Elicitation Approach based in Templates and Patterns. In *Proc. Workshop de Engenharia de Requisitos (WERÕ99)*, Buenos Aires, Argentina, 1999.

[28] A. Durán, A. Ruiz-Cortés, R. Corchuelo, and M. Toro. Supporting Requirements Verification using XSLT. In *Proc. IEEE Joint International Conference on Requirements Engineering*, pages 165–172, 2002.

[29] J. Eder and W. Liebhart. Workflow recovery. In *CoopIS*, pages 124–134, 1996.

[30] EFQM. EFQM excellence model 2010, 2010.

[31] U. R. T. Force. Omg unified modeling language specification, version 1.4 (final draft), Februrary 2001.

[32] F. Franceschini, M. Galetto, and D. Maisano. *Management by Measurement: Designing Key Indicators and Performance Measurement Systems*. Springer Verlag,

2007. ISBN 9783540732129. URL http://books.google.es/books?id=hkiyten_SIoC.

[33] J.-P. Friedenstab, C. Janiesch, M. Matzner, and O. Muller. Extending bpmn for business activity monitoring. In *Hawaii International Conference on System Sciences*, volume 0, pages 4158–4167. IEEE Computer Society, 2012. ISBN 978-0-7695-4525-7. doi: http://doi.ieeecomputersociety.org/10.1109/HICSS.2012.276.

[34] M. N. Frolick and T. Ariyachandra. Business performance management: One truth. *IS Management*, 23(1):41–48, 2006.

[35] F. García, M. F. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, and M. Genero. Towards a consistent terminology for software measurement. *Information & Software Technology*, 48(8):631–644, 2006.

[36] J. M. García, D. Ruiz, and A. Ruiz-Cortés. A model of user preferences for semantic services discovery and ranking. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *ESWC (2)*, volume 6089 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2010. ISBN 978-3-642-13488-3.

[37] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58:89–123, 2002.

[38] M. Golfarelli, S. Rizzi, and I. Cella. Beyond data warehousing: what's next in business intelligence? In *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, DOLAP '04, pages 1–6. ACM, 2004. ISBN 1-58113-977-2. doi: 10.1145/1031763.1031765. URL http://doi.acm.org/10.1145/1031763.1031765.

[39] O. González, R. Casallas, and D. Deridder. MMC-BPM: A Domain-Specific Language for Business Processes Analysis. *Business Information Systems*, pages 157–168, 2009. doi: 10.1007/978-3-642-01190-0_14. URL http://dx.doi.org/10.1007/978-3-642-01190-0_14.

[40] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004. doi: 10.1016/j.compind.2003.10.007.

[41] M. Hammer and J. Champy. *Reengineering the corporation: a manifesto for business revolution*. HarperBusiness, New York, 1st ed. edition, 1993. ISBN 186448392 0887306403 088730687 186448392 1863737065 0887306403 1857880293 088730687.

[42] H. Heß. From corporate strategy to process performance - what comes after business intelligence? In *Corporate Performance Management: Aris in Practice*, pages 7–29. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-30787-7. URL http://dx.doi.org/10.1007/3-540-30787-7_2.

[43] ISACA. COBIT v4.1, 2009.

[44] R. S. Kaplan and D. P. Norton. The balanced scorecard: Measures that drive performance. *Harvard Business Review*, January-February 1992:71–79, 1992.

[45] R. S. Kaplan and D. P. Norton. The balanced scorecard Ð measures that drive performance. *Harvard Business Review*, 70(1):71–79, 1992. URL http://www.ncbi.nlm.nih.gov/pubmed/10119714.

[46] R. S. Kaplan and D. P. Norton. *The Balanced Scorecard: Translating Strategy Into Action*. Harvard Business School Press. Harvard Business School Press, 1996. ISBN 9780875846514. URL http://books.google.es/books?id=mRHC5kHXczEC.

[47] B. Korherr and B. List. Extending the epc and the bpmn with business process goals and performance measures. In J. Cardoso, J. Cordeiro, and J. Filipe, editors, *ICEIS (3)*, pages 287–294, 2007. ISBN 978-972-8865-90-0.

[48] E. Krauth, H. Moonen, V. Popova, and M. C. Schut. Performance measurement and control in logistics service providing. In *ICEIS (2)*, pages 239–247, 2005.

[49] A. Kronz. Managing of process key performance indicators as part of the aris methodology. In *Corporate Performance Management: Aris in Practice*, pages 31–44. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-30787-7. URL http://dx.doi.org/10.1007/3-540-30787-7_3.

[50] P. Kueng and A. J. W. Krahn. Building a process performance measurement system: some early experiences. *Journal of Scientific and Industrial Research*, 58(3/4):149–159, 1999.

[51] C. Mayerl, K. Hüner, J.-U. Gaspar, C. Momm, and S. Abeck. Definition of metric dependencies for monitoring the impact of quality of services on quality of processes. In *Sec-*

ond IEEE/IFIP International Workshop on Business-driven IT Management (Munich), pages 1–10, 2007.

[52] J. Mendling, H. A. Reijers, and J. Cardoso. What makes process models understandable? In *BPM*, pages 48–63, 2007.

[53] P. J. Meyer. *What would you do if you knew you couldnÕt fail? Creating S.M.A.R.T. Goals*. Meyer Resource Group, Incorporated, The, 2003. ISBN 9780898113044.

[54] C. Momm, R. Malec, and S. Abeck. Towards a model-driven development of monitored processes. In *Wirtschaftsinformatik (2)*, pages 319–336, 2007.

[55] C. Momm, M. Gebhart, and S. Abeck. A model-driven approach for monitoring business performance in web service compositions. In M. Perry, H. Sasaki, M. Ehmann, G. O. Bellot, and O. Dini, editors, *ICIW*, pages 343–350. IEEE Computer Society, 2009.

[56] D. L. Moody. The é;physicsé; of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Software Eng.*, 35(6):756–779, 2009.

[57] B. Mora, F. García, F. Ruiz, and M. Piattini. Model-driven software measurement framework: A case study. In B. Choi, editor, *QSIC*, pages 239–248. IEEE Computer Society, 2009. ISBN 978-0-7695-3828-0.

[58] B. Motik, P. F. Patel-Schneider, and B. C. Grau. OWL 2 Web Ontology Language Direct Semantics, 2009. URL `http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/`.

[59] E. Motta. *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*. IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1999. ISBN 1586030035.

[60] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77 (4):541 –580, apr 1989. ISSN 0018-9219. doi: 10.1109/5.24143.

[61] D. Nardi and R. J. Brachman. An introduction to description logics. In *Description Logic Handbook*, pages 1–40, 2003.

[62] D. Narong. *Activity-based Costing and Management: Total Quality Management Solution to Quality Cost Shortcomings of the Traditional Cost Accounting Systems*. PhD thesis, California State University, 2008.

[63] A. Neely, M. Gregory, and K. Platts. Performance measurement system design: A literature review and research agenda. *International Journal of Operations & Production Management*, 25:1228 – 1263, 2005. doi: 10.1108/01443570510633639.

[64] M. Netjes, H. A. Reijers, and W. M. P. van Der Aalst. *Supporting the BPM life-cycle with FileNet*, volume 6, pages 497–508. Citeseer, 2006. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.6750&rep=rep1&type=pdf.

[65] O. O. of Government Commerce). Information technology infrastructure library (ITIL) v3. Collection of books, 2007.

[66] O. M. G. (OMG). Human-usable textual notation (hutn) specification, August 2004.

[67] O. M. G. (OMG). Umltm profile for modeling quality of service and fault tolerance characteristics and mechanisms specification, May 2006. URL http://www.omg.org/docs/formal/08-04-05.pdf.

[68] O. M. G. (OMG). Business process model and notation (bpmn) version 2.0, Jan 2011.

[69] M. Papazoglou and F. Leymann. Business process management. In *IEEE Encyclopedia of Software Engineering*. John Wiley and Sons, 2008.

[70] C. Pedrinaci, D. Lambert, B. Wetzstein, T. van Lessen, L. Cekov, and M. Dimitrov. Sentinel: a semantic business process monitoring tool. In *OBI*, page 1, 2008.

[71] Pellet. (owl) 2 reasoner for java, 2011. URL http://clarkparsia.com/pellet.

[72] B. Pernici, editor. *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings*, volume 6051 of *Lecture Notes in Computer Science*, 2010. Springer. ISBN 978-3-642-13093-9.

[73] P. O. Plugin. Ontology editor for the semantic web, 2011. URL http://protege.standford.edu/overview/protege-owl.html.

[74] V. Popova and A. Sharpanskykh. Formal goal-based modeling of organizations. In *MSVVEIS*, pages 19–28, 2008.

[75] V. Popova and A. Sharpanskykh. Process-oriented organisation modelling and analysis. *Enterprise IS*, 2(2):157–176, 2008.

[76] V. Popova and A. Sharpanskykh. Formal analysis of executions of organizational scenarios based on process-oriented specifications. *Applied Intelligence*, 2009. doi: 10.1007/s10489-009-0192-9.

[77] V. Popova and A. Sharpanskykh. Modeling organizational performance indicators. *Inf. Syst.*, 35(4):505–527, 2010.

[78] V. Popova and A. Sharpanskykh. Formal modelling of organisational goals based on performance indicators. *Data Knowl. Eng.*, 70(4):335–364, 2011.

[79] Protégé. An ontology and knowledge base editor, 2011. URL http://protege.standford.edu.

[80] (RACER). Renamed abox and concept expression reasoner, 2011. URL http://www.racer-systems.com.

[81] H. O. Reasoner. The new kid on the (owl) block, 2011. URL http://hermit-reasoner.com.

[82] A. Rolstadås and I. F. for Information Processing. *Benchmarking: theory and practice*. Ifip International Federation for Information Processing. Chapman and Hall, on behalf of the International Federation for Information Processing, 1995. ISBN 9780412626807. URL http://books.google.es/books?id=DglPAAAAMAAJ.

[83] M. L. Rosa, M. Dumas, A. H. M. ter Hofstede, and J. Mendling. Configurable multi-perspective business process models. *Inf. Syst.*, 36(2):313–340, 2011.

[84] A. Ruiz-Cortés, O. Martín-Díaz, A. D. Toro, and M. Toro. Improving the automatic procurement of web services using constraint programming. *Int. J. Cooperative Inf. Syst.*, 14(4):439–468, 2005.

[85] J. E. Rumbaugh. Notation notes: Principles for choosing notation. *JOOP*, 9(2):11–14, 1996.

[86] J. E. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual*. Addison-Wesley-Longman, 1999. ISBN 978-0-201-30998-0.

[87] L. Sánchez-González, F. García, J. Mendling, and F. Ruiz. Quality assessment of business process models based on thresholds. In *OTM Conferences (1)*, pages 78–95, 2010.

[88] A. Scheer, W. Jost, H. Heß, and A. Kronz. *Corporate Performance Management: Aris in Practice*. Springer, 2006. ISBN 9783540307037. URL http://books.google.es/books?id=nWzqV7909YEC.

[89] A.-W. Scheer and Wolfram. From process documentation to corporate performance management. In *Corporate Performance Management: Aris in Practice*, pages 1–6. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-30787-7. URL http://dx.doi.org/10.1007/3-540-30787-7_1.

[90] A.-W. Scheer, O. Thomas, and O. Adam. *Process Modeling using Event-Driven Process Chains*, pages 119–145. John Wiley and Sons, Inc., 2005. ISBN 9780471741442. doi: 10.1002/0471741442.ch6. URL http://dx.doi.org/10.1002/0471741442.ch6.

[91] A. Shahin and M. A. Mahbod. Prioritization of key performance indicators: An integration of analytical hierarchy process and goal setting. *International Journal of Productivity and Performance Management*, 56:226 – 240, 2007.

[92] A. Sharpanskykh and J. Treur. Verifying interlevel relations within multi-agent systems. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *ECAI*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 290–294. IOS Press, 2006. ISBN 1-58603-642-4.

[93] P. Soffer and Y. Wand. On the Notion of Soft-Goals in Business Process Modeling. *Business Process Management Journal*, 11:663–679, 2005. doi: 10.1108/14637150510630837.

[94] V. A. (Tilburg), S. B. (UCBL), M. B. (UOC), O. D. (USTUTT), M. H. (UCBL), W. van den Heuvel (Tilburg), D. K. (USTUTT), B. K. (Tilburg), F. L. (USTUTT), M. M. (Tilburg), K. M. (UCBL), C. N. (UOC), M. P. (Tilburg), and B. W. (USTUTT). Survey on business process management. Deliverable of S-Cube. http://www.s-cube-network.eu, July 2008.

[95] K. Tornberg, M. Jämsen, and J. Paranko. Activity-based costing and process modeling for cost-conscious product design: A case study in a manufacturing company. *International Journal of Production Economics*, 79(1):75–82, Sept. 2002. ISSN 09255273. doi: 10.1016/S0925-5273(00)00179-1. URL http://linkinghub.elsevier.com/retrieve/pii/S0925527300001791.

[96] G. Vaidyanathan. A framework for evaluating third-party logisics. *ACM*, 48:89–94, 2005.

[97] W. van den Heuvel. Survey on business process management. Technical report, 2008.

[98] W. M. van der Aalst, A. H. ter Hofstede, and M. Weske. Business process management: A survey. In *Business Process management*, volume 2678, pages 1–12. Springer, 2003.

[99] W. M. P. van der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer Berlin / Heidelberg, 1997. ISBN 978-3-540-63139-2. URL http://dx.doi.org/10.1007/3-540-63139-9_48.

[100] W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[101] W. M. P. van der Aalst. Business process simulation revisited. In J. Barjis, editor, *EOMAS*, volume 63 of *Lecture Notes in Business Information Processing*, pages 1–14. Springer, 2010. ISBN 978-3-642-15722-6.

[102] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, Nov. 2003. ISSN 0169-023X. doi: 10.1016/S0169-023X(03)00066-1. URL http://dx.doi.org/10.1016/S0169-023X(03)00066-1.

[103] W. M. P. van der Aalst, M. Pesic, and M. Song. Beyond process mining: From the past to present and future. In Pernici [72], pages 38–52. ISBN 978-3-642-13093-9.

[104] C. Vercellis. *Business Intelligence: Data Mining and Optimization for Decision Making*. John Wiley and Sons, 2009. ISBN 9780470753859. URL http://books.google.es/books?id=YvAhkBE6sBYC.

[105] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007. ISBN 978-3-540-73521-2.

[106] B. Wetzstein, Z. Ma, A. Filipowska, M. Kaczmarek, S. Bhiri, S. Losada, J.-M. Lopez-Cob, and L. Cicurel. Semantic business process management: A lifecycle based requirements analysis. In M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, and N. Stojanovic, editors, *SBPM*, volume 251 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[107] B. Wetzstein, Z. Ma, and F. Leymann. Towards measuring key performance indicators of semantic business processes. In *BIS*, pages 227–238, 2008.

[108] M. M. Yasin. The theory and practice of benchmarking: then and now. *Benchmarking An International Journal*, 9(3):217–243, 2002. URL http://www.emeraldinsight.com/10.1108/14635770210428992.