A Framework to Support Business Process Analytics

Alejandro Vera Baquero and Owen Molloy
Information Technology, National University of Ireland, Galway, Ireland

Keywords: Business Intelligence, Business Activity Monitoring, Business Performance Management, Business Process Analytics, Event Modelling, Business Process Execution Language.

Abstract: Business intelligence (BI) systems have become a powerful tool for business users in decision making. Through the analysis of historical (and increasingly, real-time) data, these systems assist end-users in achieving visibility on process and business performance. While traditionally used to discover trends and relationships in large, complex business data sets, there is a significant and growing demand for something more than the use of mere historical data and rudimentary analysis tools. There is a demand for more advanced analytics such as root cause analysis of performance issues, predictive analysis and the ability to perform "*what-if*" type simulations. This paper proposes a technological solution for one of the core components of these emerging BI systems, namely the ability to monitor and analyse the execution outcomes of business processes. This provides essential insight into business process performance, key intelligence in initiatives aimed at measuring and improving overall business performance, especially in highly distributed business processes, where this type of visibility is especially hard to achieve across heterogeneous systems.

1 INTRODUCTION

Business Intelligence (BI) and Business Performance Management (BPM) systems are complex, expensive and require considerable resources and time to implement. However, in most (distributed) process improvement initiatives we are faced with the problem of heterogeneous systems and standards. This applies of course to not only Enterprise Resource Planning (ERP) systems, but to a range of Human Resources (HR), Customer Relationship Management (CRM), Finance and other systems which may implement parts of the processes we are interested in monitoring and improving (see Figure 1).

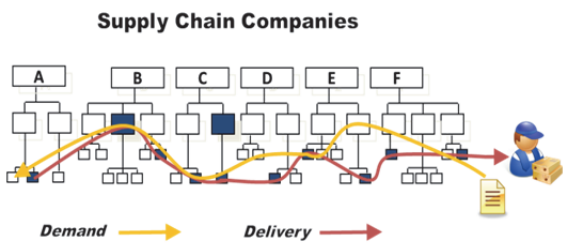


Figure 1: Heterogeneous systems challenge.

This is further complicated by the fact that very often process improvement initiatives, such as Lean/Six-Sigma projects, need to be fast and agile, easily moving from modelling to measurement and analysis without investment and overheads for what may be a rapidly changing business and process environment.

While there has been significant deployment of systems based on BPEL (Business Process Execution Language) technology in recent years, they generally still represent just parts of the distributed business processes we are interested in. While we can use Business Intelligence (BI) systems to pull information from heterogeneous systems into pre-defined data warehouses, this comes at a high cost in terms of time and resources. Another significant drawback from a process improvement perspective is that BI systems are not typically process-aware and must be re-engineered in response to changes in the process design. On the other hand process-aware, or *process-oriented* systems allow querying directly on the process data itself, while maintaining knowledge about the process design or model. In a 100% BPEL world this would not be a problem, as we could directly mine BPEL databases, however as we have already stated, this is not the case. Therefore we present a flexible, lightweight, BPEL-agnostic solution for process monitoring in this paper.

1.1 Overview of Business Activity Monitoring

It is useful at this point to discuss the common terminology to be used in describing process performance. When we discuss processes we must remember that a process may itself be composed of a number of different sub-processes or activities which in turn may be decomposed into a set of smaller related tasks (see Figure 2). There is no globally accepted limit on the number of levels, and depending on circumstances and data requirements, it may be necessary to monitor both high level and low level processes simultaneously.

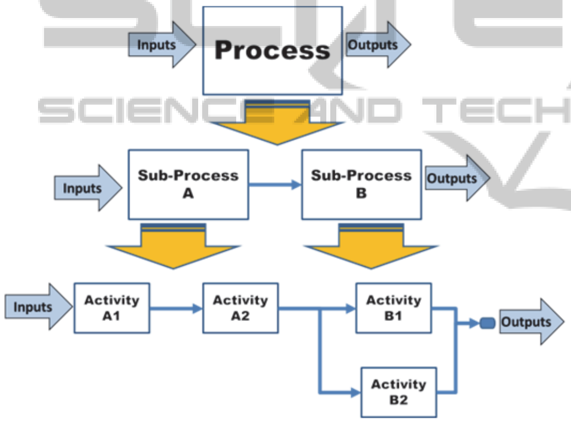


Figure 2: Sample Process Hierarchy.

An interesting aspect of this, which is relevant from a management perspective, is that we should be able to monitor and manage performance at all levels in the hierarchy. Some common process measures are presented in Figure 3.

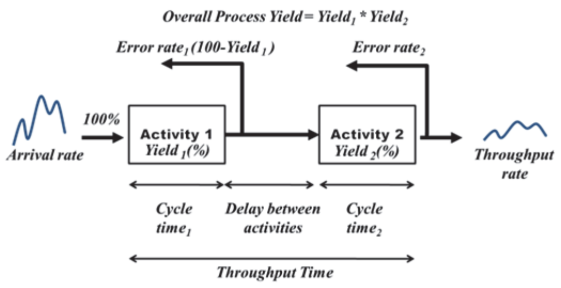


Figure 3: Sample Process Measures.

In a well running process we expect arrival (demand) and throughput rates to be in balance. This requires each stage in the process to be capable of working at the rate at which inputs (e.g. .orders) arrive. Processes or activities which do not have the capacity to work to this arrival rate become bottlenecks which must be identified and eliminated. As well as causing unnecessary delays and prompting *"fire-fighting"* responses from management, bottlenecks can starving proceeding activities of input. This results in valuable resources such as people or machines being idle and underutilized. The strategies for elimination of bottlenecks will depend on our scope of action: it may be possible to add extra processing resources, or duplicate processes, or even to modify the arrival rate through negotiation with the customer. Therefore, the concepts of yield, error rate, throughput etc. can be validly applied at all levels.

A Business Activity Monitoring (BAM) component is clearly essential "for better business performance or continuous process improvement of an enterprise, real-time measurement and analysis of the performance of managerial activities" (Kang and Han, 2008). A BI component capable of deriving higher level intelligence from the basic BAM data is necessary for the creation of knowledge from a business process perspective. Whereas the mainstream BI systems of 10 years ago were definitely not "process aware", the overall process-oriented approach (Seufert and Schiefer, 2005) is gaining importance for companies seeking to remain not just competitive but also viable in today's business environment.

2 THE FRAMEWORK

This paper proposes a framework that provides business users with the ability to monitor and analyse the execution outcomes of business processes. The framework presented aims to assist analysts in gaining an insight into business process performance.

The contribution of this research pursues two main aims. The first aim is to provide a generic event model construct that can represent the execution data of any business process regardless of the environment in which it is executed. The second aim is to provide an IT infrastructure with the ability to monitor business processes from operational systems and analyse their execution outcomes.

The next figure shows the architectural approach of the proposed framework which is broken down into two main components. A BAM component that is responsible for providing event stream processing capabilities and a BI component which is the functional unit that produces the analytical information on business process performance.

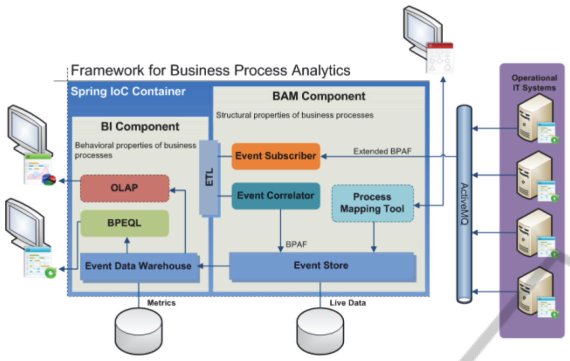


Figure 4: Architectural approach of the framework.

2.1 An Event Model for Business Activity Monitoring and Business Process Analysis

An event-based model is essential to provide the framework of a concrete understanding and representation of what needs to be monitored, measured and analysed (Costello, 2008).

The event model proposed in this paper is built upon the BPAF (Business Process Analytics Format) standard, specified in (WfMC, 2009), combined with some important features of the iWISE model discussed in (Costello, 2008). The BPAF standard has been extended in order to accommodate the event correlation features defined by the iWISE software.

2.1.1 iWISE

The iWISE software is an IT platform that provides a full infrastructure to manage process models and to monitor the activity of business processes with the aim of capturing enterprise events from business systems and leveraging such data to detect non-compliant situations.

The iWISE system is fully described in (Costello, 2008) and defines an event-based model to represent the results of business process executions as business events supplied from heterogeneous environments where processes cross both organizational and software boundaries. The main modelling constructs are depicted using a simplified UML (Unified Modelling Language) class diagram depicted in Figure 5.

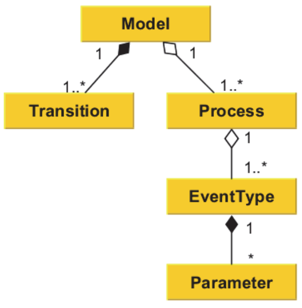


Figure 5: High-level event-based model (Costello, 2008).

The model entity is the root element of the process model. It represents the definition of a process model and contains a list of processes connected by transitions. In turn, each process has multiple event type definitions which may be associated to a set of parameters (Costello, 2008).

The iWISE event model is structured in an XML format that represents the UML specification described above. The process element contains the relevant information of a process instance, which in turn, it references to a list of events associated. These events are modelled in the `EventType` element containing the definition of an event instance. This element is used to provide some meta-data about the event. In addition, each process can have a reference to another model, thus enabling a structure for accommodating multiple levels of sub-process or activities (Costello, 2008).

A significant contribution to the event model presented in this work from iWISE is the structure used to represent a business event. An event is documented in the `Event` element and references and `EventType` through the `EventTypeID` element (see Figure 6). This element establishes the relationship between the events and their respective process instances.

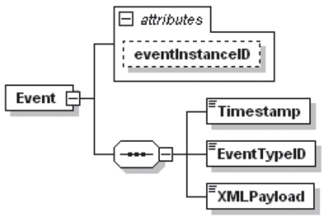


Figure 6: The iWISE Event element (Costello, 2008).

The `Event` elements are specified below.
`EventInstanceID`: Unique event identifier.
`EventTypeID`: Specifies the event type information.
`Timestamp`: Element that contains the time at which the event occurred.

`XMLPayload`: The XML data containing the business information.

The `XMLPathExpression` element, located in the `EventType` data, is used to identify an element or attribute within the XML document contained in the `XMLPayload` of the event. The iWISE software uses XPath to retrieve a part of the message payload in XML format, that will later be used to uniquely identify an event instance for a particular process instance during execution (Costello, 2008)

2.1.2 The BPAF Model

BPAF is a standard format published by the Workflow Management Coalition to support the analysis of audit data across heterogeneous business process management systems (WfMC, 2009). It enables the delivery of basic frequency and timing information to decision makers, such as the cycle times of processes, wait time, number of process instances completed against the failed ones, etc. (Zur Muehlen and Shapiro, 2009).

BPAF is designed as an XML schema and consists of a generic design for a process analytics system which provides an event format independent of the underlying process model. This format enables analytic applications and BAM technology to unify criteria and to standardize a state model for auditing event purposes in heterogeneous environments (Zur Muehlen and Shapiro, 2009).

The BPAF state model and transitions are depicted at the following figure.

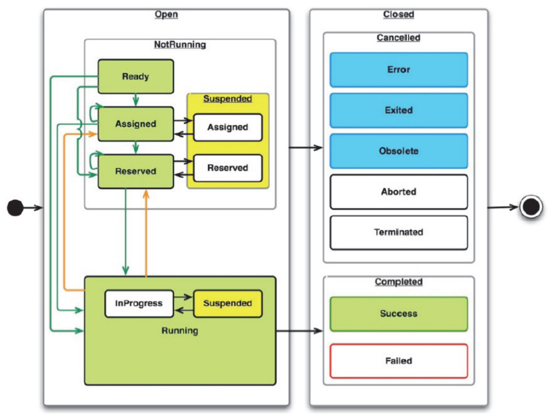


Figure 7: BPAF State Model (Zur Muehlen and Shapiro, 2009).

2.1.3 The Extended BPAF Model

The BPAF has been modified with the purpose of achieving two main aims. The first aim is to provide the framework with the data required to correlate the events produced by the execution of cross-organizational business processes. And the second aim is to accommodate the structural properties of process or activity instances that are of relevance to business analysts.

The extended and modified elements are specified below.

`ServerID`: The proposed framework requires uniquely identifying the business systems that originated the event since the business processes are to cross software boundaries throughout a diverse of disparate software systems. Therefore, this attribute must be mandatory.

`ActivityParentID`: The proposed format permits to accommodate an unlimited number of levels for sub-process / sub-activities. This is accomplished by keeping a reference to its closest parent.

`DataElement` [multiple, optional]: A name-value-pair that can be used to store additional process data.

`Payload` [multiple, optional]: A name-value-pair that is used to uniquely store the event payload of processes or activities.

`ProcessInstanceID` [optional]: The identifier of the process instance whose source system has the ability to identify a process instance by the means of an identifier. This element is optional as not all source systems are able to manage unique identifiers on the execution of their business processes.

`Correlation` [multiple, optional]: A name-value-pair that stores a subset of elements contained on the event payload and that are used to identify a determined process or activity.

2.2 Event Correlation

The event correlation is an essential part of the proposed framework for achieving the correct identification of process execution sequences. Without the ability to correlate events it would not be possible to generate metrics per process instance or activity (Costello, 2008), and thus the business analysts would be unable to identify exceptional situations and potential improvement opportunities.

This work proposes an event correlation mechanism based on the data shared between business processes during their execution. In an event-driven approach, such shared data usually makes reference to the message payload, and this information can be used to identify the start and end event data for a particular process instance or activity. The main difficulty with this approach is in determining which part of the event payload is used to identify and link the consecutive events.

The correlation process basically consists of associating every event instance to the correct process or activity. The identification of the process or activity is undertaken by retrieving the exact instance associated with a specific process model, executed at a particular source and provided with specific correlation data. This triplet allows us to determine the process instance or activity that is the owner of a specific event.

In source systems that have the ability to generate and manage identifiers on their source instances, such as BPEL engines, it is not necessary to provide any correlation information on the event message. In such cases, the instance identifier is provided instead, and in turn, this is used to correlate the subsequent events

2.3 Analytics Requirements

At the most basic level, operational systems deliver timing information on the event occurrence. The majority of process metrics are obtained by analysing the timestamp of a set of correlated events which are associated to a determined process instance or activity (Zur Muehlen and Shapiro, 2009). The use of such metrics provides business analysts with an understanding of the behavioural aspects of business processes.

The proposed framework captures and records the timestamp of events containing the time at which they occurred on the source system, not when they are packaged or delivered. This property is essential in order to identify and analyse the correct sequence of process instances, as well as ensuring that the generation of metrics produces precise information on its outcomes.

Based on the event timing information and the BPAF state model presented in previous sections, it is possible for an analyst to determine the measurement of different behavioural aspects of business processes with the aim at "evaluating what happened in the past, to understand what is happening at present and to develop an understanding of what might happen in the future" (Zur Muehlen and Shapiro, 2009).

Michael zur Muehlen in (Zur Muehlen and Shapiro, 2009) propose leveraging the state change records in the life cycle of business process to determine the following information:

- **Turnaround.** Measures the gross execution time of a process instance or activity.
- **Wait Time.** Measures the elapsed time between the entrance of a process or activity in the system and the assignment of the process or activity to a user prior to the start of its execution.
- **Change-over Time.** Measures the elapsed time between the assignment of the process or activity to a user and the start of the execution of the process or activity.
- **Processing Time.** Measures the net execution time of a process instance or activity.
- **Suspend Time.** Measures the time a execution of a process or activity is suspended.

Figure 8 illustrates the metrics outlined above by depicting a sample of the execution of an activity instance as per the BPAF state model.

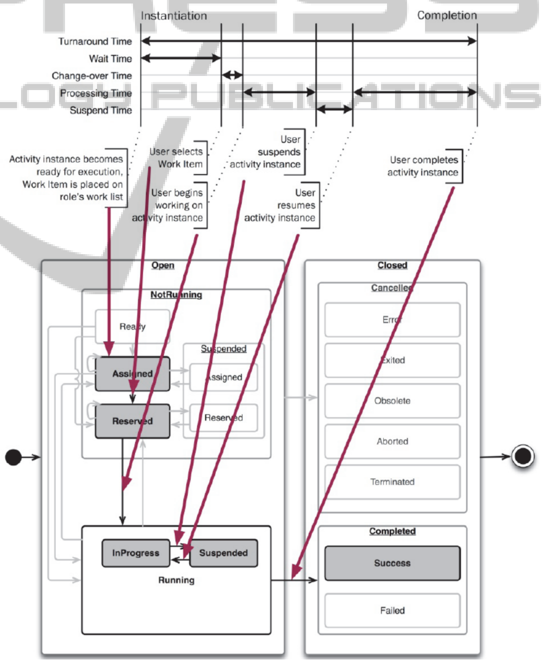


Figure 8: Activity Instance Metrics (Zur Muehlen and Shapiro, 2009).

2.4 Architecture of the Framework (F4BPA)

The architecture of the framework for business process analytics (F4BPA) is illustrated in Figure 9 and consists of two main subsystems, the Business Activity Monitoring subsystem and the Business Intelligence subsystem. They are both built upon the Spring Framework version 3, and hence, they are Java-based enterprise applications.

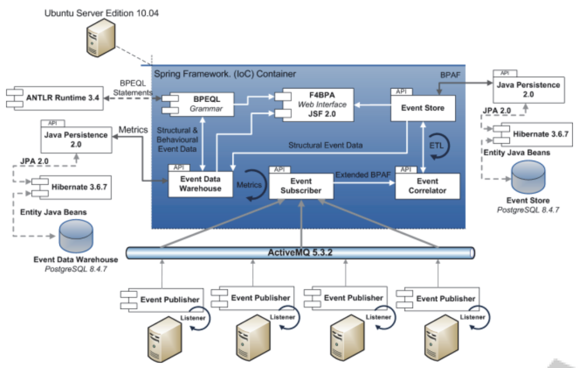


Figure 9: F4BPA high-level system architecture.

The BAM subsystem is composed of a set of listener software modules (**Event Publisher**) that collects the events from business systems and publishes them through a message broker platform, an **Event Subscriber** module that listens and processes the incoming events, an **Event Correlator** module that identifies and correlates consecutive events, and an **Event Store** module which persists the event data.

The BI subsystem is composed of an **Event Data Warehouse** and a **Business Process Execution Query Language (BPEQL)** module. The Event Data Warehouse is responsible for the generation and persistence of metrics, as well as serving as a data interface for querying the data warehouse containing metrics. The BPEQL module basically parses, executes and returns the results of query statements.

2.4.1 Business Process Execution Query Language

One key challenge in decision making is having access to all relevant information in order to undertake a performance and compliance assessment. Such information is normally distributed on diverse heterogeneous systems belonging to different organisational units. In such cases, not only the gathering, unification and correlation of event data are required, but also the ability to query the event repository and display the data thereof.

Many query languages for business processes have been proposed, using a variety of different approaches such as SQL-like languages, languages based on graphs and ontologies.

The BP-Ex query language proposed in (Balan, Milo and Sterenzy, 2010) is a user-friendly interface based on a graph representation for querying business process execution traces.

The FPSPARQL is a query language for analyzing event logs of process-oriented systems based on the concepts of *folders* and *paths*. These concepts enable analysts to join related events together and additionally, store the folders and paths to later be used in future analysis. FPSPARQL extends the SPARQL graph query language by implementing progressive techniques in a graph processing engine (Behesti, Benatallah, Motahari-Nezhad, and Shakr, 2011).

The EP-SPARQL (Event Processing SPARQL) is an extension of the SPARQL querying language for event processing and stream reasoning that enables stream-based querying (Anicic, Fodor, Stojanovic and Rudolph, 2011).

The query language proposed in this work resembles the SARI-SQL language discussed in (Rozsnyai, Schiefer and Roth, 2009). SARI-SQL defines a language comparable to ANSI-SQL from a declarative perspective. The advantage of the languages based on an SQL-like syntax is that SQL is an industry standard that is widely used in business environments. Furthermore, many non-technical people are familiar with it (Rozsnyai, Schiefer, and Roth, 2009). An SQL based language is intuitive, and easier to learn for business users who are familiar with the notion of entities and queries for reporting purposes.

The BPEQL component provides a query engine that processes query statements formulated in our proposed query language. The query engine works as a translator by parsing and converting BPEQL query statements into JPQL (Java Persistence Query Language) statements. Once the queries are translated into JPQL statements, these are forwarded to the Event Data Warehouse component, which performs the query and returns the result back to the query engine. The JPQL serves as a suitable intermediary layer for accessing the metrics stored at the data warehouse.

The query engine uses the ANTLR runtime for parsing and translating the queries into JPQL. ANTLR (*ANother Tool for Language Recognition*) is an open source product that "provides a framework for constructing recognizers, interpreters, compilers and translators from grammatical descriptors" (Parr).

The BPEQL grammar is based upon a reduced version of the ANSI-SQL standard. Likewise, it incorporates new features to adapt the language for a business process domain.

The specification of the BPEQL grammar is as follows:

```
SELECT [AGGREGATE]
  (
 (*) | (id | name  | source
     | start_time  | end_time
     | turn_around | wait
     | change_over | processing
     | suspend)
  )
FROM (ACTIVITY | PROCESS | MODEL | MAP)
[WHERE condition]
```

SELECT Clause

The select clause specifies the attributes that will be selected in the output.

The optional AGGREGATE clause groups the result in just one row and applies the average function to the metric attributes.

FROM Clause

The FROM clause specifies the context domain from which the information will be retrieved. It can specify an ACTIVITY, PROCESS, MODEL or MAP.

WHERE Clause

The optional WHERE clause may specify either an ID or a NAME or both, provided the FROM clause references to an ACTIVITY, a PROCESS or a MODEL. If the FROM clause references a MAP, then only an ID can be specified at the WHERE clause.

Grammar Specification

The grammar definition is broken down into two main components, a lexer and a parser. The lexer is specified by a set of rules that defines the lexical analysis. The parser is a grammar specification that determines if an input is syntactically correct with respect to the formally defined grammar. This is also known as syntactical analysis. Furthermore, the proposed grammar features some semantic rules which make the recognizer also work as a translator.

The next figure illustrates how the BPEQL translator works internally by generating the parse-tree and evaluating the semantic rules at the parse tree nodes against the input statement. The translation is carried out on the following input query:

```
SELECT id, start_time, end_time
  FROM PROCESS
 WHERE NAME = 'ProcesTripOrder'
```

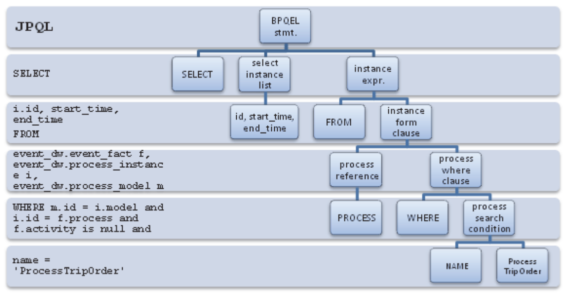


Figure 10: BPEQL syntactic tree.

The example above shows how the parse-tree generates the JPQL statement in pieces while processing their nodes. The translated BPEQL statement into JPQL is outlined below.

```
SELECT i.id, start_time, end_time
  FROM event_dw.event_fact f,
       event_dw.process_instance i,
       event_dw.process_model m
 WHERE m.id = i.model
   AND i.id = f.process
   AND f.activity is null
   AND name = 'ProcessTripOrder'
```

3 EVALUATION

The initial evaluation strategy to date has been based on a qualitative analysis in terms of effectiveness, completeness and usability.

The effectiveness is assessed by verifying that the framework addresses the success criteria, namely to be capable of monitoring and analysing the execution outcomes of business processes. Hence, the framework must meet the following research challenges:

1. Collect distributed event data from business processes executed on heterogeneous systems.
2. Unify the gathered event data in a unique central repository.
3. Identify and correlate subsequent events.
4. Generate metrics from the business process execution outcomes.
5. Query the structural and behavioural properties of business processes from the event repository.

The completeness attribute features the grade of expressiveness of the proposed query language. This qualitative metric is used to measure the coverage by the query of business processes execution information.

The usability attribute features the ease and simplicity of the query constructs with respect to the query expressiveness.

3.1 Prototype

An implementation of the framework presented in this paper has been created along with a set of tests to assess the proposed framework. The results of the test executions were used to evaluate the framework against the above quality attributes.

For capturing the events, three different instances of BPEL engines have been deployed on a local network. These engines recreate the business process scenario specified in the next section, and which is used for testing purposes. Obviously any system capable of outputting the event format information could be used instead of these test BPEL instances.

The BPEL vendor of choice is the Apache ODE (Orchestration Director Engine) 1.3.5 of the Apache Software Foundation. Every Apache ODE instance corresponds to a determined organizational unit, and under every unit is executed a particular BPEL process.

A specific plug-in (F4BPA-ODE) captures the business events produced by the Apache ODE servers. This plug-in is attached to every BPEL engine and uses the own Apache ODE API to access to the persisted data. Once the data is retrieved, the events are sent to the network in the BPAF extended format (F4BPA-BPAF), after being converted by the means of ETL processes.

The architecture of the prototype previously described is graphically depicted in Figure 11.

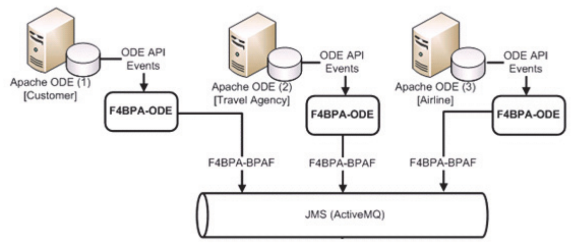


Figure 11: Architecture of the prototype on the event capturing side.

3.2 Sample Process Scenario

The event information managed by the framework must be enclosed in a business domain, thus a sample process scenario is needed for evaluating the framework.

This sample business process is based on a travel planner where the customers can book and order trips. The business process model is illustrated in Figure 12 in BPMN notation.

The business process is launched upon a plan trip requested action. The root process interacts with other sub-processes which are part of third party systems that represents the organizational boundaries of the business process.

Three different pools have been established, a Customer, a Travel Agency and an Airline, where each defines a different organization, and whose processes are part of the trip planning process.

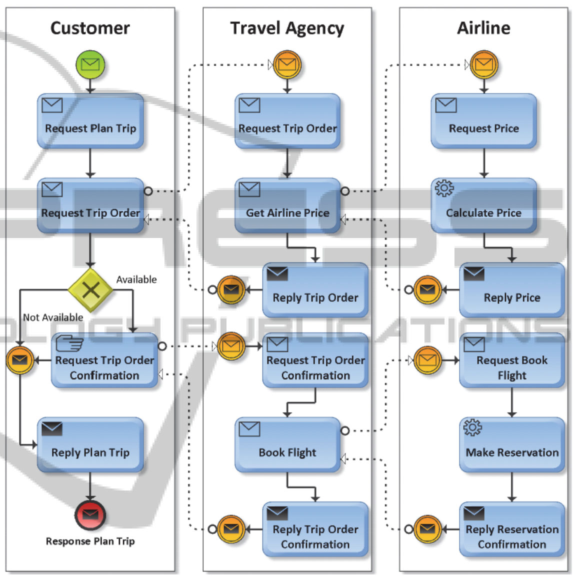


Figure 12: Sample Business Process in BPMN notation.

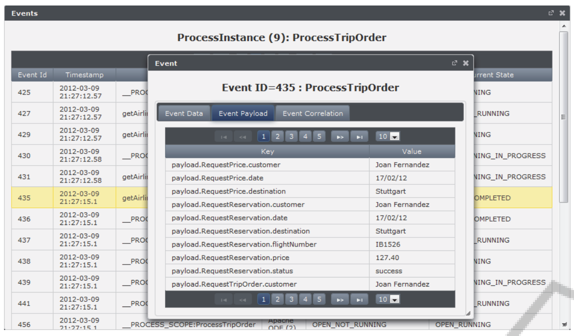
In order to simulate a distributed environment on a real test case, each sub-process has been implemented as BPEL processes which are executed in a separated BPEL instance. Likewise, there is a single BPEL instance per pool representing the system boundaries, while the BPEL engines are fully accessible throughout the network.

3.3 Tests

Several tests have been carried out over the framework aiming to produce a volume of event data large enough as to obtain fair results.

The storage of a large amount of event data, produced by the continuous execution of the business process, generates plenty of valuable information that enables analysts to gain insight into business performance.

Figure 13 is a screenshot taken from the framework. It illustrates how a list of events, for a particular process instance and related activities, are displayed on the screen. It also highlights the business data stored in the payload section of the selected event.



Events

ProcessInstance (9): ProcessTripOrder

Event Id	Timestamp				urrent State
425	2012-03-09 21:27:12.57	__PROC			NING
427	2012-03-09 21:27:12.57	getAirli			_RUNNING
429	2012-03-09 21:27:12.57	getAirli			NING
430	2012-03-09 21:27:12.58	__PROC			NING_IN_PROGRESS
431	2012-03-09 21:27:12.58	getAirli			NING_IN_PROGRESS
435	2012-03-09 21:27:15.1	getAirli			OMPLETED
436	2012-03-09 21:27:15.1	__PROC			OMPLETED
437	2012-03-09 21:27:15.1	__PROC			_RUNNING
438	2012-03-09 21:27:15.1	__PROC			NING
439	2012-03-09 21:27:15.1	__PROC			NING_IN_PROGRESS
441	2012-03-09 21:27:15.1	__PROC			_RUNNING
456	2012-03-09 21:27:15.1	__PROCESS_SCOPE:ProcessTripOrder	Apache ODE (2)	OPEN_NOT_RUNNING	OPEN_RUNNING

Event

Event ID=435 : ProcessTripOrder

Key	Value
payload.RequestPrice.customer	Joan Fernandez
payload.RequestPrice.date	17/02/12
payload.RequestPrice.destination	Stuttgart
payload.RequestReservation.customer	Joan Fernandez
payload.RequestReservation.date	17/02/12
payload.RequestReservation.destination	Stuttgart
payload.RequestReservation.flightNumber	IB1526
payload.RequestReservation.price	127.40
payload.RequestReservation.status	success
payload.RequestTripOrder.customer	Joan Fernandez

Figure 13: Screenshot of the process instance "ProcessTripOrder".

The event model allows drilling down the business process execution outcomes into multiple levels of detail, whereby it is possible to either know the waiting time of a determined activity or the overall execution time of a cross-organizational process.

The following table outlines a sample of the execution result of the activity `getAirlinePrice` associated to the `ProcessTripOrder` process.

Table 1: Execution results of the activity 'getAirlinePrice'.

Evt.Id	Timestamp	Activity	Previous State	Current State
22	2012-03-07 21:51:20.86	getAirlinePrice		OPEN_NOT_RUNNING
23	2012-03-07 21:51:20.86	getAirlinePrice	OPEN_NOT_RUNNING	OPEN_RUNNING
26	2012-03-07 21:51:20.86	getAirlinePrice	OPEN_RUNNING	OPEN_RUNNING_IN_PROGRESS
42	2012-03-07 21:51:22.07	getAirlinePrice	OPEN_RUNNING_IN_PROGRESS	CLOSED_COMPLETED

This information indicates that the activity was executed successfully without interruption of any kind, neither from a human interaction nor from activity suspension. This is extremely useful for business users to detect non-compliant situations, but it is not sufficient. Whilst the live data outlined above give an insight into the business process execution flow, they do not provide measurable information about business performance. Therefore, it is desirable to provide a fact table per process instance or activity.

Consequently, a dimensional model has been settled for this purpose. The Figure 14 shows the UML star diagram used for storing and accessing the behavioural information of process and activity instances.

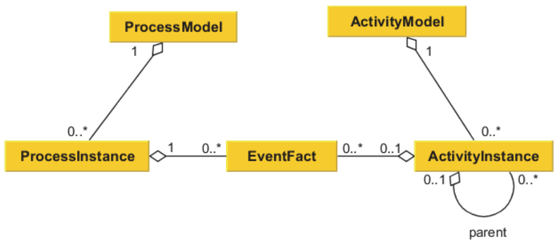


Figure 14: Star diagram.

As depicted at the above figure, this dimensional model allows the retrieval of any fact associated with any process or activity instance regardless of their nesting level. Likewise, it is possible to retrieve accumulative metrics for a determined model. This enables analysts to obtain information about the average execution times, rate failures, standard deviation, etc. for any part of a distributed business process.

The following table illustrates a set of metrics associated with the activity `getAirlinePrice` outlined in the Table 1. The information displayed states that the activity was executed in 1840 ms and it was not interrupted.

Table 2: Metrics of the activity 'getAirlinePrice'.

Start Time	End Time	Processing	Turnaround	Suspending	Wait	Change-over
2012-03-07 21:51:20.86	2012-03-07 21:51:22.07	1840	1840	0	0	0

The fact table provides valuable information for analysing the behaviour of organizational processes which are represented in single instances. However, what business users really often need is to know the average execution time for a particular business process or activity, and not only for a unique isolated instance. This is possible by grouping rows in the fact table and applying an aggregate function over the metrics. A filter around a set of processes or activities, which correspond to a specific model, will achieve the desired result.

3.4 Evaluation Results

The execution of test cases has shown that the prototype meets the research challenges. The prototype collected the data from the Apache ODE server, correlated and stored the business events in a central repository. Additionally, the metrics were generated successfully as per previous points.

Certainly, the framework has provided a knowledge base that enables analysts to track business processes, but it still requires evaluation of the proposed query language.

The initial analysis is focused on the expressive scope of BPEQL in relation to a business process domain. Namely, a scope has been defined in order to identify the level of detail of business processes that the language is able to retrieve information for.

Table 3: Expressiveness scope of BPEQL.

Expressive Scope	
Cross-organizational process (Map)	✓
Model	✓
Process	✓
Activity	✓
Sub-activity	
Event	

The BPEQL language has the capabilities of retrieving behavioural and structural information from any level of a business process, except from the event and sub-activity levels.

For instance, the language can construct queries such as “*What is the average execution time taken for the process ‘X’?*” or “*What is the suspending time for the activity ‘Y’?*”, but it cannot answer questions such as “*At what time the activity ‘Y’ failed for the last time?*” or “*What is the rate failure for the process ‘Z’?*” These last questions imply to drill down to the event level to identify a failed state.

In spite of not providing this functionality in the language, it can be easily extended to support this feature since this information is stored and managed by the framework.

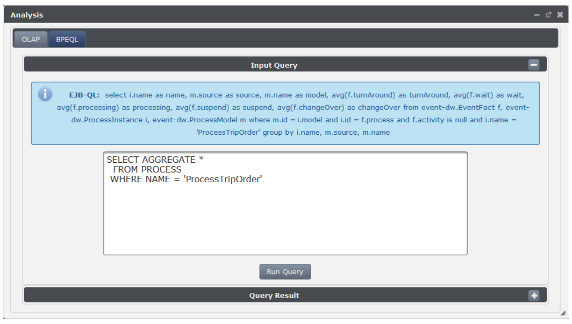
The measurable properties of business processes are also essential for identifying exceptional situations, detecting bottlenecks and discovering business opportunities.

The proposed query language may retrieve any metric presented in this paper, and also aggregate metrics over an evaluation function. This enables the framework to retrieve the average execution time of a determined process, activity or map.

The next figures are screenshots taken from the framework that illustrates how a BPEQL statement retrieves the average times for a particular process.

```
SELECT AGGREGATE *
  FROM PROCESS
 WHERE NAME = 'ProcessTripOrder'
```

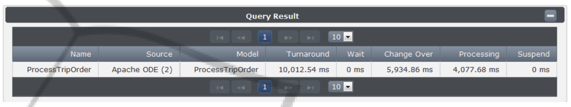
The usability of the language is pretty simple since it can query any nested level of a business process by just specifying the desired level on the FROM clause. Furthermore, it can also refer to a specific process or activity instance by filtering by an instance ID, or even grouping similar instances of a determined model in a simple manner by specifying the process name. In this regard, the



EJB-QL: select i.name as name, m.source as source, m.name as model, avg(f.turnAround) as turnAround, avg(f.wait) as wait, avg(f.processing) as processing, avg(f.suspend) as suspend, avg(f.changeOver) as changeOver from event-dw.EventFact f, event-dw.ProcessInstance i, event-dw.ProcessModel m where m.id = i.model and i.id = f.process and f.activity is null and i.name = 'ProcessTripOrder' group by i.name, m.source, m.name

```
SELECT AGGREGATE *
  FROM PROCESS
 WHERE NAME = 'ProcessTripOrder'
```

Figure 15: Screenshot of the BPEQL statement querying a particular process.



Name	Source	Model	Turnaround	Wait	Change Over	Processing	Suspend
ProcessTripOrder	Apache ODE (2)	ProcessTripOrder	10,012.54 ms	0 ms	5,934.86 ms	4,077.68 ms	0 ms

Figure 16: Screenshot of the BPEQL query result.

payload data play an important role in fetching a determined process. This would enable end-users to determine a specific instance by providing such data instead of dealing directly with instance identifiers which are complex to deal with. For example, from a business analyst perspective, it is more adequate to formulate questions such as “*What is the execution time of instance which order number is equal to ‘A525’?*” rather than “*What is the execution time of instance which identifier is ‘2834768’?*”. In terms of usability, these features will be added in future versions.

4 CONCLUSIONS AND FUTURE WORK

A proposed framework for monitoring and analysing business process performance has been presented in this paper. An event-based model was devised for supporting the data required for analysing business processes. The framework has adopted a centralized approach for monitoring the operational activities, collecting the business events and inferring knowledge from the gathered information. The system provides significant capabilities for analysing business process performance through the use of a query language developed for this purpose.

The framework, which was prototyped using an event-driven architecture, set out to tackle two main issues. First, to collect and integrate data originating from distributed heterogeneous enterprises systems. Secondly, to interpret and process event streams that are part of a cross-functional business process.

To overcome these issues, this work proposes a combination of event models that takes advantage of two complementary approaches. The iWISE (Costello, 2008) event model features cross-functional event sequences and permits the framework to be a non-BPEL exclusive dependent system. The BPAF model (WfMC, 2009), in contrast, provides powerful capabilities for enabling the analysis of business processes behaviour.

In the absence of standards for querying business processes, a query language has been proposed. The successful implementation and evaluation of the prototype has demonstrated that it is possible to monitor and query the structural and behavioural properties of business processes through the construct of a general purpose event model. Moreover, the business data can be unified and centralized seamlessly regardless of the underlying source systems.

In future works, the BPEQL grammar will be extended to improve its expressive power. Additionally, its usability will also be improved by incorporating references to business data without using identifiers, so that query construction will be significantly eased.

The framework is sufficiently flexible to incorporate easily the extensions mentioned above. There are plenty of possibilities for incorporating metrics and key performance indicators (KPI) without affecting the normal functionality of the existing system. Consequently, the BPEQL grammar can also be improved by incorporating these new elements gradually, thus improving the power and expressiveness of the language.

Other potential further research using the framework includes support for predictive analysis and integration with simulation and optimisation techniques and systems. This would pave the way for enabling the user to augment existing data with hypothetical information in order to perform what-if analysis over simulated scenarios.

Behavioural patterns recognition is another technique that could be leveraged by the proposed system in order to detect undesirable business process behaviours that are experienced frequently or on a continuous basis.

On a final note, it worth noting that event data centralization is not the only option to store and analyse distributed business data. Handling collaborative analytics on a fully distributed BI environment is a challenging task. Nonetheless, this work could be complemented with the federative approach, presented in (Rizzi, 2012), in terms of data warehousing and distributed query processing. The BI subsystem component presented in this paper could be attached to every operational business system along with their own local event repository. The event-based model presented herein represents the global schema proposed by Rizzi's approach. Thus, business process analytics could be carried out collaboratively in each organization independently by performing distributed queries along the collaborative network.

REFERENCES

Anicic, D., Fodor, P., Stojanovic, N., and Rudolph, S. (2011). EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. *WWW '11 Proceedings of the 20th international conference on World wide web.*

Balan, E., Milo, T., and Sterenzy, T. (2010). BP-Ex: A uniform query engine for Business Process. *EDBT '10 Proceedings of the 13th International Conference on Extending Database Technology.*

Behesti, S., Benatallah, S., Motahari-Nezhad, H., and Shakr, S. (2011). FPSPARQL: A Language for Querying Semi-Structured Business Process Execution Data. *UNSW-CSE-TR-1103, School of Computer Science and Engineering, University of New South Wales, Australia.*

Costello, C. (2008). *Incorporating Performance into Process Models to Support Business Activity Monitoring.* National Universisty of Ireland, Galway.

Kang, J., and Han, K. (2008). A Business Activity Monitoring System Supporting Real-Time Business Performance Management. *Convergence and Hybrid Information Technology, 2008. ICCIT '08.*, 473-478.

Parr, T. (n.d.). *ANTLR Parse Generator*. Retrieved 6 11, 2012, from http://www.antlr.org

Rizzi, S. (2012). Collaborative Business Intelligence. In M. Afaure, and E. Zimanyi (Ed.), *First European Summer School (eBISS 2011)* (pp. 186-205). Paris: Springer.

Rozsnyai, S., Schiefer, J., and Roth, H. (2009). SARI-SQL: Event Query Language for Event Analysis. *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing .*

Seufert, A., and Schiefer, J. (2005). Enhanced Business Intelligence - Supporting Business Processes with Real-Time Business Analytics. *Database and Expert Systems Applications*, (pp. 919 - 925). Copenhagen.

WfMC. (2009). *Workflow Management Coalition - Business Process Analytics Format Specification*. Retrieved February 8, 2012, from Workflow Management Coalition - Business Process Analytics Format Specification: http://www.wfmc.org/Download-document/Business-Process-Analytics-Format-R1.html

Zur Muehlen, M., and Shapiro, R. (2009). Business Process Analytics. *Handbook on Business Process Management, Vol. 2, Springer Verlag*.