PPI Definition and Automated Design-Time Analysis (v 1.0)

Adela del-Río-Ortega, Manuel Resinas, Antonio Ruiz-Cortés
{adeladelrio,resinas,aruiz}@us.es

Applied Software Engineering Research Group
University of Seville, Spain
March 2012

Technical Report ISA–12–TR–02

This report was prepared by the

Applied Software Engineering Research Group (ISA)
Department of computer languages and systems
Av/ Reina Mercedes S/N, 41012 Seville, Spain
http://www.isa.us.es/

Support: This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project SETI (TIN2009-07366) and by the Andalusian Government under projects ISABEL (TIC-2533) and THEOS (TIC-5906).

Contents

Abstract

The measurement of process performance and its analysis is crucial for the consecution of strategic and operational goals in any process-oriented organisation. These activities, like other activities carried out during a business process lifecycle, are considered time-consuming and error-prone. Therefore, providing an automated support for them is very appealing from a practical point of view. In this paper, we focus on providing such automated support for the definition and analysis of Process Performance Indicators (PPIs) at design-time. To this end, we present PPINOT, a framework that relies on Description Logics (DLs) to define and analyse PPIs automatically. The advantages of PPINOT are three. First, 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. Second, it enables a seamless relationship between PPIs and business process models, which makes the use of PPIs along the business process lifecycle easier. Third, it specifies and implements three mostly novel analysis operations families that can assist business process analysts during PPIs definition and business process evolution. Furthermore, since PPINOT relies on Description Logics (DLs), new analysis operations can be defined to extend those described in this paper, being the reasoning power of DLs the only limit.

Chapter 1

Introduction

Business Process Management (BPM) intends to support business processes using methods, techniques, and software to design, enact, control and analyse operational processes involving humans, organisations, applications, documents and other sources of information [van der Aalst et al., 2003a]. There exists a growing interest in business processes (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 [Alexander Grosskopf and Weske, 2009].

To achieve this improvement of processes, it is important to evaluate the performance of business processes. Key Performance Indicators (KPIs) are recognized as a key asset to carry out this evaluation [(Tilburg) et al., 2008]. KPIs are quantifiable metrics that an organisation uses to measure performance in terms of meeting its strategic and operational objectives Neely et al. [2005]. KPIs provide critical information to the organisation for monitoring and predicting business performance in accordance with strategic objectives (Tilburg) et al. [2008]. KPIs can thus be defined for organisations, in order to define and measure progress towards their goals. In our work we focus on those KPIs defined to measure performance concretely in the business processes defined within the organisation; understanding the measurement of process performance as "the formal planned monitoring of process execution and the tracing of results to determine the effectiveness and efficiency of the process" [of Business Process Management Professionals (ABPMP), 2009]. This kind of KPIs can be coined with the acronym PPIs (Process Performance Indicators). Several research proposals ([Momm et al., 2007; Adela del Río-Ortega and Ruiz-Cortés, 2010; Messer et al., 2011] for instance) and communities (e.g. [(SAP), 2007; Orientation, 2010]) use this acronym, but no definition has been found. Thus, as a result of the literature review, we found that a PPI can be defined as a measure that reflects the critical success factors of a business process defined within an organisation, in which its target value reflects the objectives pursued by the organisation with that business process (some examples of PPI can be found in Table 2.1). Nowadays, many methodologies and frameworks like, for instance, COBIT [ISACA, 2009], ITIL [of Government Commerce), 2007] or the EFQM [EFQM, 2010] excellence model, confirm this importance by including the definition of these PPIs within their recommendations as a means to evaluate the performance of the existing business processes.

In order to make the aforementioned evaluation of business processes, it is necesary to inte-

grate the management of PPIs into the whole business process lifecycle [Weske, 2007; del Río-Ortega and Resinas, 2009] as follows: in the design and analysis phase, PPIs should be modelled together with the business process. Furthermore, 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. During the configuration phase, the instrumentation of the process necessary to take the measures must be defined. During the business process enactment, when valuable execution data is gathered, the PPIs' values have to be calculated and the monitoring of these PPIs should be carried out. For instance, this can be done based on execution logs that store information about the process such as the start or end of activities. Finally, during the evaluation phase, the monitoring information obtained in the previous phase will help to identify correlations and predict future behaviour.

Handling business process lifecycle is recognized as an error-prone and time-consuming activity which requires of a formal basis and automated support [van der Aalst et al., 2003b; Weske et al., 2004]. The same reasoning can be applied to the management of the PPI lifecycle introduced above. An appropriate definition of PPIs is key to enable this automated support. Such definition must fulfill the following requirements:

1. **Seamless BP-PPI relationship**: The representation of the relationship between PPIs and business process elements enables the use of PPIs together with other business process analysis techniques and helps in the instrumentation of the information systems that is necessary to obtain measures automatically.

2. **Expressiveness**: traditionally, only those PPIs related to control flow and time were considered, but those related to data objects or calculated applying a function over several measures must also be taken into account.

3. **Enable automated analysis**: 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. In order to perform such an automated analysis, a formal foundation is required for the definition of PPIs; formal models do not leave any scope for ambiguity and increase the potential for analysis (since formal languages may have associated analysis techniques) [van der Aalst et al., 2003b; van der Aalst, 1996; Wodtke and Weikum, 1997].

Unfortunately, in practice, such definition is missing; PPIs are usually defined in an informal and ad-hoc way, since there not exists any standard model to define such PPIs over business processes (defined for example in BPMN (OMG) [2009]). Furthermore, although there are several research proposals to define PPIs, none of them are well-suited because they do not meet the aforementioned requirements together(*cf.* Section 8 for more details).

The contribution of this paper is twofold. First, we present an ontology, called PPINOT Ontology, that fulfill the aforementioned requirements for the definition of PPIs. Second, taking advantage from this ontological definition of PPIs, we propose a mechanism to analyse them in order to obtain information useful in assisting during evolution of business process or predicting future behaviour. The main benefits of our contribution can be summarised as follows:

1. The model for the definition of PPIs also includes the connection to the business process elements (fulfilling thus the first requirement established above)

2. The definition of a wide variety of PPIs is supported, including those related to data objects. Furthermore, an expressive analysis period of a PPI can be defined. In fact, our ontology supports the definition of PPIs that, as far as we know, cannot be expressed in any other similar proposal (*cf.* Section 8).

3. Since PPINOT ontology has been defined in OWL DL, automated reasoners can be used to infer knowledge from PPIs in different ways:

   - *PPI-BP elements interaction*: It is possible to obtain information about the way PPIs and business process elements influence each other. Two cases can be distinguished:
     - To obtain information about which are the elements of a business process measured by a PPI (i.e. *given a particular indicator, what are the business process elements involved in its definition?*). A scenario where this information can be useful is when a PPI is costly and it must be replaced with others PPIs, in order to assure that every element of the business process that was measured before is measured in the new case.
     - To obtain information about which PPIs are associated to a concrete element or part of a business process (i.e. the opposite direction, *given a particular business process element, what are the PPIs associated to them?*) This can help in assisting during the evolution of business process (e.g. if a part of the business process has evolved and is modified, if an activity is deleted for instance, this analysis allows to identify which PPIs will be affected and should be updated).
   - *PPI internal information*: It can also be obtained other kind of information related to the PPIs definition such as *how many PPIs need several measures to calculate their values?*, *how many PPIs measure time?* or *how many PPIs are defined on a concrete analysis period?*.
   - *Dependencies*: Finally, dependencies between measures can be automatically obtained from the ontology. This can help, for instance, to detect if two PPIs inversely dependent are tried to be optimised).

A prototype has been developed to show the use of PPINOT ontology and the benefits of its DL-based semantics.

Furthermore, we have applied our proposal in several real scenarios: the Information Technology Department of the Andalusian Health Service and the Justice and Public Administration Department of the Andalusian Local Government, in order to study the applicability of our solution.

This paper extends the contribution made in [Adela del Río-Ortega and Ruiz-Cortés, 2010]. In particular, regarding the ontology, minor changes (refinements and extensions of some concepts) has been made. With respect to the analysis, except for the dependencies between PPIs (that have been also refined and extended) everything is new. The tool is also a new contribution.

The remainder of this techinacl report is organised as follows. In Chapter 2 we present a motivating scenario using a real case. A brief introduction to Description Logics and OWL-DL is presented in Chapter 3. Then in Chapter 4 we establish some concepts regarding business process models useful for subsequent sections. In Chapter 5 we propose an ontology for the definition

of PPIs, that serves as a basis for the automated analysis presented in Chapter 6, where the way reasoners can be used to automatically infer information from PPINOT Ontology is described. Some implementation details are given in Chapter 7. In Chapter 8 we present related work. Finally, Chapter 9 draws the conclusions from our work, summarizes the paper and outlines our future work.

Chapter 2

Case Study: Process of the Request For Change Management

In this section, we present an excerpt of a real scenario that takes place in the context of the Information Technology Department of the Andalusian Health Service. We focus on the business process of managing Request for Changes in the 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 2.1.

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. According to several factors like the availability of resources, the requirements requested, and others, the RFC will be either approved, cancelled, or raised to a committee for them to make the decision.

The RFC is a DataObject with its DataStates and DataProperties defined. Amongst others, the RFC has three DataProperties: *Project*, which refers to the project to which such RFC is associated, *InformationSystem*, that defines the information system for which a change is required or to which that change affectt and *priority*, that makes reference to the importance of resolving that RFC.

After modelling the process, this department also defined a set of indicators associated with it, but 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 2.1). Target values ("Target V" for short) reflected in this table are invented due to privacy reasons. The responsible for all these PPIs defined in Table 2.1 is the "planning and quality manager" and does not appear in the table due to space constraints.

In the current status of this scenario, it is not easy to identify the relationship between the PPIs listed in Table 2.1 and the elements of the process depicted in Figure 2.1, making difficult, thus, the instrumentation of the process in order to take the values of PPIs. Furthermore, the IT Department of the Andalusian Health Service is about to modify the current business process that supports the RFC management to fulfill new requirements, and they need to update PPIs according to the changes made, defining new ones if it becomes necessary. As a consequence, they are also interested in identifying possible dependencies between existing PPIs and the new ones to be

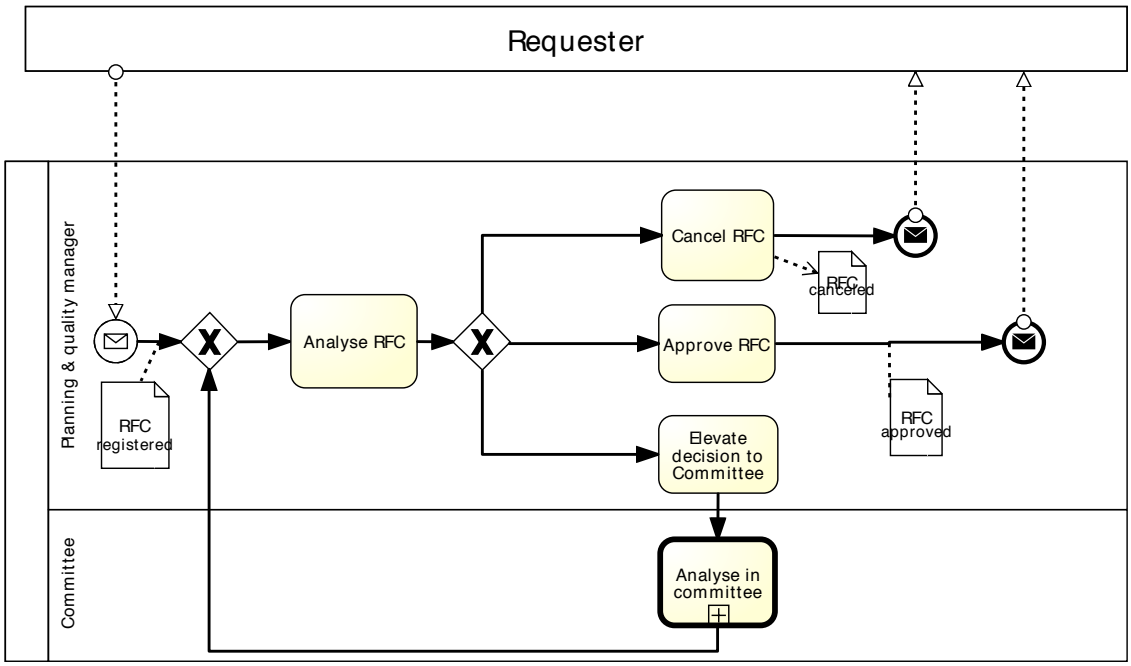

Figure 2.1: Process of the request for change management (simplified)

defined. The existing support to manage PPIs together with business processes does not allow them to carry out these activities in an automatic way. This situation together with the fact that they are error prone and infeasible by humans in reasonable time were the reasons for us to define the requirements established in Section 1 and to present the approach proposed in the following sections.

Name	Description	Target V	Periodicity
PPI1	RFCs cancelled-registry error from RFCs registered)	1	weekly
PPI2	Average time of committee decision	1 day	weekly
PPI3	corrective RFCs from approved RFCs	0	weekly
PPI4	perfective and adaptive RFC from approved RFCs)	4	weekly
PPI5	Average time of the "analyse RFC" activity	1 day	weekly
PPI6	Number of RFCs with the state "in analysis"	2	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	2 days	monthly

Table 2.1: PPIs defined for the RFC management process

Chapter 3

OWL-DL in a Nutshell

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 Description Logics (DLs). DLs 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 [Nardi and Brachman, 2003]. As exemplified in Table 3.1 and Table 3.2 [2], OWL consists a rich set of knowledge representation constructs that can be used to formally specify PPI-domain knowledge, which in turn can be exploited by description logic reasoners for purposes of inferencing, i.e., deductively inferring new facts from knowledge that is explicitly available [Bhatt et al., 2009]. Knowledge representation systems based on DLs 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. As stated in [Bhatt et al., 2009], 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).

Our use of the OWL language to represent the PPINOT ontology is driven by several reasons: first, the definition of a set of PPIs for a business process fits nicely into the way DLs express their concepts and, hence, OWL-DL provide a very natural way to describe the problem; second, the fact that OWL [Motik et al., 2009] is industry standard and is recommended by the W3C for the representation of ontologies. Furthermore, numerous semantic web tools supporting OWL, for example, Protégé [Gennari et al., 2002; Protege, 2011] and its associated OWL Plugin [Plugin,

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

[2] In both tables and in Section 6 a syntax commonly used for DLs [Baader et al., 2003] is utilised

Table 3.1: OWL axioms

Axiom	**DL Syntax**	**Example**
Sub class	$C_1 \sqsubseteq C_2$	$BaseMeasure \sqsubseteq InstanceMeasure \sqsubseteq MeasureDefinit$
Equivalent class	$C_1 \equiv C_2$	$MeasureDefinition \equiv InstanceMeasure \sqcup ProcessMea$
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	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq 1isDefinedOver$
Inverse functional property	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq 1isUsedBy^-$

Table 3.2: OWL class constructors

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	$\leq nP$	$\leq 1appliesTo$
Min cardinality	$\geq nP$	$\geq 1isCalculated$

2011], have been already developed in the open-source community. In addition, powerful reasoning systems that support reasoning with ontologies represented in the OWL language (like, for instance, Racer [(RACER), 2011], Hermit [Reasoner, 2011] and Pellet [Pellet, 2011]) have been developed, allowing thus to automatically and efficiently analyse PPIs.

Chapter 4

Business Process Models Considerations

In this section we establish some considerations regarding process models we will use later on in our proposal by means of an ontology heavily based on BPMN. We do not intend to provide a comprehensive ontology that covers all aspects of BPMN (in [Francescomarino et al., 2009, 2011; Nicola et al., 2007] some examples of ontological descriptions for business process models can be found), but we focus on those aspects that will be needed for the definition of PPIs and their analysis. The ontology is depicted in Figure 4.1. Note that, although the ontology is based on BPMN, any other notation or language can be used provided that their concepts can be mapped to those of this ontology.

A process is composed of different `BPElements`: we consider `Event`, `Activity`, `Gateway`, `DataObject` and `Pool`. Every `BPElement` (except `DataObjects`) can `directlySucceeds` or `directlyPrecedes` another `BPElement`, i.e., it is located after or before that `BPElement` in the `Process` respectively (this information is represented through the sequence flow in BPMN). Note that we depict in Figure 4.1 the objectProperties `succeeds` and `precedes`. These are the transitive properties of the previous ones. Furthermore, when running business processes, `BPElementInstances` (instances of `BPElements` are created. There can be more than one instance per `BPElement` in the same business process instance as a consequence of loops. There also exist precedes and succeeds relationships between `BPElementInstances` with the same meaning as in the case of `BPElements`, but in this case they are called `instanceSucceeds` and `instancePrecedes`.

When defining `DataObjects`, the user can specify `DataProperties`, that contain specific information about the `DataObject`. Furthermore `DataObjects` can have a state (`DataState`) defined (this concept corresponds to the one defined in BPMN). The set of possible `DataStates` for each `DataObject` is defined by the user when defining such `DataObject`. In the case of `Activities` and `Pools`, they are not the ones who have a state associated but their instances during process execution. BPMN also defines the possible values for `ActivityState` (applicable to activities and pools instances): ready, active, withdrawn, completing, completed, failing, failed, terminating, terminated, compensating and compensated. Finally, `Events` instances also can have a state defined. The set of possible values we consider for `EventState` is: `none`, `waiting`, `completed`

Furthermore, there exists a relation between `Activity` and `DataObject`, called `dataOutputAssociatio` (also considered in BPMN), that refers to the fact that a `DataObject` can be the result or can be

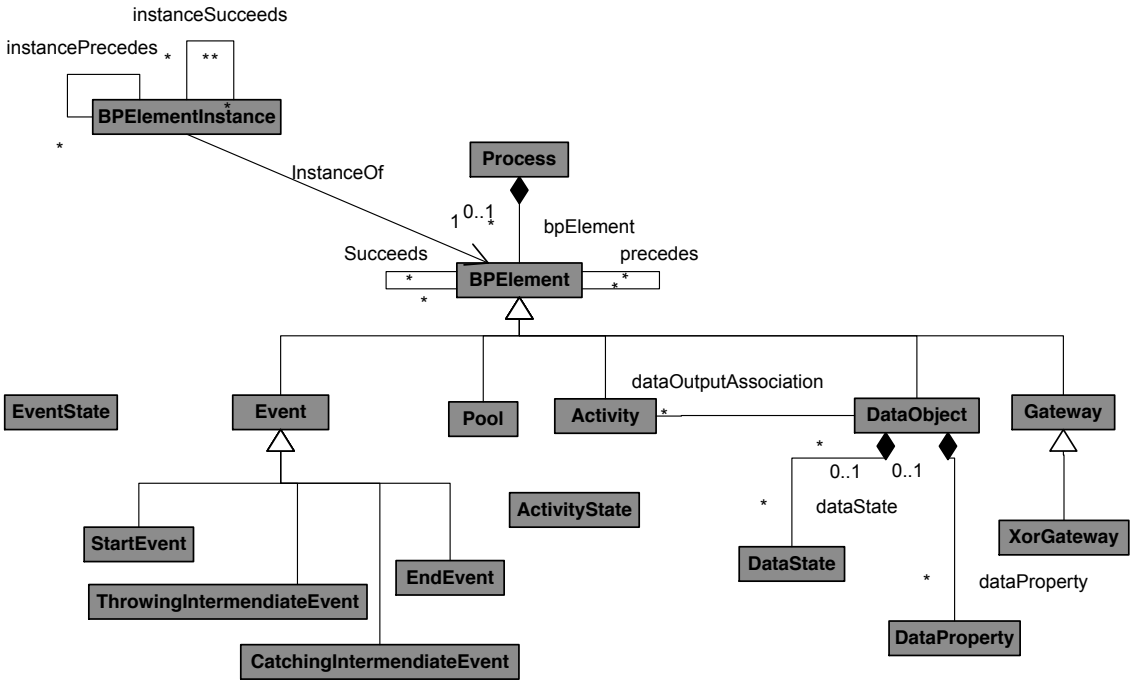

Figure 4.1: Our partial ontology of BPMN focused on those aspects that will be used in this paper

modified by an `Activity`.

Table 4.1 summarizes the set of object properties depicted in Figure 4.1 and introduced above, together with their characteristics and inverses.

Table 4.1: Summary of object properties of our partial BPMN ontology

ObjectProperty	**Characteristics**	**Inverse Property**
instanceOf	functional	hasInstance
succeeds	Transitive	precedes
directlySucceeds	SubPropertyOf succeeds	directlyPrecedes
instanceSucceeds	Transitive	instancePrecedes
instanceDirectlySucceeds	SubpropertyOf instanceSucceeds	instanceDirectlyPrecedes
dataOutputAssociation	-	-
dataState	-	-
dataProperty	-	-
BPElement	-	-

Chapter 5

PPINOT Ontology

In the following, we present our PPINOT ontology defined in OWL DL. This ontology is an extended version of the one presented in[Adela del Río-Ortega and Ruiz-Cortés, 2010]. Figure **??** depicts the UML representation (as proposed in[Brockmans et al., 2004]) of the PPINOT ontology. Those elements depicted in a dark grey color represent concepts of BP models (already introduced in Section 4)

As shown in Figure 5.1, a PPI is referred to by means of a `hasName`, described through its `hasDescription`, has a `hasTarget` restriction, which is the objective to achieve, has an `analysisPeriod`, that corresponds to the time period where such target must be achieved (this concept wil be explained in detail in Subsection 5.4), and a `hasResponsible`, that is the human resource in charge of such PPI. PPIs are `relatedTo` a process, and are defined by a `MeasureDefinition`. In order to define a measure, we need to specify its `hasName`, `hasScale` (domain, i.e. set of values with defined properties, e.g. natural, integer, float, map, [0..100]) and, in some cases, `hasUnitOfMeasure` (e.g. seconds, hours or euros). `MeasureDefinitions` are used to define `Measures`, and these `Measures` take values (`hasValue`) from one or several instances (`hasInstance`) in different time instants (`hasTime`).

When formulating measures for PPIs, we can identify two classifications attending to different criteria:

- Attending to whether we consider one single process instance to take the measure or we calculate the value using a set of instances, we can distinguish between **`InstanceMeasures`** and **`ProcessMeasures`**. For instance, in our case study, an `InstanceMeasure` could be "the duration of the activity *analyse RFC*" for a given process instance, and a `ProcessMeasure` "the average duration of that activity in the last month". Usually, most PPIs will be defined using `ProcessMeasures`.

- Attending to which is the way to get the value of the measure, there are `BaseMeasures`, `AggregatedMeasures` and `DerivedMeasures`. In the following subsections we detail all of them.

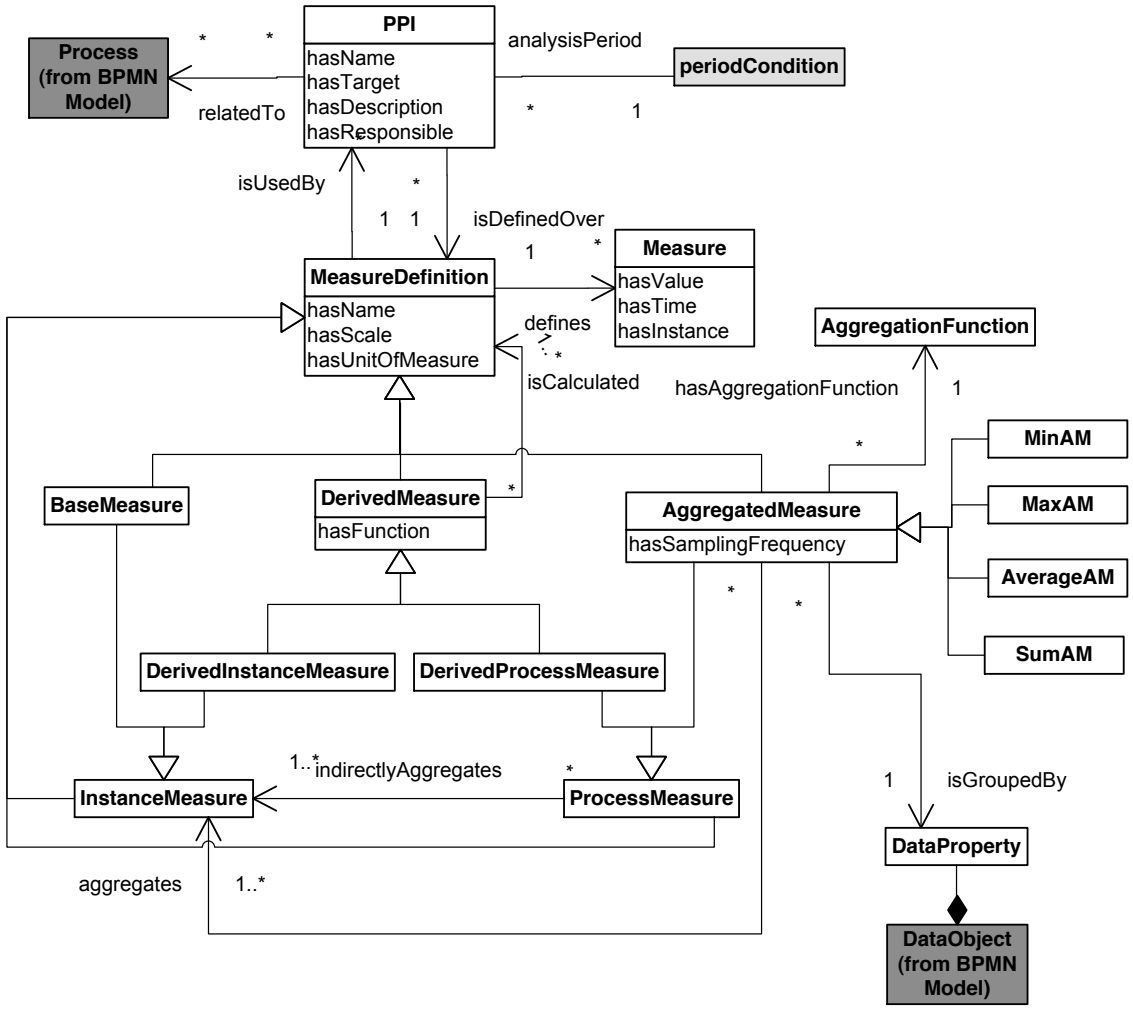

Figure 5.1: Measure definition in PPINOT Ontology

5.1 BaseMeasures

In this case, the value of the measure is obtained by executing certain measurement method over a single process instance, i.e. it is an `InstanceMeasure` that is not calculated using any other measure. Depending on what needs to be measured, a different measurement method will be applied. As depicted in Figure 5.2, we can measure time (**`TimeMeasure`**), the number of times that something happens (**`CountMeasure`**), whether certain condition is fulfilled (**`ConditionMeasure`**) or we can obtain the value of a `DataProperty` of a `DataObject` (**`DataMeasure`**).

`TimeMeasure` : In this case, the measure is defined as the duration between the occurrence of two `TimeInstantConditions` (`from` and `to`). These `TimeInstantConditions` can be associated with the start or the end of an activity (`ActivityStart`, `ActivityEnd`) or a

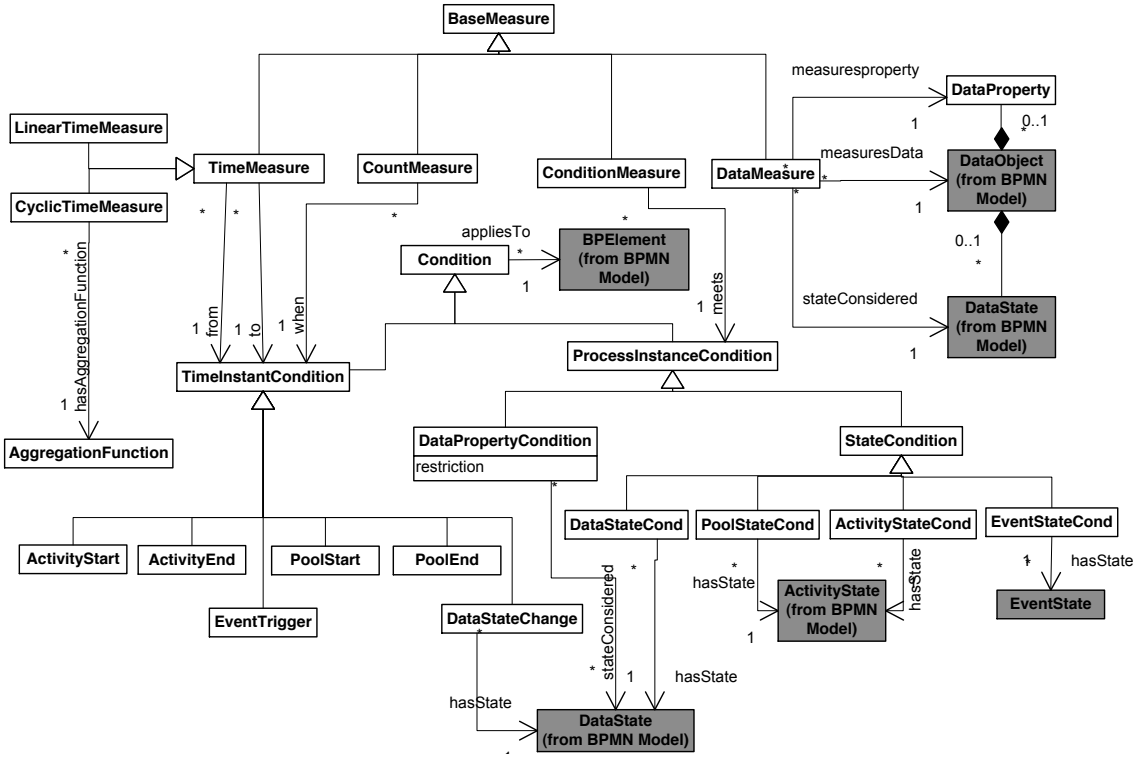

Figure 5.2: PPINOT Ontology (`BaseMeasure` classification)

process contained in a pool (`PoolStart`, `PoolEnd`), with the trigger of an event(`EventTrigger`), or with the change of the state of a data (`DataStateChange`). In this last case, the instant considered is the one in which the data changes to the state specified through `hasState`. These `TimeMeasures` can be subdivided into two subclasses: `LinearTimeMeasures` and `CyclicTimeMeasures`. This distinction makes sense if the `TimeMeasure` is taken between elements located within a loop. In the first case (`LinearTimeMeasures`), the measure is defined taking into account the first occurrence of the `TimeInstantCondition from` and the last occurrence of the `TimeInstantCondition to`. In the second case (`CyclicTimeMeasures`), the measure is defined by aggregating (using `hasAggregationFunction`) the values for the time between the pairs of the `TimeInstantConditions from` and `to` of each iteration (obtaining thus a single value for each instance). An example of `TimeMeasure` is "the duration of the analysis of a RFC". Since the activity *analyse RFC* is located within a loop, we can distinguish two cases: the `LinearTimeMeasure`, where the `from` and `to` conditions match the first instance of the `ActivityStart` and the last instance of the `ActivityEnd` of this activity; and the `CyclicTimeMeasure`, where an `AggregationFunction` must be defined (e.g., "the average duration of the analysis of a RFC in a process instance") and applied to the different values obtained for the time between the pairs of `from` (`ActivityStart`) and `to` (`ActivityEnd`) conditions of each iteration.

CountMeasure : In this case, the measure is defined as the number of times a TimeInstantCondition is met. For instance, "The number of times a RFC is analysed" is an example of this measure, and it is measured by counting the number of times condition ActivityEnd for the activity *analyse RFC* is met in a single instance of the process.

ConditionMeasure : In this case, the measure is defined as the fulfillment of certain ProcessInstanceConditions, i.e., it checks if certain ProcessInstanceConditions are being (for running process instances) or have been (for finished process instances) met. The measures defined by a ConditionMeasure are always boolean values. We classify them into two groups: StateConditionMeasure and DataPropertyConditionMeasure (not reflected in Figure **??** to maintain clarity). In the case of **StateConditionMeasure**, whether a FlowElement (that can be a DataObject, a pool, an activity, or an intermediate catching event) is currently in a given state or not is checked. An example of this kind of measure is "check if the activity *analyse RFC* in a given Process Instance is in state *active*", i.e., if the RFC is being analysed at that moment. In the other possibility, **DataPropertyConditionMeasure**, we can check if one property of a DataObject meet a particular restriction, e.g. RFC with priority = "high".

DataMeasure : In this case, the measure is defined as the value of a DataProperty contained in a DataObject, e.g. number of information systems that a RFC affects to. It is possible to specify the state in which the DataObject must be when the measure is performed (stateConsidered).

Note that all these BaseMeasures are defined over a concrete BPElement (or two in the case of TimeMeasures) of the associated BPMN diagram. This is represented through relation appliesTo in the case of measures with conditions. Depending on the kind of Condition, this element will be an activity, a pool, an event or a DataObject (we do not depict all these correspondences in Figures 5.2 and **??** for the sake of simplicity and readability). In the case of DataMeasures, this is represented through the object property measuresData.

5.2 DerivedMeasures

In this case, the measure is defined by performing a mathematical function over two or more MeasureDefinitions. Depending on whether these MeasureDefinitions are instance or process measures, the result will be a **DerivedInstanceMeasure** or a **DerivedProcessMeasure** respectively. Note that it is not allowed to mix InstanceMeasures and ProcessMeasures in the same DerivedMeasure. Examples for both cases are respectively "the percentage of time spent in the activity *analyse RFC* with respect to the duration of the whole process", and "percentage of rejected RFCs with respect to all the registered RFCs".

5.3 AggregatedMeasures

In this case, the measure is defined by applying a certain aggregationFunction on the set of values of an InstanceMeasure belonging to different instances, to obtain one single

value. Property aggregates is used to specify the InstanceMeasure aggregated. Depending on whether the aggregationFunction applied is minimum, maximum, average or sum, these measures can be MinAM, MaxAM, AvgAM, SumAM respectively. An example of AggregatedMeasure is "the number of RFC rejected in the last year", which is calculated through the aggregation function SUM (it is a SumAM).

It is worth mentioning that when the InstanceMeasure aggregated by an AggregatedMeasure is a DerivedInstanceMeasure, the AggregatedMeasure aggregates such DerivedInstanceMeasu and indirectlyAggregates the measures used to calculate such DerivedInstanceMeasure.

Moreover, there are several issues to be considered regarding which are the process instances whose measure will be used to calculate the AggregatedMeasure.

First, a samplingFrequency 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). In the case a sampling frequency is not defined, it is assumed that there not exists such restriction (all the instances within the analysis period will be considered).

Second, when aggregating measures, it may be useful to group them by a DataProperty of a certain DataObject, for instance, "the number of RFCs per project". In such a case, the number of RFCs is added (SumAM) and grouped (isGroupedBy) by the DataProperty project. The measure is a map, with a value for each project.

Third, when defining an AggregatedMeasure, the set of instances that must be taken into account when aggregating (that is, the temporal range to consider when measuring) is defined by means of the analysisPeriod specified in the PPI defined by such AggregatedMeasure. This analysisPeriod is explained in detail in next subsection.

5.4 AnalysisPeriod

The analysisPeriod (highlighted in Figure 5.3 with a light grey colour) must be understood as a temporal condition that every process instance must hold for its measure's value to be included in the comparison with the target value. In the case of a PPI defined over an InstanceMeasure, this analysisPeriod makes reference to the set of instances whose value must be compared to the target one. If the PPI is defined over a ProcessMeasure, this analysisPeriod determines the instances that have to be taken into account when calculating (either aggregating or through a mathematical function) the value of the ProcessMeasure.

The condition (PeriodCondition) can be basic or composed. The basic one, called TemporalCondition, can be of different types (ConditionType: $>$, $\geq$, $<$ and $\leq$), and is defined between the start/end of process instances and the moments in time (TimeDefinition); for instance start $>$ 15-09-2011 will take into account all the process instances that started after the mentioned date, and end $>$ 31-8-2011 will take into account all the process instances that ended after such date. Furthermore, these TimeDefinitions can refer to either an AbsoluteTimeDefinitior defined by a concrete date, like the previous examples, or to a RelativeTimeDefinition defined with respect to the current instant (i.e. two months ago). Composed PeriodConditions (Composition) can be obtained by composing TemporalConditions by means of an AND (AndComposition) and/or an OR (OrComposition) operator. For instance, it could be interesting to measure the number of RFCs rejected during the last holidays' periods, both Christmas

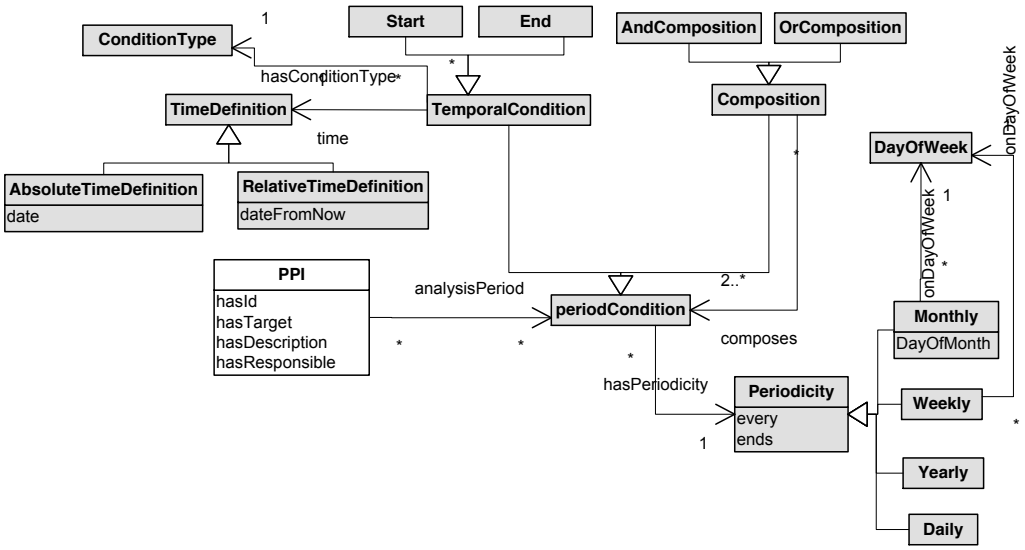

Figure 5.3: PPINOT Ontology (Analysis period definition)

and summer. Thus, only the measures whose process instances finished between 1-8-2010 and 31-8-2010, or between 18-12-2010 and 4-1-2011 would be considered. We would have, then the following analysis period: `end` > 1-8-2010 AND `end` > 31-8-2010 OR `end` > 18-12-2010 AND `end` > 4-1-2011. The most usual case is where two `TemporalConditions` are used to define the period in which to take the measures.

Furthermore, a `periodicity` can be defined for this `analysisPeriod`. Thus, the corresponding `PPI` will be measured (starting from the moment/s established by the PeriodCondition) `daily`, `weekly`, `monthly` or `yearly` (depending on the type of periodicity defined). In the case of `weekly` periodicity, the day of the week can be chosen, 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) can be selected. It can also be very useful to indicate 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).

Finally, Tables **??** show a summary of the object properties of PPINOT ontology, a brief description and their characteristics and inverses (if they have), that, together with those shown in Table 4.1, will be used in next section. The object properties contained in Table 5.1 and Table 5.2 are depicted in Figure **??**, but the ones contained in Table 5.3, added for the definition of relationships between PPIs (see Subsection 6.3) do not appear in such figure for the sake of readability and simplicity.

ObjectProperty	Description	Characteristics	Inverse
isDefinedOver	a PPI is defined over a MeasureDefinition	Functional	isUsedBy
defines	a MeasureDefinition defines measures (instances)	Inverse Functional	
relatedTo	a PPI is always related to a BP	Functional	-
aggregates	an AggregatedMeasure aggregates IMs	Functional, SubPropertyOf indirectlyAggregates, SubPropertyOf dependsDirectlyOn, disjoint with isCalculated	isAggregatedBy
indirectlyAggregates	a ProcessMeasure indirectlyAggregates IMs	Transitive	isIndirectlyAggregatedBy
hasAggregationFunction	an AggregatedMeasure uses a function to aggregate IMs	Functional	-
isGroupedBy	it allows to group AggregatedMeasures by a DataProperty	Functional	groups
isCalculated	defines the MeasureDefinition used to calculate DerivedMeasures	Disjoint with aggregates	isUsedToCalculate
from	where time starts in a TimeMeasure	Functional	isFromFor
to	where time ends in a TimeMeasure	Functional	isToFor
when	defines the TimeInstantCondition that is count	Functional	isCountIn
meets	defines the condition that must be fulfilled	Functional	isMetBy
appliesTo	indicates the bp element to which the condition is applied	Functional	isUsedInCondition
measuresData	defines the DataObject measured in a DataMeasure	Functional	isMeasuredIn
measuresProperty	defines the DataProeprty measured in a DataMeasure	Functional	-
stateConsidered	allows to specify the state the DataObject must be in when measuring it	Functional	-
hasState	indicates the bpelement state that the condition requires	Functional	-
analysisPeriod	time period where PPI is calcu-	-	-

Table 5.2: Summary of PPINOT Ontology object properties-2 (related to the `AnalysisPeriod`)

ObjectProperty	Description	Characteristics
hasPeriodicity	to define a Periodicity for the PeriodCondition	Functional
onDayOfWeek	to indicate the day of week where measures are taken	Functional
composes	to create composition of periodConditions	-
hasConditionType	to specify the type of Condition	Functional
time	to define the moment in time with which the periodCondition is compared	Functional

Table 5.3: Summary of PPINOT Ontology object properties referring to relationships between PPIs (not depicted in Figure **??**)

ObjectProperty	Description	Characteristics
isCalculatedPositively	if the changes in the MeasureDefinition affect the DerivedMeasure in the same direction	SubPropertyOf isCalculated, SubPropertyOf dependsDirectlyOn, Disjoint with isCalculatedNegatively
isCalculatedNegatively	if the changes in the MeasureDefinition affect the DerivedMeasure in the opposite direction	SubPropertyOf isCalculated, SubPropertyOf dependsInverselyOn, Disjoint with isCalculatedPositively
includes	allows to define the inclusion relationship between MeasureDefinitions	Transitive, SubPropertyOf dependsDirectlyOn
includesPPI	allows to define the inclusion relationship between PPIs	Transitive
dependsOn	allows to define dependencies between MeasureDefinitions	Transitive
dependsOnPPI	allows to define dependencies between PPIs	Transitive
dependsDirectlyOn	the changes in one MeasureDefinition affect in the same direction to the other one	SubPropertyOf dependsOn, Disjoint with dependsInverselyOn
dependsInverselyOn	the changes in one MeasureDefinition affect in the opposite direction to the other one	SubPropertyOf dependsOn, Disjoint with dependsDirectlyOn

Chapter 6

Automated Design-Time Analysis of PPIs

An advantage of formalising the definition of PPIs using OWL-DL is the possibility to automate the analysis of such definitions 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. Such analysis can provide information about the way each PPI is influenced by the different elements of business processes (i.e. *given a particular indicator, what are the business process elements involved in its definition?*), or which PPIs are associated to a concrete element or part of a business process (i.e. the opposite direction, *given a particular business process element, what are the PPIs associated to them?*). In the first case, this information 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 measured before is measured in the new case. In the second case, the information obtained can help in assisting during the evolution of business processes (e.g. if a part of the business process 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). Furthermore, this analysis can help to detect dependencies and potential conflicts amongst them.

6.1 PPIs-BPElements Interaction

It is possible to obtain information about the way PPIs and business process elements influence each other. Two directions are possible:

6.1.1 Given a PPI P, Which are the process model's elements involved?

The elements involved in a PPI P are the same as those involved in the `MeasureDefinition` that defines the PPI. Therefore, let M be the measure definition used by PPI P ($M \in \exists isUsedBy.\{P\}$) and $InvolvedBPElement_M$ the elements involved in M. Then:

$$InvolvedBPElement_{PPIP} \equiv InvolvedBPElement_M$$

where $InvolvedBPElement_{PPIP}$ are the elements of the BP model involved in the PPI.

Hence, to answer the question raised above, our objective is to obtain the elements involved in or that influence a MeasureDefinition (that is, the class $InvolvedBPFlowElement_M$), obtaining, thus, those elements involved in PPI P. These elements may vary depending on the kind of such `MeasureDefinition` (i.e.: `TimeMeasure`, `CountMeasure`, `ConditionMeasure`,`DataMeasure`,`Aggre` or `DerivedMeasure`). In the following we detail each of the possible cases.

BaseMeasure In this case we have to take into account the four possible sorts of baseMeasure.

TimeMeasure In a `TimeMeasure`, every BPElement between the element where time starts to be counted and the element where it ends will be involved (except `DataObjects`, since they do not consume time); we also have to include the start(`from`) and the end (`to`) (except if the start(`from`) is an `ActivityEnd` or a `PoolEnd`,since in this case the element does not influence in the time measured, or if the end (`to`) is not an `ActivityStart` or a `PoolStart`). It is noteworthy that whenever a gateway is in the middle of this way, all the different paths have to be taken into account. Therefore for a `TimeMeasure` M, the elements involved are:

$$\begin{aligned} InvolvedBPElement_M \equiv & IsFromNotEnd_M \\ & \sqcup\ IsInPath_M \sqcup IsToNotStart_M \end{aligned}$$

I$sFromNotEnd_M$ defines the element in which the time starts to be counted, i.e., the `from`, if the `TimeInstantCondition` is not an `ActivityEnd` or a `PoolEnd`.

$$\begin{aligned} IsFromNotEnd_M \equiv \exists isUsedInCondition.(\exists isFromFor.\{M\} \\ \sqcap\ \neg ActivityEnd \\ \sqcap\ \neg PoolEnd) \end{aligned}$$

$IsToNotStart_M$ is similar to the previous case, but in this case, is the element where time ends (i.e., the `to`), if the `TimeInstantCondition` is not an `ActivityStart`, a `PoolSart` or an `eventTrigger`

$$\begin{aligned} IsToNotStart_M \equiv \exists isUsedInCondition.(\exists isToFor.\{M\} \\ \sqcap\ \neg ActivityStart \\ \sqcap\ \neg PoolStart \\ \sqcap\ \neg EventTrigger) \end{aligned}$$

Finally, $IsInPath_M$ corresponds to every element that lies between the start (from) and the end (to), i.e., that is in the path. Depending on the type of `TimeMeasure`, this class is defined in different ways.

If it is a `LinearTimeMeasure`, the first occurrence of `from` and the last occurrence of `to` are taken into account. If no type of `TimeMeasure` is specified, it is assumed by default that it is a `LinearTimeMeasure`. In this case, $IsInPath_M$ is defined as follows:

$$IsInPath_M \equiv \exists precedes.(IsTo_M) \sqcap \exists succeeds.(IsFrom_M)$$

being $sFrom_M$ the element in which the time starts to be counted(`from`):

$$IsFrom_M \equiv \exists isUsedInCondition.(\exists isFromFor.\{M\})$$

and $IsTo_M$ the element where time ends (`to`):

$$IsTo_M \equiv \exists isUsedInCondition.(\exists isToFor.\{M\})$$

If it is an `CyclicTimeMeasure`, the first occurrence of `from` and the first occurrence of `to` are taken into account; and $IsInPath_M$ is defined as follows:

$$IsInPath_M \equiv \exists hasInstance.(IsInPathInstance_M)$$

With isInPathInstance we define the elements whose instance is between the first occurrence (instance) of $IsFrom_M$ and the first occurrence (instance) of $IsTo_M$

$$\begin{aligned} IsInPathInstance_M \equiv & \exists instancePrecedes.(\exists instanceOf.(FirstInstanceIsTo_M)) \\ & \sqcap \exists instanceSucceeds.(\exists instanceOf.(FirstInstanceIsFrom_M)) \end{aligned}$$

being $FirstInstanceIsTo_M$ the first instance of IsTo_M:

$$\begin{aligned} FirstInstanceIsTo_M \equiv & \exists instanceOf.(IsTo_M) \\ & \sqcap \neg(\exists instancePrecedes.(\exists instanceOf.(IsTo_M))) \end{aligned}$$

and $FirstInstanceIsFrom_M$ the first instance of $IsFrom_M$:

$$\begin{aligned} FirstInstanceIsFrom_M \equiv & \exists instanceOf.(IsFrom_M) \\ & \sqcap \neg(\exists instancePrecedes.(\exists instanceOf.(IsFrom_M))) \end{aligned}$$

Let's take as an example of `TimeMeasure` the *ProcessDuration* depicted in Figure 6.1, that measures the duration of the RFC management process. In this case only the `LinearTimeMeasure` makes sense. The elements involved in this `TimeMeasure` are: the element where time starts to be counted ($IsFromNotEnd_M$), that is the event *ReceiveRFC*, the element where time ends ($isToNotStart_M$), that corresponds to the event *ReportRFcApproved* and the

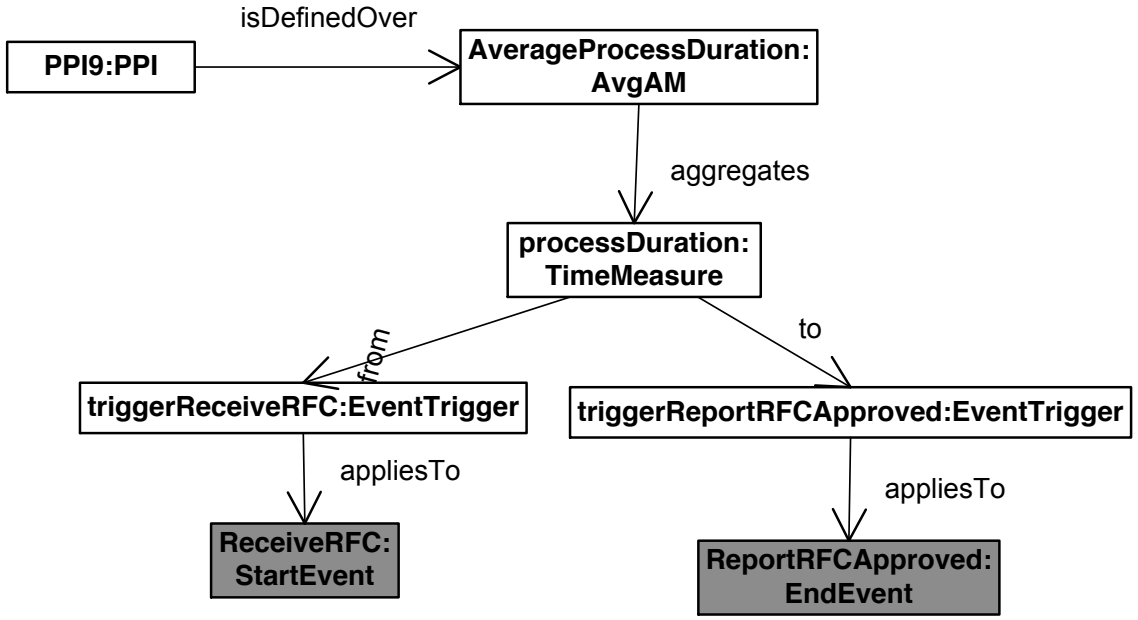

Figure 6.1: Definition of PPI9 "Average Lifetime of a RFC" (i.e.,"Average process duration")

set of elements belonging to all possible paths ($IsInPath_M$), that, in this case, taking into account the existing loop after the activity *Analyse in Committee* includes: the exclusive gateway *merge* , the activity *Analyse RFC*, the exclusive gateway *RFCAnalysisBasedDecision*, the activity *Elevate decision to committee*, the activity *Analyse in committee* and the activity *Approve RFC*. Let us illustrate the way of proceeding following the process previously described:

$$\begin{aligned} InvolvedBPElement_{ProcessDuration} \equiv & IsFrom_{ProcessDuration} \\ & \sqcup\ IsInPath_{ProcessDuration} \\ & \sqcup\ IsTo_{ProcessDuration} \\ & \equiv \{ReceiveRFC, Merge, AnalyseRFC, \\ & RFCAnalysisBasedDecision, \\ & ElevateDecisionToCommittee, \\ & AnalyseInCommittee, ApproveRFC, \\ & ReportRFCApproved\} \end{aligned}$$

As an example of `CyclicTimeMeasure` we have the *AvgAnalyseInCommitteeDuration* depicted in Figure 6.2, that measures the average duration of the activity *Analyse in committee* in a process instance. The elements involved in this `TimeMeasure` are: the element where time starts to be counted ($IsFromNotEnd_M$), that is the activity *Analyse in committee*, the element where time ends ($isToNotStart_M$), that coincides with the previous

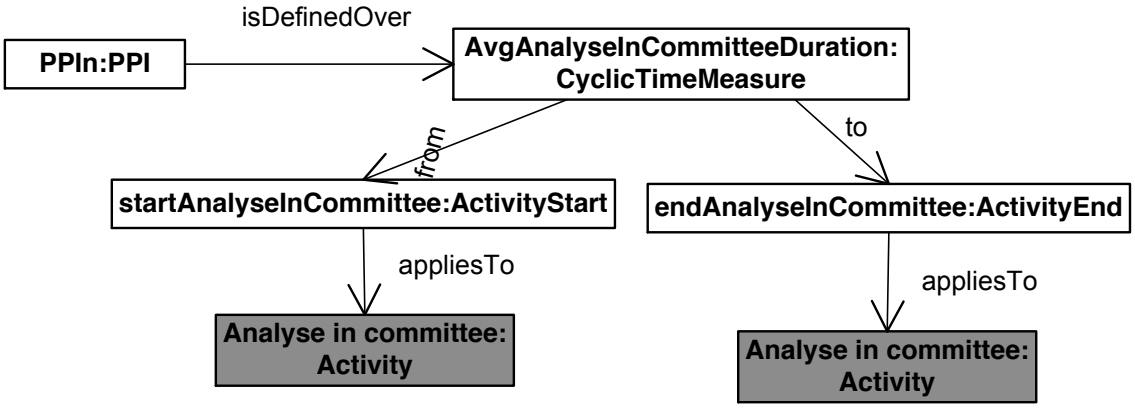

Figure 6.2: Definition of PPIn "Average duration of the analysis in committee" (i.e.,"Average analyse in committee duration")

one (activity *Analyse in committee*, and the set of elements belonging to all possible paths ($IsInPath_M$), that, in this case is empty.

CountMeasure In this case, the element whose `TimeInstantCondition` is checked is involved. But there may be other elements that influence the MeasureDefinition value: if this element is located in one of the paths that follows a decision gateway, depending on the decision (based on data or events), such path will be taken or not. For instance, in the process of our case study, the number of times the activity *Approve RFC* is executed depends on the path taken after the data-based exclusive gateway *RFCAnalysisBasedDecision*. Therefore, the element that makes to take one path or another (such as the preceding activity *Analyse RFC* in our example, or a following event in the case of an event-based exclusive gateway) should be included. Thus, we include in the elements involved the corresponding gateway and the user will have to do a post-processing to identify the element that made the decision to take a concrete path. Therefore, for a `CountMeasure` M, the elements involved are:

$$InvolvedBPElement_M \equiv \exists isUsedInCondition.(\exists isCountIn.\{M\}) \sqcup InvolvedXorGateway_M$$

InvolvedXorGateway corresponds to every gateway that `precedes` the element whose `TimeInstantCondition` is met. A limitation in our proposal is that 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 spliting gateway with its corresponding merge).

$$InvolvedXorGateway_M \equiv XorGateway \sqcap \exists precedes.(\exists isUsedInCondition.(\exists isCountIn.\{M\}))$$

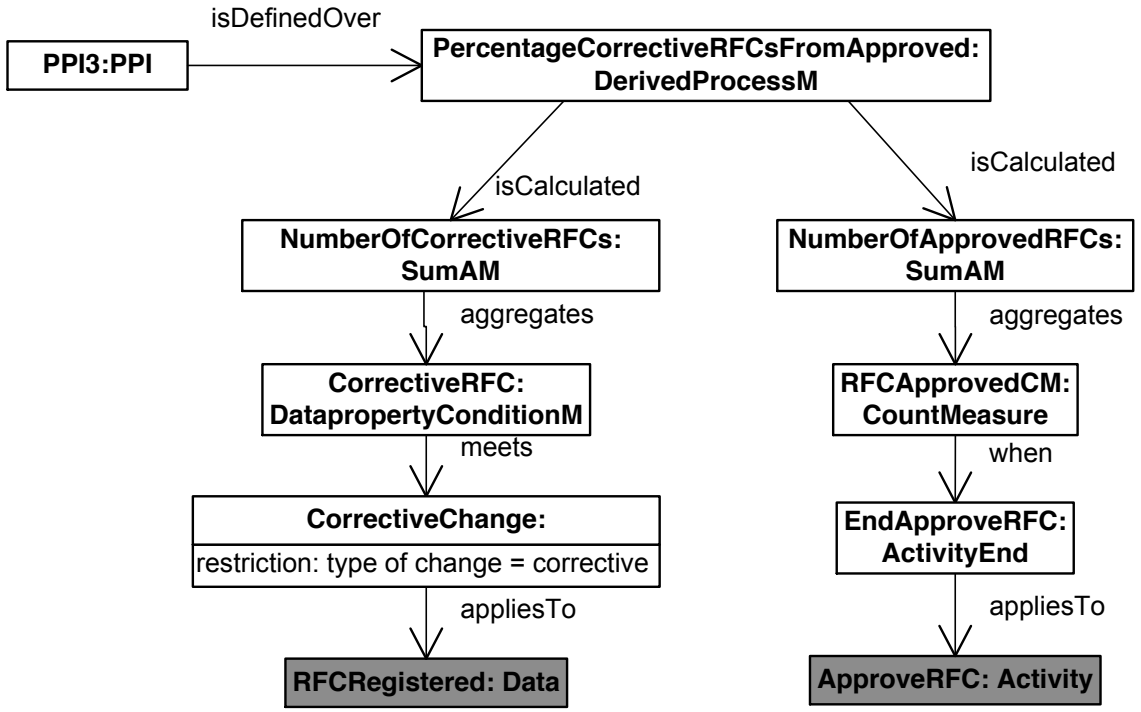

Figure 6.3: Definition of PPI3 "Number of RFCs per project"

Let's take the Figure 6.3 and let's focus on the `CountMeasure` *RFCApprovedCM*. In this example, the elements involved in such `CountMeasure` are the activity *Approve RFC*, the exclusive gateway *Merge* and the exclusive gateway *RFCAnalysisBasedDecision* (corresponding both to $InvolvedXorGateway_M$). Note that actually, the exclusive gateway *Merge* does not really influence the measure. This is a commented limitation of our proposal.

ConditionMeasure In this case, the element whose `ProcessInstanceCondition` is checked is involved. Furthermore, if this element is a `DataObject` (`DataPropertyConditionMeasure` or `DataStateConditionMeasure`), then the activities that could have modified that `DataObject` (i.e., those that have an OWL object property `dataOutputAssociation` to that `DataObject`, for instance, for the `DataObject` *RFCApproved*, the activity *Approve RFC*) also have to be included.

$$InvolvedBPElement_M \equiv \exists isUsedInCondition.(\exists isMetBy.\{M\}) \sqcup WriterActivity_M$$

$WriterActivity_M$ correponds to every activity that can modify, i.e., write the `DataObject` whose `ProcessInstanceCondition` is met in the `ConditionMeasure`.

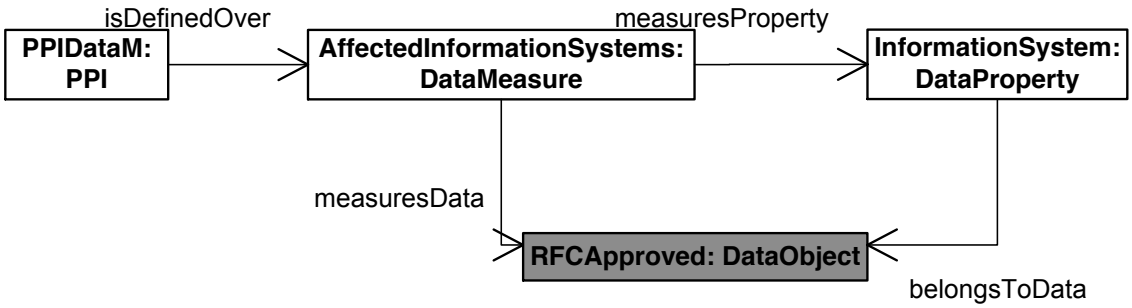

Figure 6.4: Definition of PPIDataM "Affected information systems"

$$WriterActivity_M \equiv \exists DataOutputAssociation.(DataObject \sqcap \exists isUsedInCondition.(\exists isMetBy.\{M\}))$$

To give an example of this case, we take again Figure 6.3, and focus on the `DataPropertyConditionMeas` *CorrectiveRFC*, that measures the number of `DataObject` *RFCs* that have the property *type of change* with the value *corrective*. The only element involved is the `DataObject` *RFCRegistered* (since there not exists any activity that writes on this `DataObject`).

DataMeasure In this case, the elements that influence the `MeasureDefinition` are the `DataObject` itself and, as in the previous case, the activities that could have modified that `DataObject`.

$$InvolvedBPElement_M \equiv \exists isMeasuredIn.\{M\} \sqcup WriterActivity_M$$

In this case, $WriterActivity_M$ can be defined as:

$$WriterActivity_M \equiv Activity \sqcap \exists DataOutputAssociation.(DataObject \sqcap \exists isMeasuredIn.\{M\})$$

In the `DataMeasure` represented in Figure 6.4, called AffectedInformationSytems, the number of information systems affected by the changes requested in the approved RFCs is measured. In this case, the elements involved in such measure are the `DataObject` itself *RFCApproved* and the witerActivityForAffectedInformationSytems, that would correspond to the activity *ApproveRFC*.

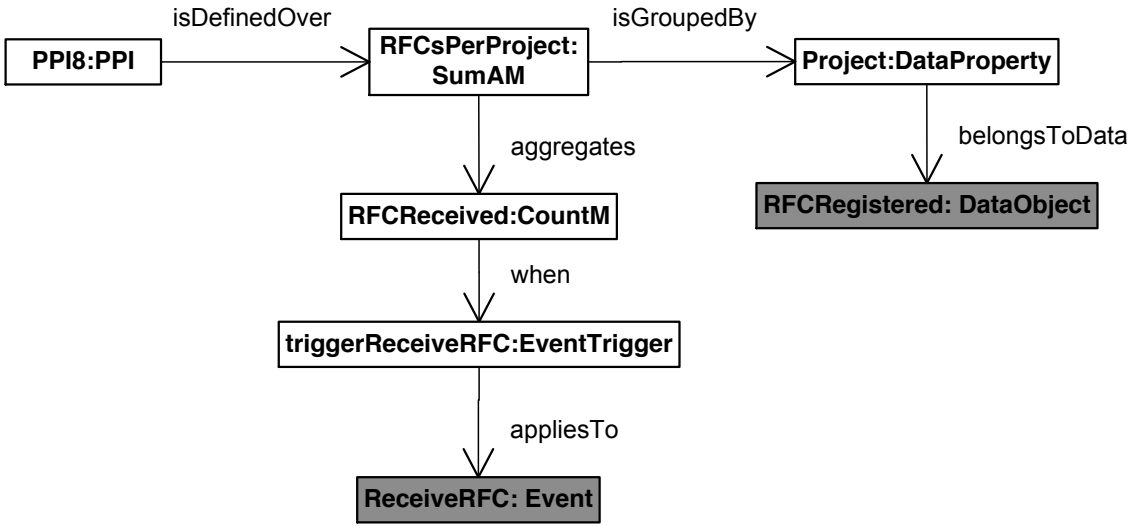

Figure 6.5: Definition of PPI8 "Number of RFCs per project"

AggregatedMeasure When the `MeasureDefinition` is an `AggregatedMeasure`, the elements involved will be those involved in the `InstanceMeasure` that such `AggregatedMeasure` `aggregates`. Therefore, it is necessary to fall back on the `InstanceMeasure` it `aggregates` and check which are the elements that influence it. Furthermore, if this `AggregatedMeasure` `isGroupedBy` certain `DataProperty`, this element also influences the `MeasureDefinition`. Thus, let IM be the measure definition aggregated by `AggregatedMeasure` M ($IM \in \exists isAggregatedBy.\{M\}$). The class containing the elements involved will be the defined below:

$$InvolvedBPElement_M \equiv InvolvedBPElement_{IM} \sqcup \exists DataProperty.(\exists groups.\{M\})$$

To obtain the class $InvolvedBPElement_{IM}$, the process explained above for each of the possible cases will be followed.

In Figure 6.5, PPI8 is defined over an `AggregatedMeasure` (*RFCsPerProject*). To obtain the elements involved in *RFCsPerProject*, we have to focus on the `BaseMeasure` it `aggregates`, in this case, the `CountMeasure` *RFCReceived*. Following the process described above, we find out that the only element involved in such `CountMeasure` is the event *ReceiveRFC*. Furthermore, this `AggregatedMeasure isGroupedBy` the `DataProperty` *Project* of the `DataObject` *RFCRegistered* . The elements involved are thus, the event *ReceiveRFC* and the `DataObject` *RFCRegistered*.

DerivedMeasure Finally, for the `DerivedMeasure`, 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 M ($\{D_1, \ldots, D_n\} \in \exists isUsedToCalculate.\{M\}$),

Then, the elements involved in the `DerivedMeasure` are the union of the elements involved in these measures:

$$InvolvedBPElement_M \equiv InvolvedBPElement_{D_1} \sqcup \cdots \sqcup InvolvedBPFlowElement_{D_n}$$

As in the previous case, to obtain the classes $InvolvedBPElement_{D_1}$, ..., $InvolvedBPElement_{D_n}$, the process explained above for each of the possible cases will be followed.

Figure 6.3 depicts a `DerivedProcessMeasure` (***PercentageOfCorrectiveRFCsFromApproved***), that calculates the percentage of RFCs with correctives changes from all the RFCs approved. This `DerivedMeasure isCalculatedUsing` two `AggregatedMeasures`: ***NumberOfCorrectiveRFCs***, which`aggregates` a `DataPropertyConditionMeasure`, described in paragraph "ConditionMeasure" and ***NumberOfAppovedRFCs***, which `aggregates` a `countMeasure`, described in paragraph "CountMeasure". The elements involved in this `DerivedInstanceMeasure` are the union of the elements involved in both of them: the activity ***ApproveRFC*** , the `DataObject` ***RFCRegistered***, and the exclusive gateways ***Merge*** and ***RFCAnalysisBasedDecision***.

6.1.2 Given a BPElement E, Which are the PPIs associated or applied to them?

Once we have obtained all the elements involved in or that influence a `MeasureDefinition` M, and hence, the elements involved in or that can influence the value of the corresponding PPI, it is a straightforward and direct process to answer the question in the opposite direction, i.e. the question presented above. In fact, this information has already being extracted: after having performed the aforementioned process for every PPI defined for a business process this `BPElement` E will belong to several OWL classes of type MInvolvedBPElement. The next step is to isolate the name of the `MeasureDefinition` M and to obtain the PPI defined over M.

Let $M_1, ..., M_n$ be the first, ..., nth `MeasureDefinition` in which E is involved ($E \in M1InvolvedBPElement \sqcap ... \sqcap E \in MnInvolvedBPElement$).

Then, the set of PPIs associated or applied to E (EAssociatedPPI) can be defined as:

$$AssociatedPPI_E \equiv \exists isDefinedOver.\{M_1\} \sqcup \cdots \sqcup \exists isDefinedOver.\{M_n\}$$

To illustrate this case, let's see how we proceed to obtain the PPIs related to the activity ***Analyse RFC***.

We have that $AnalyseRFC \in AverageTimeAnalysingRFCInvolvedBPElement \sqcap AnalyseRFC \in RFCsUnderAnalysisInvolvedBPElement \sqcap AnalyseRFC \in AverageRFCLifetimeInv$

Then, the set of PPIs associated or applied to ***Analyse RFC*** is:

$$\begin{aligned} AssociatedPPI_{AnalyseRFC} \equiv & \exists isDefinedOver.\{AverageTimeAnalysingRFC\} \\ & \sqcup \exists isDefinedOver.\{RFCsUnderAnalysis\} \\ & \sqcup \exists isDefinedOver.\{AverageRFCLifetime\} \\ \equiv & \{PPI5, PPI6, PPI9\} \end{aligned}$$

6.2 PPI Internal Information

Regarding the information implicitly contained in PPIs definition, there are a plethora of scenarios and kind of queries to be performed in order to obtain such information. Some examples are: *which PPIs are defined on a concrete* `AnalysisPeriod`*?*, this information is useful if, fro instance, the law changes and monthly reports are required instead of annual reports; *which PPIs measure time (i.e. are defined over* `TimeMeasures` *or over* `AggregatedMeasures` *that* `aggregates TimeMeasures`*)? which ones measures conditions (over* `ConditionMeasures` *or* `AggregatedMeasures` *that* `aggregates ConditionMeasures`*? and so on*, this information is useful, fo example, in order to organise information when creating dashboards.

Let's show the way to proceed to obtain the aforementioned information.

In the first case, we answer the question *which PPIs are defined on* `AnalysisPeriod` *A?* as follows:

$$PPIsDefinedOnAnalysisPeriodA \equiv \exists analysisPeriod.\{A\}$$

In the second case, we answer the questions *which PPIs measure time? which ones counts? which ones measures conditions? and which ones measures data?* as follows:

$$\begin{aligned} TimeMPPI \equiv & \exists isDefinedOver.(TimeMeasure \\ & \sqcup \exists aggregates.(TimeMeasure)) \\ CountMPPI \equiv & \exists isDefinedOver.(CountMeasure \\ & \sqcup \exists aggregates.(CountMeasure)) \\ ConditionMPPI \equiv & \exists isDefinedOver.(ConditionMeasure \\ & \sqcup \exists aggregates.(ConditionMeasure)) \\ DataMPPI \equiv & \exists isDefinedOver.(DataMeasure \\ & \sqcup \exists aggregates.(DataMeasure)) \end{aligned}$$

6.3 PPI-PPI Relationships

6.3.1 Inclusion Relationship

PPI1 includes PPI2 if both measure time, and the fragment of the process measured by PPI2 (pf2) is contained in the fragment of the process measured by PPI1 (pf1)(i.e., the `from` for PPI2 (from2) succeeds or coincides with the `from` for PPI1 (from1), from2$\geq$ from1, and the `to` for PPI2 (to2) precedes or coincides with the `to` for PPI1 (to1), to2$\leq$ to1) ; this can happen in three cases:

1. if both of them are defined over `TimeMeasures` and pf2 is contained in pf1.
2. if PPI1 is defined over an `AggregatedMeasure` that aggregates a `TimeMeasure` and PPI2 is defined over a `TimeMeasure` and pf2 is contained in pf1.
3. if PPI1 and PPI2 are defined over `AggregatedMeasures` that aggregate a `TimeMeasure` and pf2 is contained in pf1.

In the following we define the set of inference rules[1] that allow to obtain these inclusion relationships:

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

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

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

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

6.3.2 Dependencies between MeasureDefinitions

The PPI ontology defines two types of relationships between `MeasureDefinitions`: `aggregates` and `isCalculated`. The former means that the measures defined by the `MeasureDefinition` are calculated as an aggregation of the same type of measures (i.e., defined by the same `MeasureDefinition`) from different process instances. The latter means that the measures are calculated as a mathematical function of either several different process measures, or several different measures from the same process instance. This last property can be further refined into two subproperties:

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

`isCalculatedPositively` and `isCalculatedNegatively`, depending on whether the changes in one measure affect the other measure either in the same direction (positive) or the opposite direction (negative). For instance, if $PercentRequestsApproved = \frac{RequestsApproved}{RequestRegistered} \times 100$, then:

$$isCalculatedPositively(PercentRequestsApproved, RequestsApproved)$$
$$isCalculatedNegatively(PercentRequestsApproved, RequestRegistered)$$

These relationships together with the inclusion relationship (`includes`) described in the previous subsection define a dependency between `MeasureDefinitions` in the sense that changes in the measures defined by one `MeasureDefinition` have an influence on the measures defined by the other `MeasureDefinition`. Therefore, a new property called `dependsOn` can be added to the ontology. Furthermore, this relationship can be refined again into other two different relationships in the same way as `isCalculated`, i.e., depending on whether the changes in one measure affect the other measure either in the same direction (`dependsDirectlyOn`) or the opposite direction (`dependsInverselyOn`).

Therefore, if a `MeasureDefinition` $m1$ aggregates a `MeasureDefinition` $m2$ or if a `MeasureDefinition` $m1$ includes a `MeasureDefinition` $m2$, then $m1$ depends directly on $m2$. Similarly, if a `MeasureDefinition` $m1$ is calculated on another `MeasureDefinition` $m2$, then $m1$ depends either directly or inversely on $m2$ depending on whether $m1$ is calculated positively or negatively from $m2$ respectively. These statements can be expressed defining `aggregates`, `includes` and `isCalculatedPositively` as subproperties of `dependsDirectlyOn` and `isCalculatedNegatively` as subproperty of `dependsInverselyOn`.

Furthermore, some inference rules can be defined to propagate the dependencies throughout all `MeasureDefinitions`. These rules infer the dependencies between two `MeasureDefinitions` ($m1$ and $m2$) by means of the dependencies they have with another `MeasureDefinition` y as follows:

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

Once the dependencies between `MeasureDefinitions` have been obtained, it is a direct process to obtain the dependencies between the PPIs defined by those `MeasureDefinitions`. It is enough to define new properties for the dependency between PPIs (dependsDirectlyOnPPI, dependsInverselyOnPPI as subproperties of dependsOnPPI) and inference rules as the following ones:

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

Since most modern OWL-DL reasoners allow the use of SWRL rules together with the ontology as a means to extend the expressiveness of OWL DL, the rules defined above can be used to infer all of the dependencies amongst MeasureDefinitions (and hence amongst PPIs), and, then, all these inferred knowledge can be used to answer queries regarding the PPIs defined in one organisation. For instance, we may identify potentially conflicting PPIs if they are defined over two `MeasureDefinitions` $m1$ and $m2$ and there is a third `MeasureDefinition` $m3$ such as $m1$ depends directly on $m3$ and $m2$ depends inversely on $m3$ or *viceversa*.

Chapter 7

Implementation

We have developed a prototype whose component model is shown in Figure 7.1. This tool analyses PPIs defined over BPMN diagrams and returns the information required (as explained in Section 6, BP elements-PPIs relationship, PPI internal information or dependencies among PPIs). This tool proceeds as follows: A business process diagram (BPD) is defined in *Oryx* [Decker et al., 2008]. Then, the set of PPIs is defined over such BPD using the *PPINOT Oryx Plugin* (developed by us). 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 Section 5 (and taking into account the considerations described in Section 4). This is done by the *XML2OWL Mapper*. This OWL file with the BPD + PPIs information 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 implied in a PPI definition, PPIs associated to a given BP element, PPI internal information or relationships among PPIs) is automatically generated. This tool can be tested following the instructions described at `http://www.isa.us.es/PPINOT`.

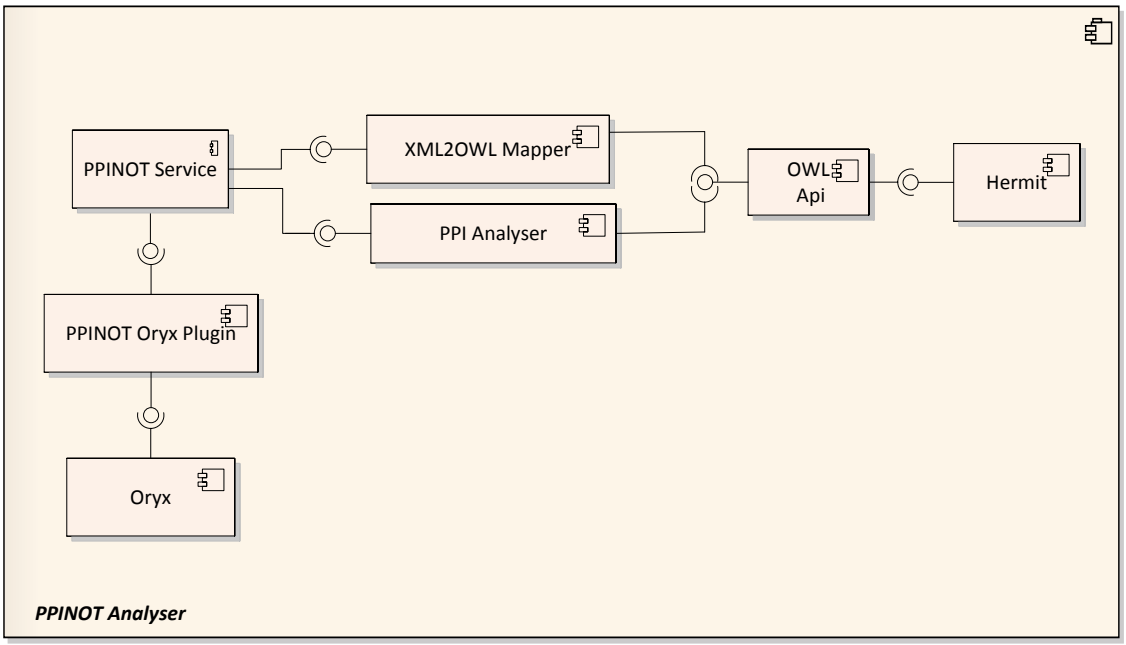

Figure 7.1: PPINOT Component model

Chapter 8

Related Work

8.1 Process Performance Measurement

Performance measurement is a current research field in management science that have gained interest in both academia and business [Popova and Sharpanskykh, 2010]. Many works have been done in the identification and classification of key performance indicators for any company [Kaplan and Norton, 1992] and those relevant for specific domains such as logistics, production, supply chains, etc. (e.g. Brewer and Speh [2000]; Chan [2003]; Krauth et al. [2005]; Vaidyanathan [2005])

Process management is becoming a part of the language and actions of many organisations. Many companies are taking this process-oriented perspective in their business as a way of identifying which steps really create the 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 [Alexander Grosskopf and Weske, 2009]. Hence, many authors recognize the importance of process orientation of performance management systems [M.J. Benner, 2003]. Actually, the Academia of Business Process Management Professionals present in [of Business Process Management Professionals (ABPMP), 2009] the following definition for process performance measurement: "the formal planned monitoring of process execution and the tracing of results to determine the effectiveness and efficiency of the process".

There already exists a number of proposals to evaluate the performance of business processes defined in the literature and, in some cases, implemented in products.

Popova et al. present in [Popova and Sharpanskykh, 2010] a framework for modeling performance indicators within a general organisation modeling framework. They define indicators by assigning values to a set of attributes, but they do not point out the way these indicators are calculated. They do it, instead, in [Popova and Sharpanskykh, 2009], where they present formal techniques for analysis of executions of organizational scenarios. They also define, in this work, relations between PPIs and the processes, and relationships between PPIs (causality, correlation and aggregation), introduced briefly in [Popova and Sharpanskykh, 2010] and explained in detail in [Popova and Sharpanskykh, 2009]. According to the definition of the analysis period, they define temporal properties over PPIs (called PI expressions) in [Popova and Sharpanskykh, 2008]. They do not consider derived measures nor queries to obtain the so-called in this work PPI-BPElements

Interaction in their works.

In[Mayerl et al., 2007] Mayerl et al. discuss how to derive 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 metric dependencies, based on concepts of the CIM metrics model [(DMTF), 2003] and the QoS UML profile described in [(OMG), 2006]. They also introduce a mathematical formalism in order to describe dependency functions and the so-called metric characteristics or metrics calculable based on other metric values. Finally they cover the mapping of these models to a monitoring architecture that contains functions to instrument and collect metrics, functions to aggregate and compare metrics with agreed service levels and functions to report SLA compliance and violations. However, they do not delve into the definition of measures, they only set the semantics of some elements to consider when defining measures.

Castellanos et al.'s approach [Castellanos et al., 2005] is implemented in the IBOM platform, that allows, among other things, to define business measures and perform intelligent analysis on them to understand causes of undesired values and predict future values. The user can define business measures (through a GUI) to measure characteristics of process instances, processes, resources or of the overall business operations. Specifically, they characterize metrics through four attributes: name (unique), target entity (objet to be measured), data type (numeric, boolean, taxonomy or SLA) and desirable metric values. For the computation logic definition, templates are used. These templates map data and metadata about process executions into numeric and boolean measures. This approach is not focused on business processes but on the whole organisation. Anyway, during the definition of metrics, as far as we can deduce from the paper, they do not take into account some aspects we do, as the analysis period, the unit of measure, the dimension to be measured. It is not possible to know which is the set of measures than ca be defined with this approach.

Momm et al.'s approach [Momm et al., 2007] consists of a top-down approach for developing an uniform IT support based on 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. However, the metamodel for the specification of performance indicators does not consider those related to data or events (PPI6 from our example can not be defined according to this metamodel); and it lacks some properties when defining PPIs like the analysis period or the function to calculate derived measures. Moreover, the absence of a formal foundation for this PPI definition does not allow an automated analysis of them

Another work which is close to ours is the one presented by Wetzstein et al. in [Wetzstein et al., 2008]. This paper introduces a framework for BAM as part of the semantic business process management. The authors describe a KPI ontology using WSML to specify KPIs over semantic business processes. However, our ontology improves this one, since they do not take into account indicators related to data (they can not define PPI6).

The integrated methodology GRAI/GIM Chen et al. [1997]; Doumeingts et al. [1998] ex-

plicitly models performance indicators. They establish three parameters or attributes to define performance indicators: name, value domain or dimension and procedure to calculate the value. However this definition is not process-aware and is only made informally and without taking into account the relationships among the performance indicators and between them and the BP elements.

ARIS [Davis and Brabänder, 2007] models key performance indicators and allow for using the Balance Scorecard approach for modelling cause-and-effect relationships and assign KPIs to the strategic objective. However without formal foundations of the modelling approach the possibilities for analysis are limited.

Pedrinaci et al. [Pedrinaci et al., 2008] describe a Semantic Business Process Monitoring Tool called SENTINEL. This tool can support automated reasoning, though the authors point out that one aspect to be improved is the analysis engines in order to support deviations. In this paper, they also present a metric ontology to allow the definition and computation of metrics, which take into account many of the aspects we do , for instance the concept of population filter, which is somehow similar to our "analysis period" but not only limited to time. However, it is not clear how PPIs can be analysed and queried based on this concept nor it is clear whether it allows an explicit relation between PPIs and the elements of a business process. Furthermore, they deal with runtime analysis, but not design-time.

Regarding the use of ontologies in business process management, apart from the above mentioned semantic web-based approach [Pedrinaci et al., 2008], there are also several approaches ranging from ontologies supporting the definition of organisational structures [Abramowicz et al., 2008], business processes [Abramowicz et al., 2007] or even business goals to ontologies for capturing execution logs and supporting business process analysis [Pedrinaci et al., 2008]. There are also some other ontologies dedicated to define measures in different areas, like [García et al., 2006] for software measurement.

InTable 8.1 we establish an explicit comparison between the previously commented approaches for the process performance measurement and PPINOT (our proposal). We highlight those benefits we assigned to our proposal in Section 1: Seamless BP relationship (feature 1), expressiveness (feature 2- 8) and automated analysis, in the table represented through "Aut An" due to space constraints (features 9 and 10). 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 addresses this feature does not take into account the possibility of grouping by certain property (`isGroupedBy`).

8.2 Business Process Analysis

In the area of analysis of business processes, the interest in querying techniques for BPs has arisen in recent years. “Some of existing approaches for querying BPs Awad [2007]; Beeri et al. [2008]; Eshuis and Grefen [2007]; Francescomarino and Tonella [2008] focused on querying the definitions of BP models. They provide business analysts with a visual interface to search for certain patterns and analyse and reuse BPs. These query languages are based on graph matching techniques. BP-QL [Beeri et al., 2008] and [Eshuis and Grefen, 2007] are designed to query business processes expressed in BPEL. BPMN-Q Awad [2007]; Sakr and Awad [2010] and VQL

Proposal	PPI-BP	Expressiveness							Aut An		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
Popova and Sharpanskykh [2010]	✓	✓	N/A	N/A	N/A	✓*		✓		∼	✓
Mayerl et al. [2007]	∼	✓	✓	N/A		✓*	✓				∼
Castellanos et al. [2005]	∼	✓	✓	∼	N/A	✓*	∼				
Momm et al. [2007]	✓	✓	✓			✓*					
Wetzstein et al. [2008]	✓	✓	✓	✓		✓*		∼			∼
Chen et al. [1997][1]		N/A	N/A	N/A		✓*	N/A				
[Davis and Brabänder, 2007]	∼	✓	✓	N/A		N/A	N/A	∼			
Pedrinaci et al. [2008]	N/A	✓	✓	N/A		✓*	✓	✓	∼	∼	∼
PPINOT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

(1) Relationship between PPIs and BP models
(2) Time measures
(3) Count measures
(4) Condition measures
(5) Data measures
(6) Aggregated measures
(7) Derived measures
(8) Definition of analysis period
(9) PPI-BP Interaction
(10) PPI internal information
(11) Relationships among PPIs

Table 8.1: Comparison of the analysed approaches and our proposal

[Francescomarino and Tonella, 2008] are oriented to query generic process modelling concepts «« [Beheshti et al., 2011].

Another aspect that is frequently engaged in querying BPs is to query running instances of business processes Beeri et al. [2007]; Momotko and Subieta [2004]; Weidlich et al. [2011]. These approaches can be used to monitor the status of running processes and trace the progress of execution. Beeri et al. [Beeri et al., 2007] propose BP-MON, a query language for monitoring business processes defined in BPEL. To monitor process instances this query language use execution pattern, that are matched to execution traces.

In [Momotko and Subieta, 2004], Momotko et al present BPQL, a business process query language that extends the object-oriented query language SBQL (Stack-Based QUery Language), and allows to express some requirements on the process definition that depends on the process execution data. They integrate BPQ with XPDL.

Finally, in [Weidlich et al., 2011], Weidlich et al. describe a method to monitor control flow deviation during the process execution. Based on process models, they extract behavioural profiles and generate complex event queries, that are executed over process events. Detected deviations (after applying filters and aggregation to avoid overload of information) are presented to the analyst.

In our approach, we define a mechanism to query the BP-model together with the PPI model, allowing thus to perform a design-time analysis, not only on the business processes themselves, but also on the PPIs defined over such processes and the existing relationship between these PPIs and the elements of the corresponding business processes. This is, to the best of our knowledge, a novel approach.

Chapter 9

Conclusions and Future Work

In this paper we demonstrate that it is possible to extract important information (useful during BPM/PPI lifecycle) from PPI models in an automatic way. This is done by presenting PPINOT ontology, an OWL-DL ontology for the definition of Process Performance Indicators that allows the representation of the relationships between these PPIs and the business process elements. Furthermore, 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). Finally, this formal foundation in the definition of PPIs enables their analysis at design-time in a way that is amenable to automated reasoning, allowing thus to infer interdependencies between PPIs and BP elements, dependencies between PPIs and internal information of the definition of PPIs; helping thus to, for instance, assist during the evolution of business processes or to predict conflicts between PPIs and future behaviour. It is noteworthy that we have applied this approach to several real-world case studies (and some others are planned) in order to evaluate the proposal and to get feedback to improve it.

A number of directions for future work are considered. The extension of the PPINOT ontology in order to support the specification of PPIs related to (human) resources and of the analysis mechanism will be necessary in order to extract information related to the resources assigned to the BP activities (workload for instance). A graphical notation to depict PPIs and its tool support in PPINOT tool together with a methodological guide to assist the user in the process of defining PPIs are being developed. Another aspect to work in is a further research on the concrete activities related to the PPI lifecycle that must be performed along the BPM lifecycle (identifying which of them must be accomplished in each phase), i.e., a complete integration of both lifecycles (BPM- and PPIM-). In this sense, more investigation is needed in the integration of the tool with the execution aspects of the supporting information systems in the context of performance evaluation (some first steps using the Activiti BPM Platform `http://activiti.org` has been made).

Acknowledgments

The authors would like to thank Ana Belén Sánchez for her help implementing part of PPINOT tool. They also would like to thank the Quality Office of the Information Technology Department

of the Andalusian Health Service for kindly providing us their internal business process models and its PPIs. Finally they would like to thank J. Mendling, A. Sharpanskykh, V. Popova, D. Ruiz, C. Pedrinaci and E. Cardoso for their helpful comments in earlier versions of this report.

Bibliography

ABRAMOWICZ, W., FILIPOWSKA, A., KACZMAREK, M., AND KACZMAREK, T. 2007. Semantically enhanced business process modelling notation. In *Proceedings of SBPM 2007 Semantic Process and Product Lifecycle Management in conjunction with the 3rd European Semantic Web Conference (ESWC 2007)*.

ABRAMOWICZ, W., FILIPOWSKA, A., KACZMAREK, M., PEDRINACI, C., STARZECKA, M., AND A., W. 2008. Organization structure description for the needs of semantic business process management. In *3rd international Workshop on Semantic Business Process Management colocated with 5th European Semantic Web Conference*.

ADELA DEL RÍO-ORTEGA, M. R. AND RUIZ-CORTÉS, A. October, 2010. Defining process performance indicators: An ontological approach. In *Proceedings of the 18th International Conference on Cooperative Information Systems (CoopIS). OTM 2010, Part I*. 555–572.

ALEXANDER GROSSKOPF, G. D. AND WESKE, M. Mar 2009. *The Process. Busisness Process Modeling using BPMN*. Megan-kiffer Press.

AWAD, A. 2007. Bpmn-q: A language to query business processes. In *EMISA*. 115–128.

BAADER, F., CALVANESE, D., MCGUINNESS, D. L., NARDI, D., AND PATEL-SCHNEIDER, P. F., Eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

BEERI, C., EYAL, A., KAMENKOVICH, S., AND MILO, T. 2008. Querying business processes with bp-ql. *Inf. Syst. 33,* 6, 477–507.

BEERI, C., EYAL, A., MILO, T., AND PILBERG, A. 2007. Monitoring business processes with queries. In *VLDB*. 603–614.

BEHESHTI, S.-M.-R., BENATALLAH, B., NEZHAD, H. R. M., AND SAKR, S. 2011. A query language for analyzing business processes execution. In *BPM*. 281–297.

BHATT, M., RAHAYU, W., SONI, S. P., AND CARLO. 2009. 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. <ce:title>Semantic Web challenge 2008</ce:title>.

BREWER, P. AND SPEH, T. 2000. Using the balance scorecard to measure supply chain performance. *Journal of Business Logistics 21*, 75–93.

BROCKMANS, S., VOLZ, R., EBERHART, A., AND LÖFFLER, P. 2004. Visual modeling of OWL DL ontologies using UML. In *ISWC 2004*. Vol. 3298. Springer, 198–213.

CASTELLANOS, M., CASATI, F., SHAN, M.-C., AND DAYAL, U. 2005. ibom: a platform for intelligent business operation management. In *Proceedings. 21st International Conference on Data Engineering, 2005.* Hewlett-Packard Laboratories, 1084– 1095.

CHAN, F. 2003. Performance measurement in a supply chain. *International Journal of Advanced Manufacturing Technology 21*, 534–548.

CHEN, D., VALLESPIR, B., AND DOUMEINGTS, G. 1997. Grai integrated methodology and its mapping onto generic enterprise reference architecture and methology. *Computers in Industry 33*, 387–394.

DAVIS, R. AND BRABÄNDER, E. 2007. Aris design platform. Springer.

DECKER, G., OVERDICK, H., AND WESKE, M. 2008. Oryx - an open modeling platform for the bpm community. In *BPM*. 382–385.

DEL RÍO-ORTEGA, A. AND RESINAS, M. 2009. Towards modelling and tracing key performance indicators in business process. In *Actas del II Taller sobre Procesos de Negocio e Ingeniería de Servicios (PNIS 2009) en el marco de las XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*.

(DMTF), D. M. T. F. 2003. Common information model (cim) metrics model.

DOUMEINGTS, G., VALLESPIR, B., AND CHEN, D. 1998. *Handbook on Architectures of Information Systems*. Springer-Verlag, Chapter Decisional Modelling Using the GRAI Grid, 313–338.

EFQM. 2010. EFQM excellence model 2010.

ESHUIS, R. AND GREFEN, P. W. P. J. 2007. Structural matching of bpel processes. In *ECOWS*. 171–180.

FRANCESCOMARINO, C. D., GHIDINI, C., ROSPOCHER, M., SERAFINI, L., AND TONELLA, P. 2009. Semantically-aided business process modeling. In *International Semantic Web Conference*, A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, Eds. Lecture Notes in Computer Science Series, vol. 5823. Springer, 114–129.

FRANCESCOMARINO, C. D., GHIDINI, C., ROSPOCHER, M., SERAFINI, L., AND TONELLA, P. 2011. A framework for the collaborative specification of semantically annotated business processes. *Journal of Software Maintenance 23,* 4, 261–295.

FRANCESCOMARINO, C. D. AND TONELLA, P. 2008. Crosscutting concern documentation by visual query of business processes. In *Business Process Management Workshops*. 18–31.

GARCÍA, F., BERTOA, M. F., CALERO, C., VALLECILLO, A., RUIZ, F., PIATTINI, M., AND GENERO, M. 2006. Towards a consistent terminology for software measurement. *Information & Software Technology 48,* 8, 631–644.

GENNARI, J. H., MUSEN, M. A., FERGERSON, R. W., GROSSO, W. E., CRUBZY, M., ERIKSSON, H., NOY, N. F., AND TU, S. W. 2002. The evolution of protégé:an environment for knowledge-based systems development. *International Journal of Human-Computer Studies 58*, 89–123.

ISACA. 2009. COBIT v4.1.

KAPLAN, R. S. AND NORTON, D. P. 1992. The balanced scorecard: Measures that drive performance. *Harvard Business Review January-February 1992*, 71–79.

KRAUTH, E., MOONEN, H., POPOVA, V., AND SCHUT, M. C. 2005. Performance measurement and control in logistics service providing. In *ICEIS (2)*. 239–247.

MAYERL, C., HÜNER, K., GASPAR, J.-U., MOMM, C., AND ABECK, S. 2007. Definition of metric dependencies for monitoring the impact of quality of services on quality of processes. In *Second IEEE/IFIP International Workshop on Business-driven IT Management (Munich)*. 1–10.

MESSER, M., PANCHAL, J. H., ALLEN, J. K., AND MISTREE, F. 2011. Model refinement decisions using the process performance indicator. *Journal of Engineering Optimization 43,* 7, 741–762.

M.J. BENNER, M. L. T. 2003. Exploitation, exploration and process management: the productivity dilemma revisited. *Academy and Management Review 28*, 238–256.

MOMM, C., MALEC, R., AND ABECK, S. 2007. Towards a model-driven development of monitored processes. In *Wirtschaftsinformatik (2)*. 319–336.

MOMOTKO, M. AND SUBIETA, K. 2004. Process query language: A way to make workflow processes more flexible. In *Advances in Databases and Information Systems*, A. Benczúr, J. Demetrovics, and G. Gottlob, Eds. Lecture Notes in Computer Science Series, vol. 3255. Springer Berlin / Heidelberg, 306–321.

MOTIK, B., PATEL-SCHNEIDER, P. F., AND GRAU, B. C. 2009. OWL 2 Web Ontology Language Direct Semantics.

NARDI, D. AND BRACHMAN, R. J. 2003. An introduction to description logics. In *Description Logic Handbook*. 1–40.

NEELY, A., GREGORY, M., AND PLATTS, K. 2005. Performance measurement system design: A literature review and research agenda. *International Journal of Operations & Production Management 25*, 1228 – 1263.

NICOLA, A. D., LEZOCHE, M., AND MISSIKOFF, M. 2007. An ontological approach to business process modeling. In *IICAI*. 1794–1813.

OF BUSINESS PROCESS MANAGEMENT PROFESSIONALS (ABPMP), A. 2009. Guide to the business process management common body of knowledge.

OF GOVERNMENT COMMERCE), O. O. 2007. Information technology infrastructure library (ITIL) v3. Collection of books.

(OMG), O. M. G. 2006. Umltm profile for modeling quality of service and fault tolerance characteristics and mechanisms specification.

(OMG), O. M. G. 2009. Business process modeling notation (bpmn) version 1.2.

ORIENTATION, B. P. 2010. Survey on process management: Process performance measurement.

PEDRINACI, C., DOMINGUE, J., AND ALVES DE MEDEIROS, A. K. 2008. A core ontology for business process analysis. In *5th European Semantic Web Conference*.

PEDRINACI, C., LAMBERT, D., WETZSTEIN, B., VAN LESSEN, T., CEKOV, L., AND DIMITROV, M. 2008. Sentinel: a semantic business process monitoring tool. In *OBI*. 1.

PELLET. 2011. (owl) 2 reasoner for java.

PLUGIN, P. O. 2011. Ontology editor for the semantic web.

POPOVA, V. AND SHARPANSKYKH, A. 2008. Formal goal-based modeling of organizations. In *MSVVEIS*. 19–28.

POPOVA, V. AND SHARPANSKYKH, A. 2009. Formal analysis of executions of organizational scenarios based on process-oriented specifications. *Applied Intelligence*.

POPOVA, V. AND SHARPANSKYKH, A. 2010. Modeling organizational performance indicators. *Inf. Syst. 35,* 4, 505–527.

PROTEGE. 2011. An ontology and knowledge base editor.

(RACER). 2011. Renamed abox and concept expression reasoner.

REASONER, H. O. 2011. The new kid on the (owl) block.

SAKR, S. AND AWAD, A. 2010. A framework for querying graph-based business process models. In *WWW*. 1297–1300.

(SAP), V. H. 2007. Process management lifecycle.

(TILBURG), V. A., (UCBL), S. B., (UOC), M. B., (USTUTT), O. D., (UCBL), M. H., VAN DEN HEUVEL (TILBURG), W., (USTUTT), D. K., (TILBURG), B. K., (USTUTT), F. L., (TILBURG), M. M., (UCBL), K. M., (UOC), C. N., (TILBURG), M. P., AND (USTUTT), B. W. 2008. Survey on business process management. Deliverable of S-Cube. http://www.s-cube-network.eu.

VAIDYANATHAN, G. 2005. A framework for evaluating third-party logisics. *ACM 48*, 89–94.

VAN DER AALST, W. M., TER HOFSTEDE, A. H., AND WESKE, M. 2003a. Business process management: A survey. In *Business Process management*. Vol. 2678. Springer, 1–12.

VAN DER AALST, W. M. P. 1996. Three good reasons for using a petri-net-based workflow management system. In *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*. 179–201.

VAN DER AALST, W. M. P., TER HOFSTEDE, A. H. M., AND WESKE, M. 2003b. Business process management: A survey. In *Business Process Management*. 1–12.

WEIDLICH, M., ZIEKOW, H., MENDLING, J., GÜNTHER, O., WESKE, M., AND DESAI, N. 2011. Event-based monitoring of process execution violations. In *BPM*. 182–198.

WESKE, M. 2007. *Business Process Management: Concepts, Languages, Architectures*. Springer.

WESKE, M., VAN DER AALST, W. M. P., AND VERBEEK, H. M. W. E. 2004. Advances in business process management. *Data Knowl. Eng. 50,* 1, 1–8.

WETZSTEIN, B., MA, Z., AND LEYMANN, F. 2008. Towards measuring key performance indicators of semantic business processes. In *BIS*. 227–238.

WODTKE, D. AND WEIKUM, G. 1997. A formal foundation for distributed workflow execution based on state charts. In *Proceedings of the 6th International Working Conference on Database Theory (ICDT'97)*. 230–246.