Defining Process Performance Indicators: An Ontological Approach*

Adela del-Río-Ortega, Manuel Resinas, and Antonio Ruiz-Cortés

Universidad de Sevilla, Spain

Abstract. It is increasingly important to evaluate the performance of business processes. A key instrument to carry out this evaluation is by means of Process Performance Indicators (PPIs) as suggested in many methodologies and frameworks like, for instance, COBIT, ITIL or EFQM. As a consequence, it is convenient to integrate the management of PPIs into the whole business process lifecycle from its design to its evaluation. In this paper, we focus on the definition of PPIs as a necessary step to achieve that integration. Unfortunately, current proposals are not able to specify several usual types of PPIs, specially those related to data, and are not well designed to enable the automated analysis of PPIs at design-time. In this paper, we present an ontology for the definition of process performance indicators that overcomes this issue, explicitly defines the relationships between the indicators and the elements defined in a business process modelled in BPMN, and enables the analysis of PPIs at design-time. Furthermore, this ontology has been validated by means of several real-world scenarios.

1 Introduction

An important aspect in the business process lifecycle is the evaluation of business processes performance, since it helps organisations to define and measure progress towards their goals. Performance requirements on business processes can be specified by means of Process Performance Indicators (PPIs) with target values that must be reached in a certain period. A PPI is 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. Note that we use PPI as a kind of Key Performance Indicator (KPI) that focuses exclusively on the indicators defined on the business processes. Nowadays, many methodologies and frameworks like, for instance, COBIT, ITIL or the EFQM 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 this evaluation of business processes easier, it is convenient to integrate the management of PPIs into the whole business process lifecycle [1,2]

* This work has been partially supported by the European Commission (FEDER), Spanish Government under the CICYT project SETI (TIN2009-07366); and project P07-TIC-2533 funded by the Andalusian local Government.

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 processes that are 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, where the monitoring information obtained in the previous phase will help to identify correlations and predict future behaviour.

An appropriate definition of PPIs is key to enable the automated support of the aforementioned PPIs lifecycle. Unfortunately, in practice, such definition is done 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 [3]). Furthermore, although there are several research proposals to define PPIs, none of them are well-suited because they cannot express commonly used PPIs or they are not ready to enable a design-time analysis of PPIs or they do not define explicitly their relationship with the business process and, hence, make it difficult their use together with business process analysis techniques (*cf.* Section 6 for more details).

To overcome this issue, we present an ontology to define PPIs whose main benefits can be summarised as follows:

1. The relation between PPIs and the business process is explicitly established. This 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. It supports the definition of a wide variety of PPIs, including those associated with data objects. It also supports the definition of an expressive analysis period of a PPI. 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 6).
3. Dependencies between `ProcessMeasures` and `InstanceMeasures` can be automatically obtained from the ontology, which enables the analysis of PPIs at design time. Furthermore, since the ontology has been defined in OWL DL, automated reasoners can be used to make queries about the PPI model such as *how many PPIs are defined on the same `MeasureDefinition`?* or *how many PPIs are defined on a `TimeMeasure`?*.

Furthermore, we have validated the suitability of the ontology for the definition of real PPIs by using several real scenarios in different environments (the Information Technology Department of the Andalusian Health Service and the Justice and Public Administration Department of the Andalusian Local Government), in order to prove the applicability of our solution to actual scenarios.

The remainder of this paper is organised as follows. In Section 2 we present two case studies. Then in Section 3 we propose an ontology for the definition of PPIs. The ontology will be used in Section 4 to express the PPIs defined for the two case studies and to explain the way our ontology is applied to that definition. Section 5 details how dependencies between measure definitions can be inferred from the ontology. In Section 6 we present related work. Finally, Section 7 draws the conclusions from our work, summarizes the paper and outlines our future work.

2 Case Studies

In this section we introduce two real scenarios from different environments. Their processes were modelled by process modellers from each organisation using diverse languages (BPMN and flow diagrams) and the way PPIs were described by their process analysts was also different, even when it was always in a natural language (These activities were conducted before the definition of our ontology). We have unified them by modelling both scenarios in BPMN and presenting their corresponding PPIs in several tables. We intend to show the applicability of our approach to real cases from completely different organisations. Because of space constraint, in this paper we just include one process for each case study. However, more processes from these scenarios can be found at `http://www.isa.us.es/ppiontology/`.

2.1 Process of the Request For Change Management

First, 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, 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 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

Table 1. PPIs defined for the RFC management process

Description	**Periodicity**	**id**
(RFCs cancelled-registry error/RFCs registered)	annual	PPI1
Average time of committee decision	monthly and annual	PPI2
(corrective RFCs/approved RFCs)	monthly and annual	PPI3
(perfective and adaptive RFC/approved RFCs)	monthly and annual	PPI4
Average time of the ”analyse RFC” activity	annual	PPI5
Number of RFCs with the state ”in analysis”	monthly	PPI6
Number of RFCs per type of change	annual	PPI7
Number of RFCs per project	annual	PPI8
Number of RFCs per application	annual	PPI9
Average lifetime of a RFC	annual	PPI10

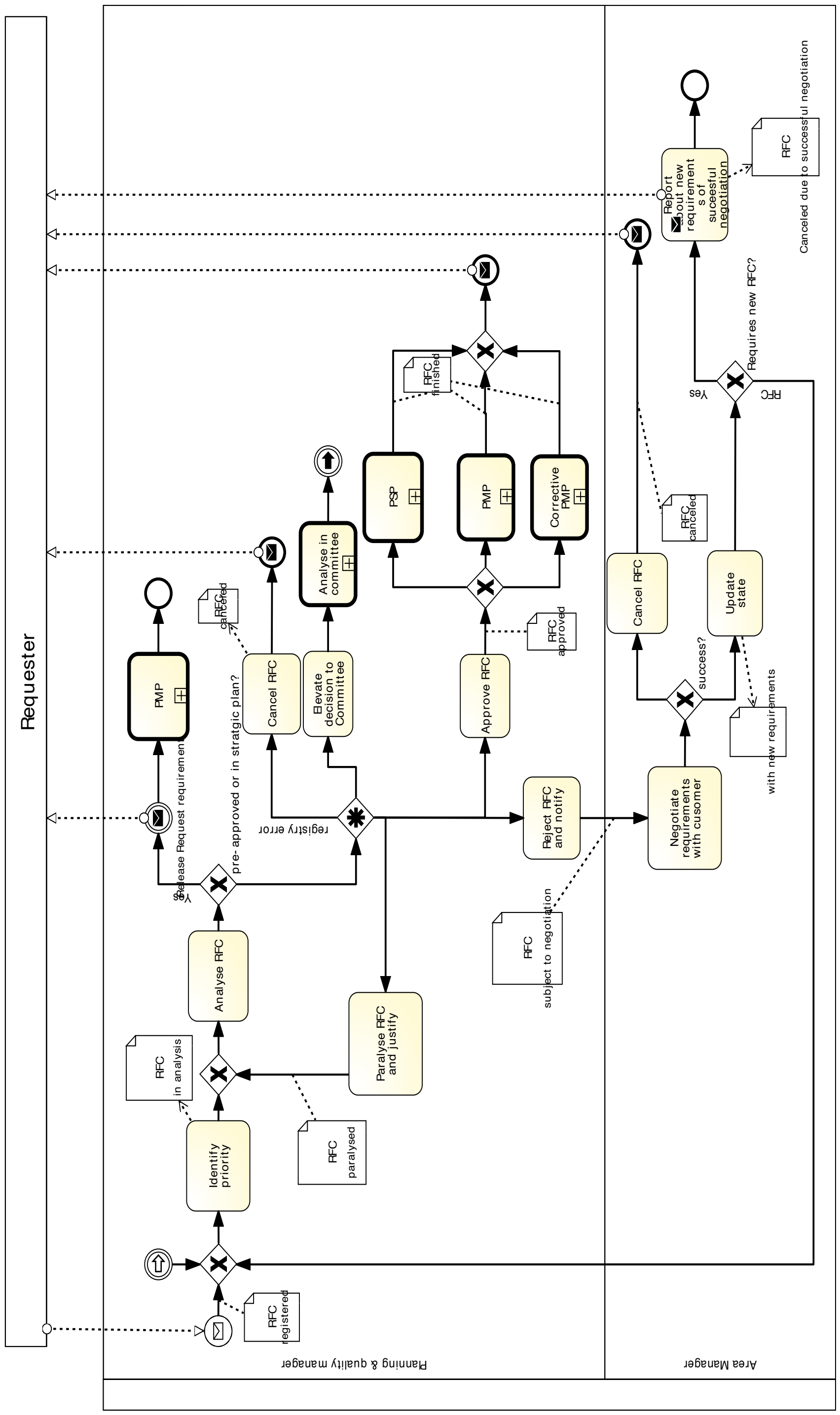

Fig. 1. Process of the request for change management

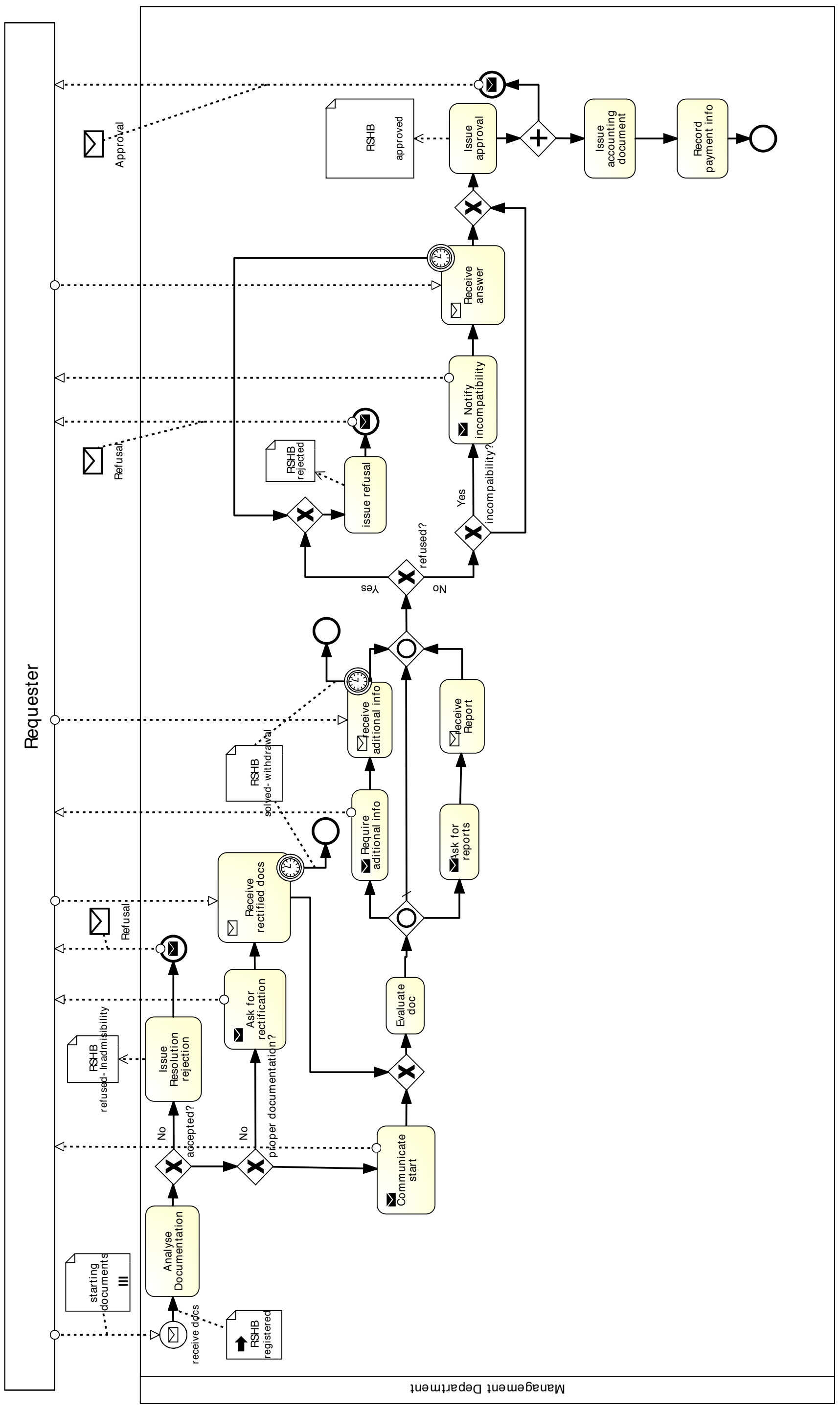

Fig. 2. Process of the social and health benefits management

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

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

After modelling the process, this department also defined a set of indicators associated with it, but they did it in a spreadsheet in a natural language. Table 1 lists the PPIs they defined for the Request For Change management process.

2.2 Process of the Social and Health Benefits Management

The second case study takes place in the context of the Consejería de Justicia y Administración Pública(Andalusian Local Government). It is the business process associated with the management of social and health benefits (Figure 2).

This process starts when the requester sends the required documentation. After that, the management department analyses the documentation and decides whether to deny, ask for rectifying the documentation or evaluate the request. Sometimes further information or reports are required. Finally, the decision of rejection or approval will be sent and the payment information will be recorded. The set of PPIs defined by the process analysts of the aforementioned organisation for this process are detailed in Table 2.

Table 2. PPIs defined for the RSHB management process

Description	**Periodicity**	**Id**
Number of requests submitted	monthly and annual	PPI11
Combined budgets of requests granted	monthly and annual	PPI12
(Requests approved/requets submitted) × 100	monthly and annual	PPI13
(Requests rejected/requests submitted) × 100	monthly and annual	PPI14
(Requests rejected$_{\text{inadmissibility}}$/requests submitted) × 100	monthly and annual	PPI15
(Requests rejected$_{\text{withdrawal}}$/requests submitted) × 100	monthly and annual	PPI16
$(resolutiondate - registrationdate)/requestsregistered$	annual	PPI17
$((resolutiondate - registrationdate) - dayswithrequestsparalised)/requestsregistered$	annual	PPI18

3 PPI Ontology

In the following, we present the ontology we have defined to specify PPIs[1]. We decided to use OWL DL [4] due to the high expressivity it offers and also because

[1] The OWL file can be downloaded from `http://www.isa.us.es/ppiontology/`

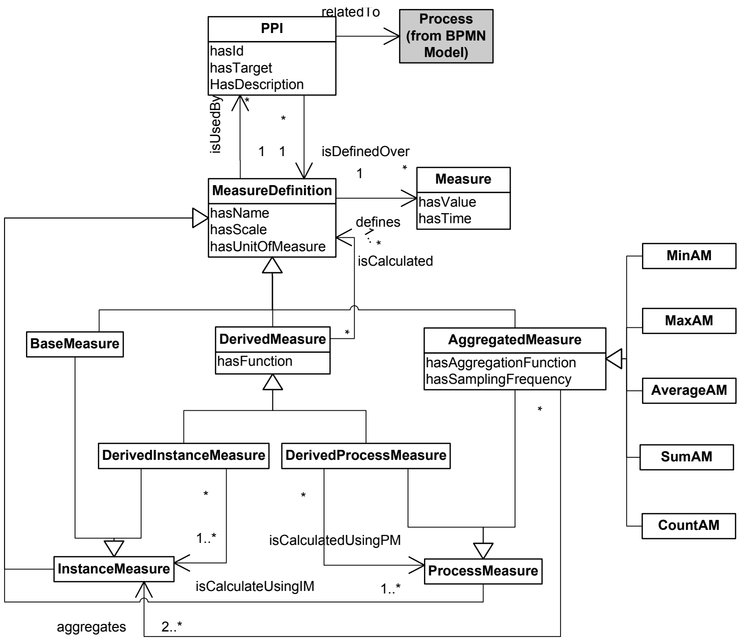

Fig. 3. PPI Ontology (overview)

there exist reasoners that allow to infer knowledge and make queries. Figure 3 depicts the high level view of our ontology. The ontology is represented using UML as proposed in [5]. Those elements depicted in a grey color do not belong to our ontology but they are concepts borrowed from BPMN.

A PPI is referred to by means of a `hasID`, described through its `hasDescription` and has a `hasTarget` restriction, which is the objective to achieve. PPIs are related to a process, and are defined by a `MeasureDefinition`. In order to define a measure, we need to specify its `hasName`, `hasScale` (set of values with defined properties, e.g. natural, integer, float, map) and, in some cases, `hasUnitOfMeasure`. `MeasureDefinitions` are used to define measures, which take values (`hasValue`) 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 or we calculate the value using a set of instances by aggregating them (`aggregates`), 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`.

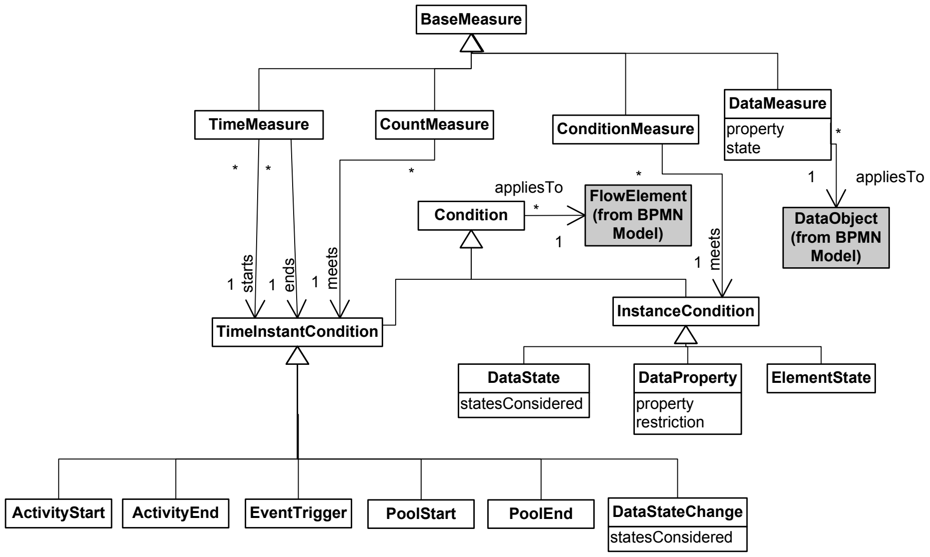

Fig. 4. PPI Ontology (`BaseMeasure` classification)

– 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.

3.1 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 shown in Figure 4, we can measure time (`TimeMeasure`), the number of times that something happens (`CountMeasure`), or whether certain property of a data (`DataMeasure`), or a concrete condition (`ConditionMeasure`) is fulfilled.

`TimeMeasure` : In this case, the duration between two `TimeInstantConditions` (`start` and `end`) will be measured. These `TimeInstantConditions` can be associated with the start or the end of an activity or a process contained in a pool, with the trigger of an event, or with the change of the state of a data. An example of this kind of measure would be "the duration of the activity *analyse RFC*", and the start and end conditions would match the beginning and the end of this activity.

`CountMeasure` : It counts the number of times a `TimeInstantCondition` is met, i.e. the number of times an activity or a pool starts or ends, an event is triggered

or a DataObject passes through certain states. For instance, “The number of times a RFC is analysed in an instance” is an example of this measure, and it would be measured by counting the number of times condition `activityEnd` for the activity *analyse RFC* is met. Note that it can be greater than 1 for an instance since there may be loops in the process.

ConditionMeasure : We can also check if certain `InstanceConditions` are being or have been met. These `ConditionMeasures` always take a boolean value for each instance. For instance, it could be useful to know whether data instances are or have finished in a given `DataState` or a set of them (e.g. “RFCs in state *paralysed*”). Or even we can check if some of the properties of a dataObject (`DataProperty`) meet a particular restriction, e.g. data with priority = “high”. The last possibility we consider for `ConditionMeasures` is the `ElementState`, in which whether a `FlowElement` is currently in execution or not is checked, i.e. if it contains or not an execution token while taking the measure.

DataMeasure : It measures certain properties contained in the `DataObjects` themselves, e.g. number of information systems that a RFC affect to.

Note that all these `BaseMeasures` are defined (`appliesTo`) over a concrete `FlowElement` (or two in the case of `TimeMeasures`) of the associated BPMN diagram. Depending on the kind of `Condition`, this element will be an activity, a pool, an event, a data object, etc. We do not depict all these correspondences in our figure for the sake of simplicity and readability.

3.2 AggregatedMeasures

In this type of `MeasureDefinition`, the value of the measure is calculated by applying a certain `aggregationFunction` on a set of measures (belonging to different instances) to obtain one single value. Depending on whether the `aggregationFunction` applied is minimum, maximum, average, sum or count (number of ”trues” for the boolean values), these measures can be `MinAM`, `MaxAM`, `AvgAM`, `SumAM` or `CountAM` respectively. An example of `AggregatedMeasure` would be “the number of RFC rejected in the last year”, that would be calculated through the aggregation function SUM (it would be a `SumAM`). However, there are three issues to be considered regarding which are the process instances whose measure will be used to calculate the `AggregatedMeasure`. First, a sampling frequency can be defined, so that we do not need to measure every instance, but one out of X, being X the sampling frequency. This makes sense in environments where taking a measure is hard or costly (e.g. when the measure can not be obtained automatically). Second, when aggregating measures, it may be useful to group them by certain condition (`InstanceCondition`), for instance, ”the number of RFCs per project”. In such a case, the number of RFCs would be added (`SumAM`) and then they would be grouped (`isGroupedBy`) by the `DataProperty` project. The result would be a map, with a value per each project. Third, usually an `AggregatedMeasure` defines a temporal range to consider when measuring. This temporal range is defined by means of an `analysisPeriod` (depicted in

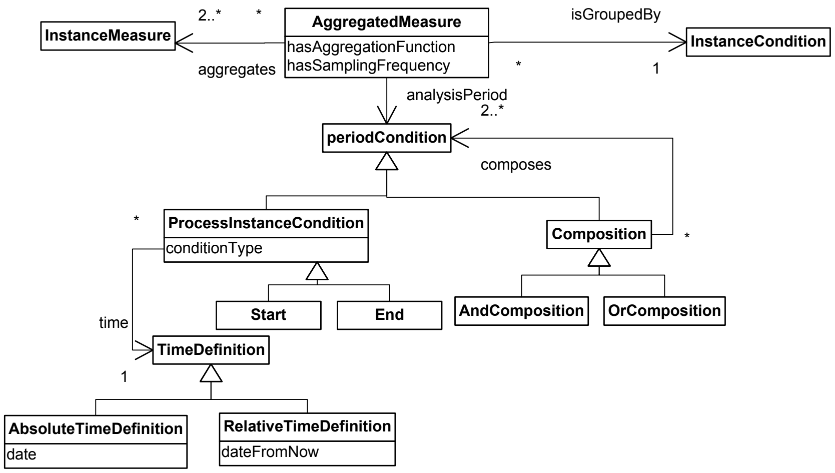

Fig. 5. PPI Ontology (Analysis period definition)

Figure 5) and must be understood as a temporal condition that must hold every process instance for its measures to be included in the aggregation.

The condition (`ProcessInstanceCondition`), which can be of different types (`conditionType`: $>$, $\geq$, $<$ and $\leq$), is defined between the start/end of process instances and the moments in time (`TimeDefinition`), usually two, that define the period(s) in which to take the measures. Moreover, these conditions can be composed by means of an AND and/or an OR operator when necessary. For instance, it could be interesting to measure the number of RFCs rejected during the last holidays' periods, both Christmas and summer. Thus, only the measures whose process instances finished between 18-12-2009 and 4-1-2010, or between 1-8-2009 and 31-8-2009 would be considered.

3.3 DerivedMeasures

Their value is calculated by performing a mathematical function to combine two or more `MeasureDefinitions`. Depending on whether the measures combined are instance or process measures, the result will be a `DerivedInstanceMeasure` or a `DerivedProcessMeasure` respectively. 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".

4 Validating Our Ontology

To validate the suitability of the ontology for the definition of real PPIs, we translate textual descriptions of indicators defined by process analysts from several real scenarios to their equivalent representation using the PPI ontology

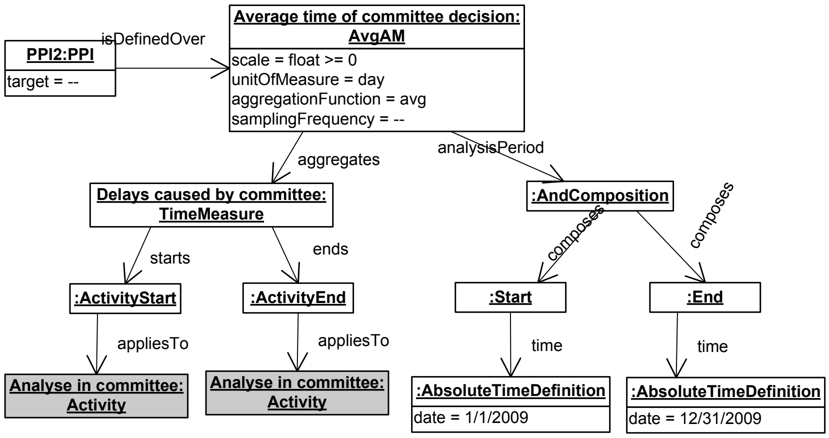

Fig. 6. Definition of the PPI2 ”Delays caused by the committee”

presented in Section 3. However, due to space constraints, we only include in the paper the definition of some of the PPIs from our two case studies (the ones we considered more representative). The definition of the remaining PPIs of the processes included in this paper together with PPIs from other processes are available at `http://www.isa.us.es/ppiontology/`.

4.1 Process of the Request for Change Management

In Section 2 we introduced a real scenario in the context of the Information Technology Department of the Andalusian Health Service whose PPIs were listed in Table 1. With our ontology, we can directly express all of those PPIs.

PPI2 (Figure 6). We omit the description in the PPI box for space reasons. This PPI2 is defined over an aggregated measure that calculates the average of a time base measure. This time measure represent the duration of the activity *analyse in committee.*

PPI8 (Figure 7). It is defined over an aggregated measure that counts all the RFCs registered, grouping them by the project they belong to. The result of this PPI will be a map, with one value per project.

4.2 Process of the Social and Health Benefits Management

Anew, we proceed as we did with the previous scenario, now refereing to the process associated with the management of social and health benefits. We transform the textual descriptions of indicators contained in Table 2 to our ontology. We choose again two indicators for doing so.

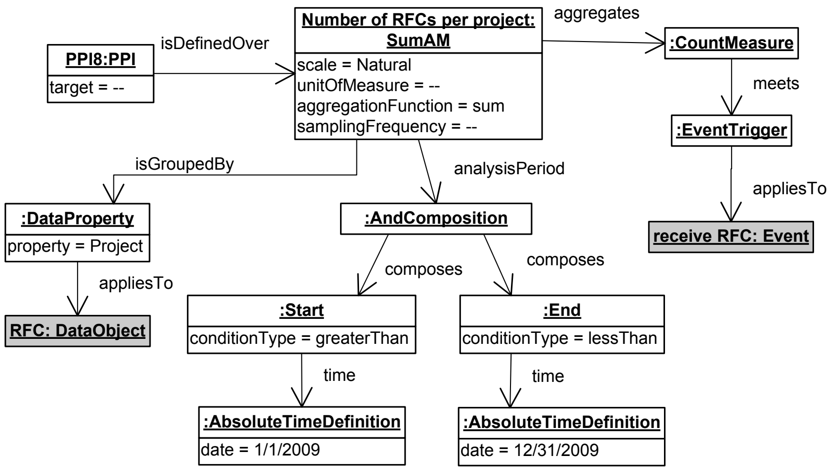

Fig. 7. Definition of the PPI8 "Number of RFCs per project"

PPI12 (Figure 8). This PPI is defined over an aggregated measure that sums the budget of each RSHB (Request of Social or Health Benefit) approved during last year (instances started on or after the first of January and ended before or on the 31st of December).

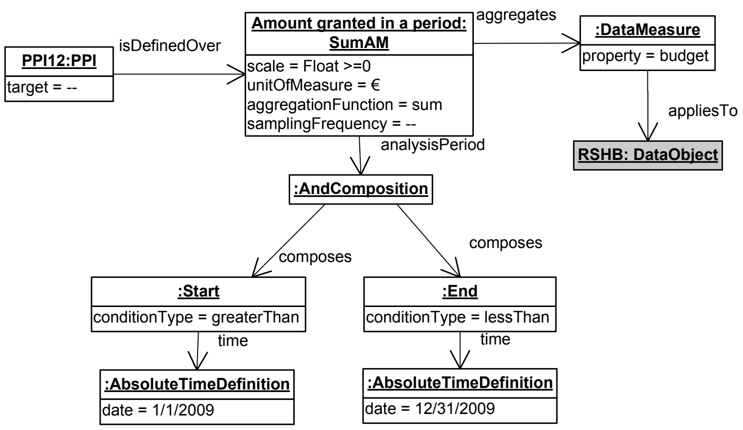

Fig. 8. Definition of the PPI12 "Amount granted in a period"

PPI13 (Figure 9). It is defined over the derived process measure *percentage of requests approved*, that is calculated using two aggregated measures, according to the defined function. The first one sums the number of requests approved last year by counting the number of times the activity *Issue approval* finished in that period. The second one sums the number of requests registered last year by

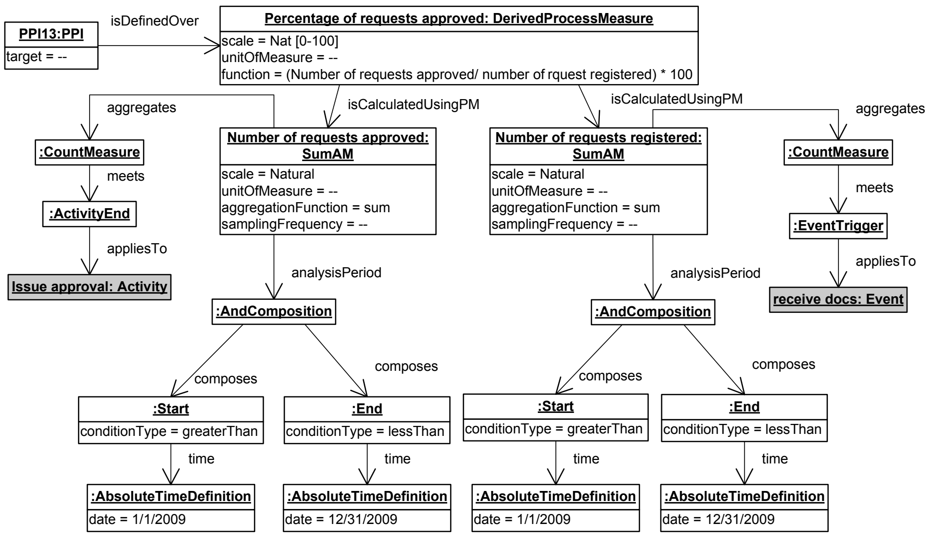

Fig. 9. Definition of the PPI13 "Percentage of requests approved"

counting the number of times the event *receive doc* is triggered. The definition of the analysis period is the same for both of them and coincides with the one of the previous PPI.

5 Modelling Dependencies on the PPI Ontology

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: `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 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, two new properties can be added to the ontology, namely: `dependsOn` and its inverse property, `isDepended`. Furthermore, each of these two relationships 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` $m2$, 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 as inference rules in the ontology as follows[2]:

$$\begin{aligned} directlyAggregates(?x, ?y) &\longrightarrow dependsDirectlyOn(?x, ?y) \\ isCalculatedPositively(?x, ?y) &\longrightarrow dependsDirectlyOn(?x, ?y) \\ isCalculatedNegatively(?x, ?y) &\longrightarrow dependsInverselyOn(?x, ?y) \end{aligned}$$

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

$$\begin{aligned} &isCalculatedNegatively(?x, ?y), \\ &\qquad dependsInverselyOn(?y, ?z) \longrightarrow dependsDirectlyOn(?x, ?z) \\ &isCalculatedNegatively(?x, ?y), \\ &\qquad dependsDirectlyOn(?y, ?z) \longrightarrow dependsInverselyOn(?x, ?z) \\ &isCalculatedPositively(?x, ?y), \\ &\qquad dependsDirectlyOn(?y, ?z) \longrightarrow dependsDirectlyOn(?x, ?z) \\ &isCalculatedPositively(?x, ?y), \\ &\qquad dependsInverselyOn(?y, ?z) \longrightarrow dependsInverselyOn(?x, ?z) \end{aligned}$$

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, 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 *vice versa*.

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

Table 3. Comparison of the analysed approaches and our proposal

Proposal	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**
Pedrinaci et al. [6]	N/A	✓	✗	✓	∼	∼
Popova et al. [10]	✓	✓	N/A	✗	✓	✗
Mayerl et al. [12]	∼	✗	✗	✓	∼	✗
Castellanos et al. [13]	∼	✗	N/A	∼	✗	✗
Momm et al. [14]	✓	✗	✗	✗	✗	✗
Wetzstein et al. [15]	✓	✗	✗	✗	∼	✗
Our Proposal	✓	✓	✓	✓	✓	✓

(1) Explicit relation with the business process (4) Derived measures
(2) Definition of analysis period (5) Analysis of PPIs (Design-time)
(3) Data measures (6) Queries about PPIs

6 Related Work

There already exists a number of proposals that use ontologies in business process management [6], ranging from ontologies supporting the definition of organisational structures [7], business processes [8] or even business goals to ontologies for capturing execution logs and supporting business process analysis [9]. There are also some other ontologies dedicated to define measures in different areas, like [11] for software measurement or the one introduced by Pedrinaci et al. [6], which describes a Semantic Business Process Monitoring Tool called SENTINEL, and, therefore, it is closer to our proposal. 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 and it is based on 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.

Outside the semantic web-based approach, there are also several approaches to evaluate the performance of business processes defined in the literature and, in some cases, implemented in products.

Popova et al. [10] present 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 [18], 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 [10] and explained in detail in [18]. According to the definition of the analysis period, they define temporal properties over PPIs (called PI expressions) in [19]. They do not consider derived measures nor queries in their works.

In [12] Mayerl et al. discuss how to derive metric dependency definitions from functional dependencies by applying dependency patterns. 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 [13] 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 can be defined with this approach.

In [14], Momm et al. 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, PPI12 and PPI16 from our examples can not be defined according to this metamodel). Moreover, it lacks some properties when defining PPIs like the analysis period or the function to calculate derived measures.

Another work which is close to ours is the one presented by Wetzstein et al. in [15]. 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).

To sum up all this information and establish an explicit comparison between the approaches analysed above and our proposal, we present Table 3. We highlight those benefits we assigned to our ontology in Section 1: relation between PPIs and BPs (feature 1), definition of a wide variety of PPIs (feature 2, 3 and 4), and analysis and queries on PPIs (features 5 and 6). We use the following notation: A ✓ sign means that the proposal successfully addresses the issue; a $\sim$ sign indicates that it addresses it partially; a ✗ sign indicates that it does not contemplate the issue; and N/A means the information is not available.

7 Conclusions and Future Work

In this paper, we argue the importance of integrating the management of PPIs into the whole business process lifecycle. Specifically, in the design and analysis phase, PPIs must be modelled together with the business process. This model should enable (at design-time) an automated or semi-automated analysis to, for instance, detect the dependencies and potential conflicts amongst them or to use them together with other business process analysis techniques such as simulation techniques. As a consequence the mechanism used to define PPIs must be expressive enough to allow the definition of a wide range of PPIs; it must establish explicitly the relation between PPIs and the business process, and it must enable the analysis of PPIs at design-time.

To this end, we present an ontology to describe these indicators (allowing the definition of such a variety of PPIs). We also identify how they depend on the performed activities and other business objects by establishing these relationships between PPIs and business processes explicitly. Finally the definition of PPIs using OWL-DL enables their analysis at design-time in a way that is amenable to automated reasoning, as outlined in Section 5. Furthermore, we have validated the ontology against several real-world case-studies to check its expressiveness.

Our future work focuses on increasing the number of types of PPIs that can be defined using the ontology and improving the automated analysis capabilities. Regarding the former, we want to extend our ontology to allow the specification of complex conditions, for instance, *number of RFCs approved after being previously rejected*. In the last case, complex queries on BPMN such as those defined in BPMN-Q [16] could be used as a type of `Condition`. Regarding the latter, we plan to extend the automated detection of dependencies between measure definitions to detect dependencies amongst different base measures. Furthermore, we are developing a graphical notation in order to depict these ontological concepts of PPIs over business processes and we are also integrating this notation into the web-based editor ORYX [17] as a result of a collaboration stay with the BPT group at the HPI Potsdam.

Acknowledgments

We would like to thank to 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. We also would like to thank F. Casati, J. Fabra, J. M. García, C. Pedrinaci, F. Ruiz, A. Sharpanskykh and M. Weske for their helpful comments in earlier versions of this paper.

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Heidelberg (2007)
2. del Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Towards modelling and tracing key performance indicators in business processes. In: II Taller sobre Procesos de Negocio e Ingeniería de Servicios, PNIS 2009 (2009)
3. OMG: Business process modeling notation (BPMN) v2.0 (2009)

4. W3C OWL Working Group: Owl 2 web ontology language (2009)
5. Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual modeling of OWL DL ontologies using UML. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 198–213. Springer, Heidelberg (2004)
6. Pedrinaci, C., Lambert, D., Wetzstein, B., van Lessen, T., Cekov, L., Dimitrov, M.: Sentinel: a semantic business process monitoring tool. In: OBI, p. 1 (2008)
7. Abramowicz, W., Filipowska, A., Kaczmarek, M., Pedrinaci, C., Starzecka, M.A.W.: 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 (2008)
8. Abramowicz, W., Filipowska, A., Kaczmarek, M., Kaczmarek, T.: 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 (2007)
9. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A core ontology for business process analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 49–64. Springer, Heidelberg (2008)
10. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. In: Information Systems (2009)
11. Garcia, F., Bertoa, M.F., Calero, C., Vallecillo, A., Ruiz, F., Piattini, M., Genero, M.: Towards a consistent terminology for software measurement. Information & Software Technology (48), 631–644 (2006)
12. Mayerl, C., Hüner, K., Gaspar, J.U., Momm, C., Abeck, S.: 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), pp. 1–10 (2007)
13. Castellanos, M., Casati, F., Shan, M.C., Dayal, U.: ibom: a platform for intelligent business operation management. In: Proceedings of 21st International Conference on Data Engineering, pp. 1084–1095. Hewlett-Packard Laboratories (2005)
14. Momm, C., Malec, R., Abeck, S.: Towards a model-driven development of monitored processes. Wirtschaftsinformatik (2), 319–336 (2007)
15. Wetzstein, B., Ma, Z., Leymann, F.: Towards Measuring Key Performance Indicators of Semantic Business Processes. In: 11th International Conference on Business Information Systems (BIS 2008) & Innsbruck, Austria, pp. 227–238 (2008)
16. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using bpmn-q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
17. Decker, G., Overdick, H., Weske, M.: Oryx - An Open Modeling Platform for the BPM Community. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 382–385. Springer, Heidelberg (2008)
18. Popova, V., Sharpanskykh, A.: Formal analysis of executions of organizational scenarios based on process-oriented specifications. In: Applied Intelligence (2009)
19. Popova, V., Sharpanskykh, A.: Formal Goal-based Modeling of Organizationss. In: MSVVEIS, pp. 19–28 (2008)