

# Querying Business Processes\*

Catriel Beerl  
The Hebrew University  
cbeeri@cs.huji.ac.il

Anat Eyal  
Tel Aviv University  
anat@exanet.com

Simon Kamenkovich  
Tel Aviv University  
simonkm@cs.tau.ac.il

Tova Milo  
Tel Aviv University  
milo@cs.tau.ac.il

## ABSTRACT

We present in this paper BP-QL, a novel query language for querying business processes. The BP-QL language is based on an intuitive model of business processes, an abstraction of the emerging BPEL (Business Process Execution Language) standard. It allows users to query business processes visually, in a manner very analogous to how such processes are typically specified, and can be employed in a distributed setting, where process components may be provided by distinct providers (peers).

We describe here the query language as well as its underlying formal model. We consider the properties of the various language components and explain how they influenced the language design. In particular we distinguish features that can be efficiently supported, and those that incur a prohibitively high cost, or cannot be computed at all. We also present our implementation which complies with real life standards for business process specifications, XML, and Web services, and is used in the BP-QL system.

## 1. INTRODUCTION

A business process (BP for short) consists of a group of business activities undertaken by one or more organizations in pursuit of some particular goal. It usually depends upon various business functions for support, e.g. personnel, accounting, inventory, and interacts with other BPs/activities carried by the same or other organizations. Consequently, the software implementing such BPs typically operates in a cross-organization, distributed environment.

It is common practice to use XML for data exchange between BPs, and *Web services* for interaction with remote processes [34]. The recent BPEL standard (Business Process Execution Language [7], also identified as BPELWS or BPEL4WS), developed jointly by BEA Systems, IBM, and Microsoft, combines and replaces IBM's WebServices Flow Language (WSFL) [27] and Microsoft's XLANG [35]. It provides an XML-based language to describe not only the interface between the participants in a process, but also the *full operational logic* of the process and its *execution flow*.

Commercial vendors offer systems that allow to design BPEL specification via a visual interface, using a conceptual, intuitive

\*The research has been supported by the Israel Science Foundation.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.

VLDB '06, September 12-15, 2006, Seoul, Korea.

Copyright 2006 VLDB Endowment, ACM 1-59593-385-9/06/09

view of the process, as a graph of data and activity nodes, connected by control and data flow edges. Designs are automatically converted to BPEL specifications. These can be automatically compiled into executable code that implements the described BP [30].

Declarative BPEL specifications greatly simplify the task of software development for BPs. More interestingly from an information management perspective, they also provide an important new *mine of information*. Consider for instance a user who tries to understand how a particular travel agency operates. She may want to find answers to questions such as: *Can I get a price quote without giving first my credit card details? What should one do to confirm a purchase? What kind of credit services are used by the agency, directly or indirectly, (i.e. by the other processes it interacts with)?* Obviously, such queries are of great interest to both individual users and to organizations interested in using or analyzing BPs. Answering them is extremely hard (if not impossible) when the BP logic is coded in a complex program. It is potentially much easier given a *declarative specification* like BPEL. For an organization that has access to its own BPEL specifications, as well to those of cooperating organizations, the ability to answer such queries, in a possibly distributed environment, is of great practical potential.

To support such queries, one needs an adequate query language, and an efficient execution engine for it. To address this need, we present in this paper BP-QL, a new query language which allows for an intuitive formulation of queries on BP specifications, and query execution in a distributed cross-organization environment.

Before presenting our results, let us highlight briefly some of the challenges in querying BP specifications in general, and BPEL ones in particular.

**Flexible granularity** BP specifications may be abstractly viewed as a set of *nested* graphs, possibly with *recursion*: The graphs structure captures the execution flow of the process components; The nesting comes from the fact that the operations/services used in a process are not necessarily atomic and may have a complex internal structure (which may itself be represented by a graph); The recursion is due to the fact that a process may call itself indirectly, through calls it makes to other processes. Users may wish to ask *coarse-grain* queries that consider certain process components as black boxes and allow for high level abstraction, as well as *fine-grained* queries that “zoom-in” on all the process components, possibly recursively. *An adequate query language must thus allow users to query the processes at different, flexible, granularity levels.*

**Distribution** As mentioned above, BPs typically operates in a cross-organization, distributed environment where each peer holds a set of BPs and may provide (resp. use) services to (of) remote peers. If a service's internal flow has been defined in BPEL, and the service providers make this specification available to their cooperating organizations (say via a web service), *users may wish to zoom-in on*

these remote components as well to query the service specification.

**Paths extraction** When querying BPs, users may be interested in retrieving, as an answer, the qualifying *flow paths* (as for instance in the query “What should I do to confirm my purchase?”). As the number of relevant paths may be large (or even infinite in recursive processes) a major challenge is to provide the users with a compact finite representation of the (possible infinite) answer.

**Ease of querying** As mentioned above, the BPEL standard offers an XML-based language for describing the operational logic of a BP. Since a BPEL specification is essentially an XML document, a natural question is why not query it directly, using XQuery? A key observation is that the BPEL XML format is (1) very complex and (2) was designed with ease of *automatic code generation* in mind; however, it is extremely inconvenient for *querying*. To express even a very simple inquiry about a process execution flow, one needs to write a fairly complex XQuery query that performs an excessive number of joins. Furthermore, even if a more query-friendly XML representation for it had been chosen (as indeed is done internally in our implementation), XQuery, as is, would still not be adequate for the task: XQuery only returns document *elements*, but not *paths*, it does not support querying at *different levels of granularity*, and it does not offer tools for *controlling distributed querying*. Last but not least, querying an XML representation is much more difficult than querying directly a conceptual model. Essentially, ease of querying requires an intuitive, conceptual, data model, coupled with a matching, equally intuitive, query language.

The BP-QL query language presented in this paper addresses these issues. It is based on an intuitive model of BPs, an abstraction of the BPEL specification, along with a graphical user interface that allows for simple formulation of queries over this model. In a sense, it follows the same design principles that guided commercial vendors in the development of graphical editors for the *specification* of BPEL processes: it hides from the users the tedious BPEL XML details and allows for more natural query formulation. Indeed, we will see that the tight analogy between how BPs are specified in such editors and how they are graphically queried in BP-QL, facilitates intuitive querying. BP-QL also offers facilities for controlling granularity and distribution in query formulation and allows paths in query results.

At the core of the BP-QL language are *BP patterns* that allow users to describe the pattern of activities/data flow that are of interest. BP patterns are similar to the tree- and graph-patterns offered by existing query languages for XML [36] and graph-shaped data [15, 13, 31], but include two novel features designed to address the issues mentioned above. First, BP-QL supports navigation along *two axis*: (1) the standard *path-based axis*, that allows to navigate through, and query, paths in process graphs, and (2) a novel *zoom-in axis*, that allows to navigate (transitively) inside process components (local as well as remote ones) and query them at any depth of nesting. Second, paths are considered first class objects in BP-QL and can be retrieved, and represented compactly, even when involving activities performed on distinct peers.

Together, these features allow for simple formulation of queries on BPs. However, they make the evaluation of queries much more intricate than that of traditional XML/graph patterns. Indeed, some queries that can easily be evaluated on flat graphs/trees may become computationally expensive (or even undecidable) when nested graphs are concerned. To keep the evaluation of queries tractable, we had identified these problematic scenarios and carefully designed the language so that they are avoided, and polynomial-time query evaluation is guaranteed. Our analysis is based on modeling systems of processes and queries as *graph grammars*[21].

Observe that, in general, several modes of querying business processes are possible. One can query the specifications as *data* (e.g. “does the specification include a path from activity A to activity B”). One can also ask about patterns that may occur when the processes are *executed* (e.g. “can there be a run of the system where activity A is followed by activity B”). One can also *monitor* runs as processes execute, or pose queries on *logs* of past runs.

BP-QL is a query language for process *specifications*,<sup>1</sup> not about their *possible runs*. This is for two main reasons. First, querying the possible runs of a system is a *verification problem* [22] and is typically of very high complexity (from NP-hard for very simple specifications to undecidable in the general case [28]). Second, the analysis of runs requires a specification to have a well defined semantics. Unfortunately, BPEL is not based on a formal model [28]. To avoid these obstacles and guaranty complexity that is polynomial in the size of the data, BP-QL ignores the run-time semantics of certain BPEL constructs such as conditional execution and variable values and focuses on the given specification flow. We believe this approach offers a reasonable balance between expressibility and complexity. Note that querying of specifications in fact “approximates” the querying of runs (e.g. only specifications that contain two given activities may potentially have runs where both occur). Hence, even when full run verification is desired, BP-QL can be used as an efficient means to narrow the search space for the more costly, interpretation dependent, verification. It can also be used to select the process parts to be monitored at run time[32].

**Contributions** We now state the contributions of this paper.

1. We present BP-QL, a new graphical query language that allows for intuitive querying of process specifications, by offering a data model and an interface similar to those used for BPs *specification*. It allows to retrieve paths, and offers facilities for querying at different levels of granularity, and for controlling distributed querying.
2. We present a formal model for systems of processes, and for our query language on such systems, based on *graph grammars* [21]. This model allows to distinguish between query features that can be efficiently supported, and those that incur a prohibitively high cost, or cannot be computed at all. Using this model, we explain how to construct a finite and intuitive representation of the (possibly infinite) answers of queries in time polynomial in the size of the specifications.
3. Finally, we describe the system’s implementation, highlighting the main challenges faced and the solutions taken.

A first prototype of BP-QL was demonstrated in [5], where only a very high level view of the language was presented. The present paper provides a comprehensive description of the language, of its underlying formal model and of its implementation. The paper is organized as follows. Section 2 introduces BP-QL informally via a running example. The underlying formal model is presented in Section 3. The system implementation is described in Section 4. We conclude in Section 5, considering related and future work.

## 2. SYSTEM OVERVIEW

We present here an informal overview of BP-QL via a running example. To illustrate the features of BP-QL, we will consider a set of business processes (BPs) used by a consortium offering travel-related services. These include flight and hotel reservation, car rental, credit and accounting services. The processes, and their BPEL specifications, reside and operate on distinct peers. The specifications include the interactions between the various processes.

<sup>1</sup>A variant for monitoring and querying of logs is planned future research.



















