Developing Graph-Based Evaluation Methods for Process Querying

Riley Moher

Department of Mechanical and Industrial Engineering, University of Toronto, Ontario, Canada M5S 3G8

Abstract. Business Process Management (BPM) is a critical element of an enterprise's toolkit to control and understand processes that are essential to business operations. With evolving and increasingly digital processes, enterprises are faced with large process repositories of complex process models. Process model querying addresses these repository issues and can help enterprises to better leverage their repositories, reduce duplicated work, and streamline process management in the case of company mergers. While there has been excellent work in developing process querying frameworks, comparability is a major concern, with no real quantitative evaluation framework in place. Many process querying works already adopt a process-querying-as-graph-querying framework, so the evaluation methods of graph querying is a natural direction to guide development of a robust evaluation framework for process querying. This paper lays out the specific components of a graph-based process querying evaluation framework and criteria for them. Benchmark repositories, query test sets, and query metrics in recent graph querying work contain clear commonalities and valuable lessons for process querying. The application of these lessons guides future process querying work towards focused, robust, and more rapid development, and a clear place within enterprise BPM.

Key words: business process management, process querying, graph querying, process repository, enterprise modelling, information systems

Introduction

Business Process Management (BPM) is an incredibly important tool for enterprises, enabling increased process understanding, improved efficiency, and greater control and consistency. In order to capture the processes that are critical to business operations, enterprises build out a repository of business processes models, which could be in many formats, like BPMN, UML Activity Diagrams, or others. Especially in mature organizations with complex structures, these process repositories can become prohibitively large, and cumbersome to navigate. For instance, it would be useful to re-use operationally similar elements of existing process models to reduce the efforts of building process models from scratch. Enter process querying, a method of retrieving process models or parts of process models, according to some user input, to address these issues.

Process querying enables business users to specify formal instructions to retrieve relevant information from a process repository. Many methods of process querying al-

ready exist, varying in numerous components including query processing, input format, and storage components. Evaluating the design decisions of these components, however, proves a great challenge, due to the lack of any methodology or framework for the evaluation of process querying methods. This paper addresses this issue by taking lessons from the related and more mature field of graph querying. While work on process querying has adopted some of the methodologies of graph querying, it has not learned from the more mature and robust evaluation methods it contains.

When examining work on graph querying, it is very common to find an evaluation section containing quantitative results, usually presented in tabular format, comparing the author's approach to the work of various others on benchmark datasets. Conversely, this kind of evaluation is simply not possible for process querying, with no purpose-designed benchmark repositories, query test sets, or evaluation metrics in use. This paper's goal is to identify specific goals and criteria to close the gap between the evaluation framework of graph querying and that of process querying.

This paper covers a review of work on both process querying and graph querying, and a subsequent analysis and integration of the two as it relates to evaluation methodologies. Firstly, section 1 provides a detailed definition of process querying, including where process querying fits within the BPM environment, the significance of process-querying-as-graph-querying, and the role of enterprise knowledge models in process querying. Next, section 2 covers recent work in graph querying, detailing the various kinds of graph querying and their process querying analogues, along with a specification of graph querying evaluation components. Subsequently, section 3 covers the criteria of an effective evaluation methodology for process querying, with section 4 judging current reference process repositories according to these criteria. The implementation and validation of these criteria are discussed in section 5 as an opportunity for future work. Finally, section 6 provides some concluding remarks.

1 Process Querying

Process querying, being a part of an enterprise's BPM capabilities, needs to have a well-defined role in order for it to be properly executed and evaluated. This section defines process querying by explaining how it fits into an enterprise BPM system, explaining the process-querying-as-graph-querying paradigm, and explaining how both behavioral and operational semantics fit into a process querying framework.

1.1 Process Querying as Part of Business Process Management

In an enterprise BPM environment, many complex systems may be at play, with process querying being just one of these. In the work of Polyvyanyy et al [1], the authors define process querying as a set of methods for managing, filtering or manipulating repositories of process models, and the relationships between these models. This definition encompasses a broad area of work and, while more broad than the definition of process querying for this paper, it does provide insight to how process querying should be integrated with other BPM systems. To that end, Polyvyanyy et al integrate work that would

retrieve processes from event logs or from execution monitoring systems. In this paper, process querying does not necessitate process mining or execution monitoring work, in which process models are implicit. Rather, process querying is done only on the enterprise process repository of modelled processes. Moreover, the framework necessary to support interoperability between explicitly modelled processes and implicit forms of process, as in event logs, are outside of the scope of the process querying system, a point which will be elaborated on in subsection 1.3

In the broader enterprise BPM environment, many views of process exist, not only in terms of granularity, which could be modelled using nesting or drill-down approaches in visual models, but also in terms of tangibility. Process querying, for the purpose of this paper, is defined with the following points:

1. The retrieval of some set of process models or elements of them
2. From a repository of processes modelled in some modelling language or formalism
3. According to some business user's information need
4. Specified in some query language

Point (1) captures the fact that what is retrieved are not necessarily end-to-end processes, but may consist of pieces of multiple, disjointed, elements of process model(s). This distinction is necessary to facilitate more focused process querying, where specific sub-processes may be able to be re-used or for discovering patterns. Point (2) excludes approaches such as natural language process descriptions, execution logs, or real-time events, differentiating this definition of process querying from that of [1]. Point (3) is consistent with the goal of more general problems of search and information retrieval, indicating that actual relevance or performance of process querying needs to be answered in terms of original information needs, rather than the query that is the actual input into a process querying system. Lastly, point (4) indicates that this original information need must be expressed in some formal language; the query processing system must have a specified set of valid inputs. While this definition is not the most broad for process querying, it does not impose significant restrictions and, importantly, implies a more focused kind of process querying more suitable for evaluation and comparison.

Process querying frameworks at enterprises will have varying levels of maturity and integration with the other elements of an enterprise's BPM capabilities. As such, the evaluation model for process querying should not make any assumptions about, or requirements based on, the other BPM capabilities that may support process querying. For example, if a process querying framework includes querying over both execution logs and process models in a repository, an implicit requirement exists for mapping event-style process semantics to that of process models. The following two subsections will elaborate on this point by introducing graph querying as a model-independent analog to process querying, and detail the specifics of process semantics vs operational semantics in process querying.

1.2 Process Querying as Graph Querying

In many research problems, it's often useful to abstract a problem to a more general model, which tends to have more mature solutions, and allows for mapping of different

formalisms to the more abstract model. In the case of process querying, modelling process models in a repository as graphs has been done in numerous cases [2–4], across different modelling languages including UML activity diagrams, BPMN, and petri nets, for exactly this reason. Surveys of this work already exist [1, 4], with Wang et al [4] in particular containing an excellent summary of process querying works including their own representation of process-querying-as-graph-querying. Wang et al's representation of process models, as compared to that of other work, contains the same fundamental idea of representing a process repository as a set of typed nodes in a graph, however, it goes further than other work like VMQL [2, 5, 6] and BPMN-Q [3] by including a greater range of objects, including data objects, resources, and actors. In this paradigm, a business process model is defined as a tuple as follows:

$$PM = \{V, E, L, l, D, Wt, Rd, R, A\} \tag{1}$$

At the most basic level, V is the set of nodes in the graph, each having its own type defined by a typing function, $V \rightarrow \{task, gateway, state, ...\}$. The set of flow relations is defined by $E \subseteq V \times V$, showing the ordering of tasks, $T \subseteq V$, within a process. As an example of the flexibility afforded by this approach, consider the following definition:

$$P_{CF} = \{T, E\} \tag{2}$$

Definition (2) defines P_{CF}, the control flow perspective of process model PM, whereby only the tasks and their orderings are included. This kind of perspective enables a more specific operational-semantics focused view of a process model, consistent with some modelling approaches like petri nets. For the complete definition of this representation, the reader is directed to [4]. For the purpose of this paper, Wang et al's representation will serve as a guide, since it contains the most complete graph representation of all the work surveyed for this paper, though it should be noted that this work is meant to be applicable to process-querying-as-graph-querying representations in general.

1.3 Process & Operational Semantics: Enterprise Knowledge Models

Business processes are understood from a variety of business perspectives, and exist in different modalities. Enterprise models like enterprise ontologies are used to provide a formal specification of business knowledge and concepts, allowing for increased interoperability between an enterprise's different models and systems. In this case, knowledge models are being utilized primarily for operational semantics, which may be tied to process models in a repository, but the semantics of process execution or control flow are not necessarily captured in the knowledge model. The role of ontologies in process querying can be separated into two categories: knowledge model-based process querying, and knowledge model-integrated BPM.

In knowledge model-based process querying, the same motivation provided for utilizing graphs as a representation of process model holds for using ontologies, semantic web services, or other methods of knowledge representation. This approach involves the mapping of some process knowledge model(s) to some business process modelling notation(s), with the knowledge model providing the basis for querying. Once this mapping has been created, the querying functionality is a direct consequence of this mapping, with the querying language being one that is based on the knowledge model. One

recent example of this is [7], where the authors develop a tool called PM2ONTO to map BPMN v2.0 diagrams to OWL ontologies to then be queried using SPARQL. Similarly to graph-based methods of process querying, however, this approach also lacks any robust evaluation methodology.

Knowledge model-integrated BPM takes a similar approach to the previous category, in that they will map elements of process model(s) to knowledge model(s), but the purpose and kind of mappings in this case is more broad. In this case, knowledge models are utilized to enable interoperability between the different systems of a BPM environment, on top of querying functionality. For example, a BPM environment may have a process querying system that is graph-based, and additionally features an enterprise ontology to specify equivalent concepts in process models. In this example, the knowledge model enhances operational semantics so a greater range of processes can be queried, with the actual querying functionality coming from the graph-based system. Knowledge model-integrated BPM may also provide process-semantic integration through a model like an upper level process ontology, like the Process Specification Language (PSL) [8], or as a metamodel for a specific process modelling language, as in [9]. Process semantics enables interoperability between different notions of processes as they exist in separate systems, for example, integrating execution logs with BPMN models.

Integrating enterprise knowledge models with a process querying system can be done in numerous ways, and may lead to a number of different mappings between software systems. Great care should then be taken to ensure complexity of these mappings and semantic integration does not become prohibitive to BPM. For example, SUPER [10], utilizes a business process modelling ontology to translate concepts from an upper level process ontology to business process models including BPMN. In a stack like SUPER (see figure 1), extending this complex interconnected stack with functionality like process querying will take a great deal of effort, only further exacerbated by a lack of evaluation methods to compare possible approaches.

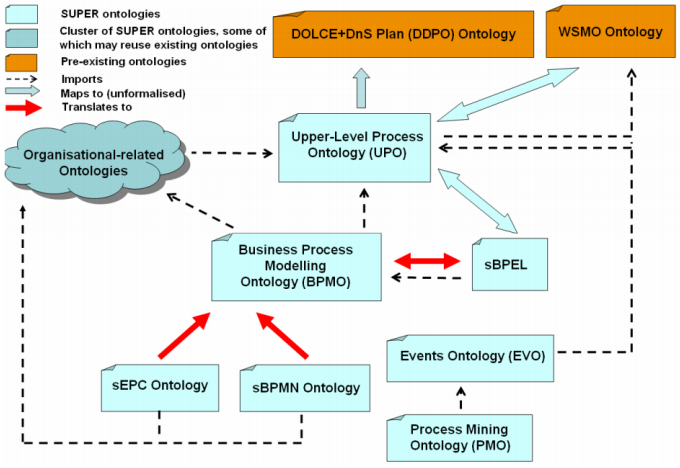

Fig. 1. Process Modelling Ontology Stack in SUPER (taken from [10])

2 Graph Querying

Graph data is a widely-used representation of many kinds of data, from molecular structures to social networks, graph data is a useful data structure in many domains. With a diverse set of large and complex graph data, applications require an efficient method of querying these graph databases. In this section, several kinds of graph querying methodologies will be detailed, with an explanation for where process querying fits in. Finally, graph querying evaluation methodologies will be presented to lay the groundwork for section 3, on their evaluation.

2.1 Types of Graph Querying

For the scope of this paper, methods of graph querying is kept at a relatively high level, going into more detail where relevant to process querying. For a more complete overview of graph querying methodologies, see [11, 12] Many different taxonomies of graph querying methodologies exist, but for the sake of simplicity, this paper will use the three categories as described in [11]; subgraph query processing, supergraph query processing, and similarity query processing.

Subgraph Query Processing Subgraph query processing is perhaps the most common form of graph querying, whereby a query graph returns all graphs in the graph data such that the original query is a subgraph of the returned graph(s). In the context of process querying, this kind of graph query is a very common use case, finding some segment of a process model that are common across various process models in a repository. For instance, this kind of query corresponds to the question ”What are all the processes that involve credit checks?”, where the input query graph is a sequence of activities representing the execution of a credit check. Formally, given a process repository R and a process query q, find all processes p in R such that q is a sub-graph of p. These kinds of process queries are useful to find the impacts of some changes or disruptions in an enterprise, going with the already provided example, a bank may wish to find all processes which will be impacted by a credit checking service outage.

Supergraph Query Processing Supergraph query processing is the converse of subgraph query processing, whereby we wish to find those graphs to which the query graph is a supergraph. What this means in an intuitive sense is to find all the find the pieces of an input query graph that exist within the graph database. In the context of process querying, this has a less intuitive application, of finding all the subprocesses of the query process that exist within the process repository. Formally, given a process repository R and a process query q, find all processes p in R such that q is a supergraph of p. It is more difficult to see the usefulness in this case, where a business requires the identification of sub-processes that exactly match those already modelled in the query graph. For example, assume the input query is a large and complex process model. In the previous kind of query, the user would anticipate the results, looking for a specific kind of process to see where it already exists in the repository. In this kind of query, the result in unanticipated, the primary purpose would be finding non-trivial subprocesses within the modelled graph that exist in the repository. Concretely, one could imagine

a process practitioner in a large organization inputting their complex process and discovering another process model which contains similar elements, perhaps leading them to other practitioners who deal with similar processes. This could enable re-use and increased collaboration for the case of querys which produce a large number of results, thereby identifying common processes.

Similarity Query Processing Similarity query processing can itself be categorized even further, as it has many different approaches and applications, even as a supporting component of the graph querying types previously mentioned. For the sake of this paper, the explanation of similarity query processing is kept at a high-level. Similarity query processing is most useful in applications where graphs may contain incomplete or semantically heterogeneous elements. For example, interesting problems in querying on knowledge graphs try to solve the problem of incomplete knowledge, through techniques like query embedding [13, 14]. In the case of process querying, process repositories will not always contain precise matches to queries, with different terminologies or structures utilized for semantically comparable processes. As pointed out in Wang et al's description of the characteristics of business process models [4], business process models often contain identical tasks with different labels, requiring some measure of label similarity to be implemented. The actual mechanics of how similarity is measured could be implemented in many ways, including utilizing enterprise and process semantics, as mentioned in section 1.3. In order to compare different approaches to problems like inconsistent task naming, evaluation methods similar to those used in recent graph querying work [13, 14] are needed.

2.2 Graph Querying Evaluation

In graph querying work, evaluation sections usually contain formal specifications of their evaluation framework, commonly with tables comparing various methodologies across dataset(s). In general, this kind of evaluation framework is comprised of 3 main elements:

1. Benchmark dataset(s) - typically a large dataset with very generally applicable data.
2. Query test-set - the set of queries used as the input for measuring the evaluation metric(s). Usually this is a set of queries created in-line with motivating scenarios.
3. Evaluation metric(s) - the choice of measurement for performance of results retrieved according to query test-set. These can be more general accuracy metrics or can consider rankings, metrics of information retrieval are often utilized.

Especially due to standardized benchmark datasets, this kind of evaluation framework is simple to draw conclusions from and enables rapid developments in future work. To demonstrate this evaluation framework, the benchmark dataset and evaluation metrics (query test set was excluded from this table as query templates are not expressible in a compact form suitable for a table) of 5 recent and impactful papers in the graph querying domain recent are presented in table 1. These papers were selected based on ranking results from Harzing's Publish or Perish [1], limited to papers published in 2017 or later.

[1] `https://harzing.com/resources/publish-or-perish`

Publication	Benchmark Dataset(s)	Evaluation Metric(s)
QUERY2BOX [13]	FB15k, FB15k-237, NELL995	Hits @ 3, Mean Reciprocal Rank
GQE [14]	Drug Interactions, Reddit Dynamics	Area under ROC Curve, Average Percentile Rank
Faithful Embeddings for Knowledge Base Queries [15]	FB15k, FB15k-237, NELL995	Hits @ 3, Mean Reciprocal Rank
Keyword Search on RDF Graphs [16]	DBPedia + QALD-6, Freebase + Free917	Prescision, Recall, F-1 Score
Graph based model for information retrieval using a stochastic local search [17]	CACM Collection	Precision, Recall

Table 1. Evaluation components of notable recent graph querying work

From this table, many overlapping elements can be observed, as well as the general conclusion that these components are simple to access and publicly available, as in the case of the benchmark datasets, and well-defined and simple to compute, in the case of the evaluation metrics. Datasets like FB15k, which is a subset of the Freebase general knowledge base containing common entities applicable for general knowledge graph problems are widely applicable datasets which contain a diverse set of general knowledge for evaluation. Publicly available datasets with a history of being used to test performance in different methodologies drive research forwards by providing straightforward comparability through quantitative results. In the context of process querying, having a consistent quantitative metric-based framework would provide a more focused direction for future research, through a more straightforward method of comparing approaches.

3 Evaluation Framework Criteria

Business process repositories are often very specialized and enterprise-specific datasets, so it is not a straightforward process to simply construct an evaluation framework to be generally applicable. Thus, it is useful to articulate the criteria for an effective process querying evaluation framework. In this section, three different criteria will be explained with regards to each of the three components of the evaluation framework. Subsequently, existing reference process repositories will be qualitatively evaluated according to these criteria, and challenges of designing an effective repository will be addressed.

3.1 Modularity

Process models can include a wide range of detail and different levels of granularity. As such, process querying techniques may place their focus on different elements of process models. For example, techniques may or may not include retrieval of data objects, or, in a more abstract sense, techniques which are more suited for representations like petri nets, may have a greater focus on behavioral semantics and little attention paid to operational semantics. As such, a successful evaluation framework should be modular to accommodate these different levels of expressiveness and/or different focuses in queries.

In terms of a benchmark repository, this means deliberate module organization. Even if a benchmark repository is assumed to be well-curated and general, having a modular structure allows greater flexibility in testing scenarios. Modules could be dataflow-focused, featuring graphs that have a large variety of data objects, or could focus on complex and diverse workflow patterns, as two examples.

The structure of the query test set should be similarly structured, with methodical organization. This not only improves flexibility, but also provides greater transparency in results. For researchers interested in a particular kind of process modelling, emphasizing workflow for example, workflow query module performance could be prioritized. Query modules could be developed in a very general sense and selectively applied depending on the motivation and expressiveness desired. In the case of work more focused on the query expressiveness, modules could be added to demonstrate the capabilities of their work vs other's existing work which may not be able to even process these queries.

In terms of query metrics, modularity is less applicable, since evaluation should be consistently applicable on any query results that are in an ordered list. However, conceptualizing query metrics as 'mini-modules' forces a focus on multiple metrics, and expands possibilities for those that may become complacent with a handful of metrics.

3.2 Model Agnosticism

Model agnosticism is an important criteria to mention, since it may be overlooked or viewed as trivial when the underlying representation is graph-based, something that should be compatible with many modelling languages. However, compatibility does not imply comparability in this case. If an evaluation framework is designed entirely with one modelling formalism, it will be biased to the concepts and level of expressiveness present in it. For example, a framework developed using petri nets exclusively, even if it is graph-based, will not have any prioritization for querying of data objects. Designing a model-agnostic evaluation framework will enable comparability between methodologies designed for different modelling formalisms and support more business process repositories.

For a benchmark repository, model agnosticism is, conceptually, simple to design. The benchmark repository should be comprised of process models captured in different modelling formalisms. The correct balance, however, and co-ordination of model diversity with modules will need to be approached with care. Designing the repository will likely need to be done in an iterative way, comparing performance across differ-

ent modelling formalisms to check for significant performance differences until some difference threshold is reached.

In terms of the query test set, care should be taken so as not to design queries around the capabilities of one langauge. For example, if all test queries involve some form of messaging, BPMN models will be matched while UML activity diagram models will not. This criteria for the query test set should be integrated into the previously mentioned modular design.

For query metrics, the judging of this criteria is along similar lines to that of the query test set. Evaluation metrics should be designed around the query test set, and not favour any set of test queries that are more applicable to any one modelling formalism than the other.

3.3 Generalization

Generalization is a critical goal in designing many kinds of systems, and a process querying framework is no different. Just as an evaluation framework should not be biased towards any kind of modelling formalism, it should similarly not be biased to a particular industry or business domain.

What generalization means in terms of a benchmark repository is a balance between enterprise domains. Similarly to model agnosticism, finding the correct balance will take careful iteration, something that will be discussed further in section 5. Generalization in this case enables testing across domains and allows for enhanced collaboration and increased progress as process practitioners and researchers across domains can work towards process querying framework development.

The query test set should also be designed with operational semantics in mind, so as not to bias towards concepts more prevalent in one domain than another. What this means is a more general set of concepts included in queries that are relevant in many businesses. For example, test queries that feature mostly tasks in the banking industry will not be applicable in general.

In terms of metrics, generalization in an evaluation framework can already be observed in graph querying literature. Works previously mentioned [13–17] already utilize similar metrics like mean reciprocal rank or hits at k. Validation will still be required in this case, but guidance can be more directly used from graph querying literature in this case.

4 Evaluation of Existing Repositories

Benchmark process repositories serve as a starting point for the development of a process querying evaluation framework, if current reference process repositories are suitable, the future work could be accelerated. When it comes to reference repositories, few are available for the BPM community to utilize. In this paper, four repositories are covered: the SAP reference model [18], the BPM Academic Initiative Model Collection (BPMAI) [19], Camunda BPMN for research [1], and RePROSitory [20].

[1] `https://github.com/camunda/bpmn-for-research`

4.1 SAP Reference Model

The SAP reference model is a database of more than 9000 event-driven process chain (EPC) based process models, originally designed to be SAP's documentation for their enterprise resource planning (ERP) system. It is an older example of a reference repository, having been in use since at least 1998, that has been referenced in many research papers. Firstly considering the criteria of modularity, the SAP reference model contains many branches including asset accounting, customer service, and recruitment. While these branches do fit a modular structure in terms of operational semantics, there is no modularity with respect to actual process model characteristics. Furthermore, an analysis of the errors within EPC models of the SAP reference model [21] reveals that only 604 of 9844 models were valid EPCS with at least one start event and one function. This work also reveals that measures like average number of events, number of models, and number of errors present are inconsistent across the branches. With these inconsistencies, and at least 5.6% of these EPC models containing errors [21], the SAP reference model does not have a suitable modular design.

In terms of model agnosticism, the reference model is exclusively captured in EPC models, meaning the SAP reference model alone would not be suitable. For generalization, the conclusion is very similar to that of modularity. With branches being imbalanced, for example retail contains 1 proper EPC model while treasury contains 48, the SAP reference model is unlikely to generalize well. Lastly, accessibility and availability is a large concern of the SAP reference model, with a history of only somewhat public availability [21] and SAP having removed all public links to the model, no longer maintaining or utilizing it. Based on the defined criteria then, the SAP reference model is not a suitable benchmark repository in its current state.

4.2 Business Process Management Academic Initiative and Camunda BPMN for Research

The next process repositories to be considered are similar and will be discussed together, BPMAI and Camunda BPMN for research. Both repositories are crowd-sourced models coming from students who are utilizing process modelling tools in an academic scenario. In the case of BPMAI, the models come from students who have modelled their processes for course work, research, etc, in the Signavio process modelling tool [1]. For the Camunda repository, models were sourced from students of Camunda BPMN training sessions who were instructed to model BPMN diagrams based on written process descriptions, as exercises. In terms of modularity, neither fulfill this criteria, each for slightly different reasons. The BPMAI repository does not have any organization beyond metadata of process models which contains information like modelling language, revision number, and model name. For BPMAI, categorization of the repository into modules would need to be done through a method like unsupervised learning, the results of which may or may not lead to suitable modules. Camunda on the other hand, has a very clear structure, process models are organized by exercise number and the corresponding natural language process descriptions given. This organization leads to

[1] `https://www.signavio.com/bpm-academic-initiative/`

a significant number of similar (nearly duplicated) process models, but an extremely low diversity of models. The Camunda repository could be a good candidate for work in understanding how process models are conceptualized and created, but it is certainly not a good representation of an enterprise repository.

In terms of model agnosticism, the BPMAI repository contains 14 different modelling languages, while Camunda's repository is exclusively BPMN. While BPMAI does contain many different modelling languages, Signavio is a tool with a BPMN focus, so the vast majority of models are BPMN based. For generalization, Camunda is certainly not successful in this criteria as models are based on such a small sample of contrived descriptions. BPMAI, on the other hand, would need to be more quantitatively assessed to verify the subject matter covered in each model, through some kind of unsupervised learning task again, which may or may not produce promising results. Finally, with both repositories being crowd-sourced, the lack of validation leads to model quality concerns. In the rePROSitory paper [20], an analysis of both these repositories revealed that approximately 14% of the models contain syntax issues that do not conform with BPMN standards. Based on all these reasons, the BPMAI and Camunda repositories would not be suitable benchmark repositories without significant work in validation and re-engineering.

4.3 rePROSitory

The last and most recent repository to be considered is rePROSitory [20], established in 2019 as an attempt to implement open science principles [22] in the BPM community, seeking to make a widely accessible, high quality repository of BPMN models. The 563 (as of writing this paper) models contained within rePROSitory come from BPM-related work in established conference and journals, so issues of quality are far less likely to occur than with the other repositories assessed thus far. In terms of modularity, rePROSitory does not feature an explicit module structure but rather relies on a search functionality which is close to process querying, with a narrower scope than has been defined in this paper. This search function allows users to filter BPMN models in the repository according to various metrics, like diameter, mean nesting depth, or number of activities, and by other publication-related metadata. This metadata includes publication year, conference of origin, modelling tool used, and many more categories. This rich tagging would aid greatly in developing modules focused more on process semantics or complexity, and may also address modules of operational semantics. In terms of generalization, the metadata also contains an application domain attribute including domains such as blockchain, data migration, and healthcare. Actual balance between these domains and appropriate diversity would need to verified more quantitatively, however, the presence of this rich metadata is promising. The largest drawback in terms of the evaluation criteria is model agnosticism, as rePROSitory consists exclusively of BPMN models. While rePROSitory may not be immediately ready to be implemented as a benchmark repository, its design philosophy and rich metadata, are very promising.

5 Future Work: Implementation and Validation

While this paper has presented criteria for graph-based process querying evaluation, the actual validation and implementation of these criteria is the immediate opportunity for future work. Design and validation of a process querying evaluation framework aligning with the criteria outlined in this paper constitutes a great amount of work. This section will detail the work associated with the implementation and validation of all of the outlined criteria for the repository, query test set, and metrics, respectively.

5.1 Repository

The criteria for a successful enterprise repository is conceptually simple; create a repository with a good balance of models from different domains, modelling languages, and with a modular design. However, realizing these criteria will require a great degree of data curation and iterative validation. This is where descriptive graph statistics, like those used within rePROSitory [20], could serve as a guide during iteration on same starting repository/ies. Holding aside the fact of generalization to different domains, this work could begin with a case study of a real, mature, and large-scale enterprise repository by creating a graph-based representation of it, using a definition similar to that of [4], as has been used in this paper. Exploratory data analysis on this repository could then set a threshold for metrics that measure graph quality, complexity and diversity. The next step could be comparing these statistics with that of rePROSitory [20], to quantify differences between real enterprise data and that of academic models. If differences are within an acceptable threshold, then papers from rePROSitory could also be used as reference, especially since process models within academia will contain far more documentation. Assuming a good degree of quality and diversity of processes according to exploratory statistics, the next issue to address would be model agnosticism and generalization. Both could be solved by performing this case study process on more enterprises, however, this would be a long and arduous process involving many stakeholders. An alternative to address a lack of model diversity would be to generate new models from existing process descriptions. Process description sourcing is simple in the case of academic work, but on the enterprise side, process descriptions may not always be present or of high quality due to poor documentation, requiring stakeholder interviews or other methods like process mining to generate new process models.

5.2 Query Test Set

Finding a successful query test set will need to be done as part of the iterative design process, checked against different versions of the repository as they progress. Firstly, a comprehensive literature review of the kinds of queries used in graph querying papers would need to be compiled to understand the morphology of these test queries and understanding what seems to be most common. From here, the kinds of queries utilized in graph querying would need to be adapted to work with process querying methodology, by formulating test queries and seeing if meaningful results are produced on the benchmark repository. Based on these results, modifications could be made both to the query test set and to the repository until results become more consistent and reach a standard of quality.

5.3 Metrics

Creating the right set of metrics would take a very similar approach to that of the query test set. Conducting a review of graph querying work with a focus on querying metrics, the various metrics could be compiled into a tabular format that would inform initial attempts at metrics for a process querying framework. Along with the query test set and benchmark repository, this would be developed in an iterative fashion. Retrieving correct and complete results for process queries is a specialized task, which may require more specialized metrics than those that work well graph querying on general knowledge. Taking this into account, subject matter experts and/or process practitioners could be used to assess which metrics produce the best results for the given repository and test queries.

6 Conclusion

Current work in process querying lacks a consistent and quantitative evaluation framework, with comparability and validation suffering as a result. In this paper, process querying was specifically defined with respect to its place in an enterprise BPM environment, the role enterprise knowledge models play, and was formalized as a form of graph querying. Process-querying-as-graph-querying is an important paradigm already being recognized in the process querying community, with graph querying methodologies already benefiting the query processing aspect of process querying work. However, the more mature and quantitative evaluation methodologies of graph querying have not been similarly adopted. The three key components of graph querying evaluation, benchmark dataset(s), a query test set and evaluation metric(s), were used as a guide for the components of a process querying evaluation framework. This evaluation framework is to be judged based on three well-defined criteria, modularity, model agnosticism, and generalization. Based on preliminary qualitative analysis, current reference process repositories show quality issues, poor modularity, and biases towards process models, though a more recent development, rePROSitory, shows promise in terms of modularity, generalization, quality, and accessibility. Future work includes the implementation and validation of this framework, an iterative process that will require significant efforts, but has been outlined in this paper. This work and its future developments will enable more rapid progress and increased understanding as process querying cements itself as an integral component of enterprise BPM.

References

1. Artem Polyvyanyy, Chun Ouyang, Alistair Barros, and Wil MP van der Aalst. Process querying: Enabling business intelligence through query-based process analytics. *Decision Support Systems*, 100:41–56, 2017.
2. Harald Störrle and Vlad Acretoaie. Querying business process models with vmql. In *Proceedings of the 5th ACM SIGCHI Annual International Workshop on Behaviour Modelling-Foundations and Applications*, pages 1–10, 2013.

3. Ahmed Awad. Bpmn-q: A language to query business processes. *Enterprise modelling and information systems architectures–concepts and applications*, 2007.
4. Jianmin Wang, Tao Jin, Raymond K Wong, and Lijie Wen. Querying business process model repositories. *World Wide Web*, 17(3):427–454, 2014.
5. Harald Störrle. Vmql: A visual language for ad-hoc model querying. *Journal of Visual Languages & Computing*, 22(1):3–29, 2011.
6. Harald Storrle. Vmql: A generic visual model query language. In *2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 199–206. IEEE, 2009.
7. Lukas Riehl Figueiredo and Hilda Carvalho de Oliveira. Automatic mapping of business process models for ontologies with an associated query system. In *International Conference on Enterprise Information Systems*, pages 215–238. Springer, 2018.
8. Michael Grüninger. Ontology of the process specification language. In *Handbook on ontologies*, pages 575–592. Springer, 2004.
9. Emilio M Sanfilippo, Stefano Borgo, and Claudio Masolo. Events and activities: Is there an ontology behind bpmn? In *FOIS*, pages 147–156, 2014.
10. Deliverable 1.1: Business process ontology framework. Project IST 026850 SUPER, 2007.
11. Yiping Ke, James Cheng, and Jeffrey Xu Yu. Querying large graph databases. In *International Conference on Database Systems for Advanced Applications*, pages 487–488. Springer, 2010.
12. Angela Bonifati, George Fletcher, Hannes Voigt, and Nikolay Yakovets. Querying graphs. *Synthesis Lectures on Data Management*, 10(3):1–184, 2018.
13. Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*, 2020.
14. William L Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. *arXiv preprint arXiv:1806.01445*, 2018.
15. Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. Guessing what's plausible but remembering what's true: Accurate neural reasoning for question-answering. *arXiv preprint arXiv:2004.03658*, 2020.
16. Shuo Han, Lei Zou, Jeffery Xu Yu, and Dongyan Zhao. Keyword search on rdf graphs-a query graph assembly approach. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 227–236, 2017.
17. Sidali Hocine Farhi and Dalila Boughaci. Graph based model for information retrieval using a stochastic local search. *Pattern Recognition Letters*, 105:234–239, 2018.
18. T Curran, Gerhard Keller, and Andrew Ladd. Sap r/3 business blueprint: Understanding the business process reference model. 1998.
19. Matthias Kunze, Alexander Luebbe, Matthias Weidlich, and Mathias Weske. Towards understanding process modeling–the case of the bpm academic initiative. In *International workshop on business process modeling notation*, pages 44–58. Springer, 2011.
20. Flavio Corradini, Fabrizio Fornari, Andrea Polini, Barbara Re, and Francesco Tiezzi. Reprository: a repository platform for sharing business process models. *BPM (PhD/Demos)*, 2420:149–153, 2019.
21. Jan Mendling, Michael Moser, Gustaf Neumann, HMW Verbeek, Boudewijn F Van Dongen, and Wil MP van der Aalst. Faulty epcs in the sap reference model. In *International Conference on Business Process Management*, pages 451–457. Springer, 2006.
22. Michael Woelfle, Piero Olliaro, and Matthew H Todd. Open science is a research accelerator. *Nature chemistry*, 3(10):745–748, 2011.