



From Complexity to Insight: Querying Large Business Process Models to Improve Quality

Kurt E. Madsen^(✉)

MetaTech, Inc., Tampa, FL, USA
kmadsen@metatech.us

Abstract. This industry study presents work in querying manufacturing processes using a portal to navigate query results in the broader context of enterprise architecture. The approach addresses the problem of helping stakeholders (e.g., management, marketing, engineering, operations, and finance) understand complex BPM models. Stakeholders approach models from different viewpoints and seek different views. Gleaning insight into improving model quality is challenging when models are large and complex. The focus of this work was on process models only, not process execution, because many legacy organizations have done initial BPM modeling but do not have BPM systems in production or have yet to realize the benefits of coupling log mining with incremental model refinement. The use cases presented address the complexity of multi-year, process models used by ~10,000 workers globally to develop new products. These models were queried to find quality issues, isolate stakeholder data flows, and migrate BPMN [1] activities to cloud-based, micro-services. The approach presented creates filtered process views, which serve as starting points for stakeholders to navigate interconnected models within TOGAF [2] enterprise architecture models. The process querying lifecycle—query, manipulate, and transform—was also applied to other enterprise models. Lessons learned from introducing BPM into a legacy organization, model refinement, limitations of research, and open problems are summarized.

Keywords: Process querying · Business process management · Business intelligence · Process compliance · Process standardization · Architecture portal

1 Introduction

This paper examines the question of how to query, filter (i.e., manipulate), and transform large, complex process models to gain insight into improving model quality. The focus is on process models only; process execution and log mining are out of scope, as many of the world's largest organizations have fragmented process management efforts, in various stages of maturity, scattered across disconnected departments. There is value in applying process querying to business process models without considering downstream process execution.

The query method presented was tested on enterprise architecture models in automotive manufacturing. The development lifecycle for new vehicles – from initial-marketing-concepts through to ready-for-mass-production – spans years. At any given time, there are dozens of concurrent vehicle programs in progress globally. The lifecycle was modeled as one process with variants by program scale and vehicle model. The process models shared 700+ different activity types (both sub-processes and tasks), instantiated as 4,000+ activity instances, interconnected via 7,000+ workflows, performed by 10,000+ process workers, assuming two dozen roles.

Modeling was performed in OpenText ProVision [3], a commercial, enterprise architecture modeling tool. ProVision integrates with Excel, providing rapid model creation and manipulation using tabular data. The process querying lifecycle involves three steps, model inquiry, manipulation, and update. Model inquiry involves searching process XML data to focus on key process areas, generally to improve model quality. Model manipulation alters the process model XML using XQuery, XSL, and Excel macros. Model update applies the changes so that ProVision renders revised process model data to highlight the query results.

Prior to establishing a BPM practice with ProVision, the product development lifecycle was planned as a Gantt-style program schedule in Microsoft Project with occasional efforts to model processes in Visio. Neither Project nor Visio adequately met process modeling needs, as model size and complexity were overwhelming. Even after the introduction of ProVision models, some stakeholders resisted enterprise modeling. To secure BPM buy-in, stakeholders needed a way to move from process complexity to insight so they could incrementally refine model quality. The solution came in the form of process queries that filter out irrelevant details to focus attention on problems and opportunities within process models. Once value was established, demand for BPM and process querying increased.

This paper is structured as follows. Section 2 states the problem being addressed, namely the challenges of understanding, analyzing, and improving large, complex manufacturing processes when the subject matter experts responsible for authoring the process definitions approach them from different perspectives. Section 3 describes our method for querying these process models (i.e., model inquiry, manipulation, and update) to generate filtered process views from multiple perspectives that focus attention on opportunities for improvement. Section 4 presents results and extensions of work into related areas. Section 5 concludes with limitations, open problems, and lessons learned.

2 The Problem: Large, Complex Process Models

The automotive product development pipeline (from marketing concept to ready-for-mass-manufacturing) takes years. In this case, the entire process spanned 50 + process maps. Each map was printed on a one by two meter poster and displayed at least 70 activities. These posters – taped across the walls of a large room – presented a degree of complexity such that even with a magnifying glass, workflows were nearly impossible to follow. These maps were unwieldy, difficult to read, with too much unfiltered, detailed process information (see Fig. 1).

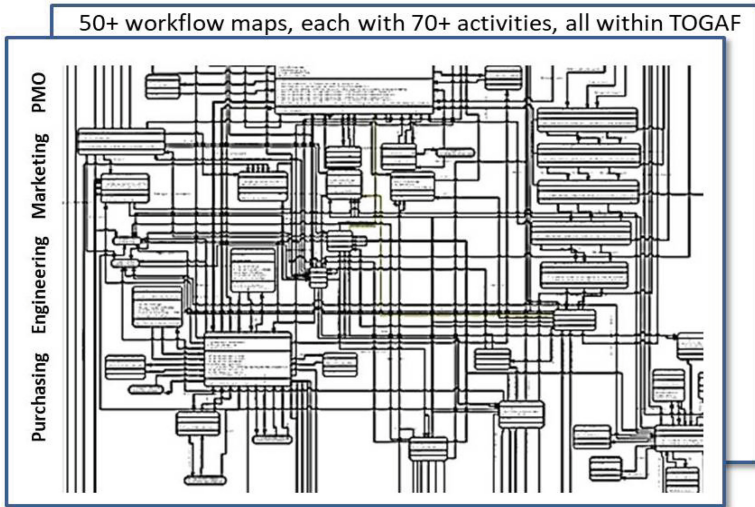


Fig. 1. Complex process maps, before query and filtering.

2.1 Differing Stakeholder Perspectives

Stakeholders, particularly the authors responsible for process design, approached models from different viewpoints (e.g., management, engineering, purchasing). From these viewpoints, they searched for different views (e.g., value streams, program schedules, workflow simulations, functional, business information, applications & services). The combination of viewpoints and views established search perspectives.

2.2 Quality Issues

BPM models were created by aggregating program management data from different teams using Microsoft Project, Visio, and custom, in-house applications. Once collected, data was loaded into Excel and split across object tables (e.g., activities) and link tables (e.g., workflows to interconnect activities). The resulting Excel files were imported into ProVision's inventory of modeling objects and links between objects. This approach to creating BPM models was faster than creating them by hand, but there were quality issues with the input data that led to model inconsistencies such as:

- Graph completeness problems [4]: missing inputs or outputs.
- Temporal problems [5]: inputs available after activity start, outputs produced too late, missing activity duration.
- Attribute quality problems: missing author/resources, typing errors in description.

3 Querying Business Processes

Process queries helped stakeholders to navigate interconnected models and to discover model improvement opportunities. For example,

- Given an author, find all of his/her activities within a workflow model.
- Given an artifact, find all activity usages (i.e., instantiations of an activity).
- Given a milestone, find the distinct list of artifacts that cross swim lane boundaries.
- Given a milestone, find the distinct set of artifacts that are associated with workflows that cross swim lane boundaries. (e.g., bill-of-material information handed off from one team to another).
- Given a role, highlight all activities performed by that role (e.g., marketing).
- Given a parameter, find model objects a matching attribute (e.g., find activities).
- Quality query: given a process model, verify that there are no workflows with one end detached (i.e., no dangling workflows).
- Quality query: given a process model, assert (number of activities where author is NOT missing = total number of activities).
- Compliance query: given a process model, compare it to the APQC reference model for automotive manufacturing to assess standards compliance noting drift [6].
- Navigation queries. Use query results as input to the next query. Repeat to navigate within and across models. For example, given an author, find all of his/her activities. Then, for each activity, trace all input workflows to find only those upstream activities owned by a different author. In this way, two authors can discover their dependencies other and coordinate their teams' planned process details.

3.1 Model Inquiry

Process data conformed to ProVision's common interchange format (CIF.xsd), an XML schema supporting model portability across vendor platforms (Fig. 2).

```

01 <activity id="157896" name="CAD-849">
02   <descr>Build prototype car parts</descr>
03   <parent refID="435524"/>
04   <workTime></workTime>
05   <performer refID="467908"/>
06   <customProperties>
07     <property name="Author">
08       <value>John Doe</value>
09     </property>
10   </customProperties>
11 </activity>

```

Fig. 2. An XML fragment of a process activity

On lines 7–9 of this XML fragment, the stakeholder responsible for this activity is stored in the author custom property, an important query search key. On line 1, the activity id “157896” is referenced throughout the process model to refer back to this activity (e.g., to connect it to workflows). These reference IDs chained together to enable navigation queries and nested searches and were invoked repeatedly as users traversed workflows. Missing data on line 4, <workTime>, is an example of poor quality input data that hinders queries such as finding critical paths to reduce time-to-market. Consider the query in Fig. 3: given an author, find all of her activities.

```

01 declare variable $author:="John Doe";
02 for $activity in /process/activities/activity
03 let $activity-id := $activity/@id
04 where $activity/customProperties/property
05   [@name="Author"]/value[matches(., $author)]
06 return <member refID="{ $activity-id}" />

```

Fig. 3. XQuery to return a collection of all activities owned by \$author

When this query runs, the result is a set of zero or more <member> elements as shown on lines 3–5 in Fig. 4. The set <members> in <modelScenario> includes only those activities owned by \$author.

```

01 <modelScenario name="Process layer, author filter">
02   <members>
03     <member refID="157896" />
04     <member refID="...etc..." />
05     ...etc...
06   </members>
07 </modelScenario>

```

Fig. 4. XQuery result set with references to all activities owned by \$author = “John Doe”

ProVision uses the <modelScenario> element to manage process simulation scenarios, which it renders as graph layers. We discovered that this element can be overloaded to create a collection of model layers, which, when superimposed on each other, filter out irrelevant process details to focus attention on query results.

3.2 Model Manipulation

This stage of the querying life cycle alters XML in a process model. For instance:

- Given query results, insert the results as rows into a new process model layer
- Given task duration data, populate the work time for each activity
- Given inter-activity timing data, populate transit time for each Workflow

This stage was challenging because most of this work was performed manually by editing boilerplate CIF.xml process data similar in format to XPDL and inserting query results. Note that the process definition files were often over 100 GB in size.

3.3 Model Update

Updating a model involved uploading a manipulated model definition into ProVision. In some cases, post-processing was applied to color workflows using JavaScript, which had the effect of highlighting workflows to draw attention to gaps, overlaps, and errors. See Fig. 5.

```
function highlightWorkflowsByStereotype(model) {
  var bpmParts = model.getComponents();
  for (i = 0; i < bpmParts.length; i++) {
    if (bpmParts[i].getType() == "Workflow") {
      var nextStereotype = bpmParts[i].getStereotype();
      if (nextStereotype == "WfOverlap")
        bpmParts[i].Line.setColor(BLUE);
      if (nextStereotype == "WfGap")
        bpmParts[i].Line.setColor(GREEN);
      if (nextStereotype == "WfError")
        bpmParts[i].Line.setColor(RED);
    }
  }
}
```

Fig. 5. JavaScript to highlight process model elements during model update

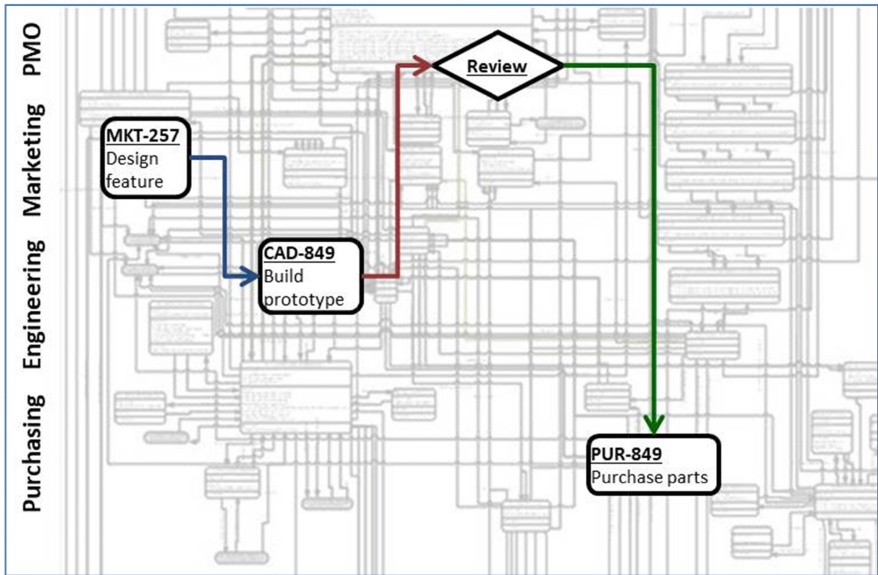


Fig. 6. Filtered workflow map with query results highlighted for visibility. (Color figure online)

When the collection of activities owned by a given author was combined with color highlighting of the workflows by stereotype, the resulting filtered process layer overlaid the ghosted process layer and was rendered as shown in Fig. 6.

4 Results and Extensions of Work

This process querying work helped a multi-disciplinary team realize a significant, undisclosed reduction in time-to-market for a multi-year manufacturing process, while improving overall model quality. Further, BPM gained acceptance among skeptics within the organization. Accordingly, the success of this work expanded beyond the original scope of improving BPM model quality.

4.1 Queries Applied to Other Enterprise Models

TOGAF, as it was used, specified seven types of enterprise models: strategy, organization, capability, process, information, application, and technology. This approach to querying process models was equally useful when applied to querying other enterprise models. Examples follow:

- An executive might start with a process model, and then search for the TOGAF business capability it implemented, and in turn, navigate to the associated TOGAF value stream.
- An enterprise architect might start with a process model, navigate to the information model it required, and then navigate to the application models that produced the information required by the process.
- A process author, assuming the role of purchasing manager, might start by searching an activity within the purchasing swim lane, then navigate upstream to work performed within other swim lanes (such as marketing or engineering). S/he could examine the attached artifacts (e.g., inputs such as marketing features or CAD data), and then create a new sub-process activity to handle bottlenecks (e.g., if substitutable parts had become necessary due to supplier issues).

4.2 Enterprise Architecture Portal

Process models were part of a broader collection of enterprise model portfolios [8], collectively containing over 800,000 model objects and covering all aspects of the company. The approach to process querying also applied to other model types within the TOGAF framework. An enterprise architecture portal was built so stakeholders could query all models types and navigate interconnected, filtered model views. This was done by selecting a model portfolio (e.g., vehicle design) and then selecting from configurable filters (e.g., filter by process map stage and organization perspective) as shown in Fig. 7.

Model metadata drove the portal's content and included support for multiple portfolios of models as shown in Fig. 8. This approach can be useful when comparing two sets of models, each from different authoring tools.

Order Management Technical Perspective: Process

Business Perspective: Product Milestone

Key	Value
Role	'BPO' 'Purchasing'
Bool.true	Pattern.matches("**error*", Activity.desc())

[Clear](#) [Open Model](#) 10/09/18
Model Publishing Timeline:

Filtered list of hyperlinks to models:

- Item 01
- Item 02
- Item 03
- ...etc...

With link syntax as:

```
<a href="https://metatech.us/portfolio.01/viewpoint.02/view.03/item.04.html" target="model_panel">Item.01</a>
```

The diagram on the right shows a process flow across swimlanes: Payer (Fill in Xpress form, Fill in order form), BPO (Enter referral details, Correct BPO errors), Processor (Enter referral details), customer portal (Order data), and Connections (Xpress request, Enter referral details, Referral data).

Fig. 7. Portal for exploring TOGAF enterprise models which included process models.

```

01 <Portfolio name="GlobalProductDev">
02   <Notebook name="OrderMgt" vers="" DateTm="">
03     <AuthoringTool product="ProVision" release="9.2"/>
04     <Models>
05       <Model>
06         Fltr01Key="Role" Fltr01Val="BPO | | Purchasing"
07         Fltr02Key="RegExp" Fltr02Val="Pattern ...etc..."
08         <Title>Order Entry</Title>
09         <Description>This order is...etc...</Description>
10         <URL>model_image.htm</URL>
11       </Model>
12     </Models>
13   </Notebook>
14 </Portfolio>

```

Fig. 8. XML data to populate model entries in enterprise architecture portal

4.3 Querying Workflows and Artifacts to Discover Micro-services

A firm-wide effort existed to replace legacy information systems with cloud-based, micro-services. Part of this work involved identifying workflows where process participants used email to hand off information across swim lane boundaries, a practice which led to document management issues and rework. Combining query results that identified sets of candidate workflows with lists of end-of-life systems provided a short-list of migration-eligible system as shown in Fig. 9.

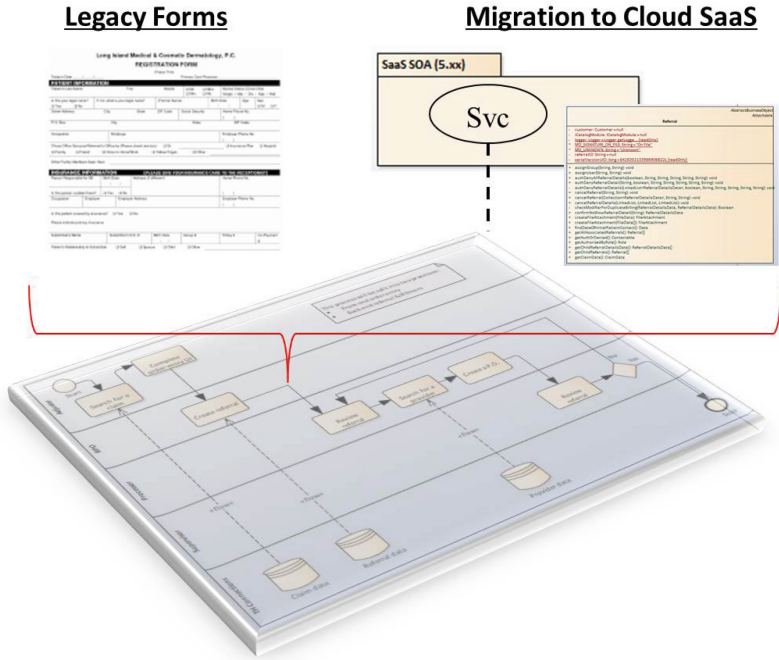


Fig. 9. Querying information flows to map legacy information systems to microservices

An example of a candidate information flow is the activity “PR-849 Purchase Parts” in the purchasing swim lane of Fig. 6. Such activities were identified by exporting process models to Excel and searching the descriptions of system, workflow, artifact, and activity objects via regular expressions to find target data (e.g., parts data, order data, CAD files, etc.). While this worked, a better approach would have been to use a process query language [7] with a search query along the lines of the following SQL-like pseudo code:

```

SELECT id FROM workflows AS w WHERE crossesSwimlaneBoundary(w.id) = true AND w.id IN (SELECT id FROM workflows AS w WHERE w.endLink.refId IN (SELECT id FROM activities AS a WHERE has_artifact(a.id) = true AND regExp(a.id, partsDataPattern) = true))

```

Ideally, such a PQL statement would be able to invoke regular expression searches (e.g., “[part|BOM].*data”) of artifacts outside the process model being searched in a manner similar to Transact-SQL’s xp_cmdshell() [9], but without security issues.

The end goal was an inventory of service-ready activities (SrActivities). In ProVision, when a process modeler drags an SrActivity onto the process designer canvas, the tool validates and instantiates the web services interface to the correct micro-service. A proof-of-concept was produced in ProVision using the Excel approach.

SrActivites were inventoried separately from non-service-ready activities, and web service interfaces were generated. Thus, it was possible to measure progress towards migrating activity inputs from legacy information systems and external MS Office documents (passed by email) to micro-services.

4.4 Filters to Normalize Models for Vendor-Neutrality

The long-term preservation of model assets is an important part of corporate records retention. Enterprise model artifacts must be portable across evolving modeling tools. Unfortunately, model fidelity is sometimes lost when exporting/importing models between tools (e.g., using BPMN or XPD). There is an inherent problem in relying on import/export features of BPM modeling tools because tools to manage enterprise content (ECM) and models (e.g., BPM) have different objectives than archival tools [13]. Anticipating the long-term need to preserve models across tools, a repository was created of rendered model views in both PDF and HTML formats. To create these static model views, process queries filtered out tool-specific branding (e.g., ProVision, Activiti, Sparks Enterprise Architect) and aggregated models published by all vendors. Thus, stakeholders could focus on unified views of enterprise models independently of the tools used to produce them. In this way, the modeling lifecycle and modeling-tool vendor management lifecycle could evolve independently.

5 Conclusion

This work focused exclusively on process querying as it relates to BPM modeling, not the mining of process logs, because many organizations are not yet ready for log mining. Even with this narrow focus, there was still much value in applying process querying to models.

5.1 What Worked Well; What Did Not

The most useful query was finding timing gaps (i.e., leads), overlaps (i.e., lags), and errors in workflows between activities. A gap exists when an upstream activity finishes one or more weeks before a downstream activity starts. An overlap exists when both activities execute concurrently for one or more weeks. A workflow error exists when the downstream activity starts before the upstream activity starts, or when one end of a workflow is unattached to a process element (activity, start, end, or gateway). Reducing time-to-market involved iteratively refining process models to close gaps, maximize overlaps, and eliminate errors.

Process models were planned backwards so the first activity (Design vehicle concept) had negative start and finish times, and the last activity (Confirm ready-to-manufacture) had a finish time of 0. Thus, given two activities A_i [S_i , F_i] and A_j [S_j , F_j], a Gap exists when $S_j < F_i$ an Overlap exists when $S_i \geq S_j > F_i$, and an Error exists when $S_j > S_i$, where A = Activity, S = Start, F = Finish, and A_j depends on input from A_i .

ProVition version 9.2 has a defect when importing model data from Excel: it does not load the `activity.workTime` column into the process model's activities, which prevents automated, critical path analysis. To circumvent this problem, workflows between activities were inventoried in Excel with one row per workflow, sorted by `activity.workTime` to prioritize leads, lags, and errors between adjacent process activities. With this prioritized list in hand, filtered views of process maps were created to highlight timing problems.

Regarding performance, the models exported by ProVition in its common interchange XML format were big, often over 100 MB. Loading them into OxygenXML Designer and ProVition led to non-linear processing delays (and occasional crashes), which seemed to grow exponentially with file size as the model was loaded into memory. In XML processing, streaming has better performance than loading large DOMs. [10] This is a consideration when designing process query languages and PQL processors.

The manipulation stage of process querying involved labor-intensive, batch work for developers, which proved challenging. The turn-around time to produce a filtered process layer could be as much as 20 min. Using shell scripts with regular expressions, experiments with XSL, and manual processing, layers were created. An area of future work would be to automate model manipulation so that stakeholders could execute ad-hoc PQL queries on the fly to explore and navigate models. Process modeling tools would have to support a PQL-compliant API in order to dynamically render ad-hoc queries.

5.2 Limitations, Open Problems, and Lessons Learned

Resources such as code snippets are available on github [11]. Areas for future research include:

- Improving XML processing performance—not enough effort was spent on measuring model size vs. processing time.
- PQL portability across modeling tools—based on this experience using a specific BPM modeling tool (ProVition and its common interchange format), it is clear that PQL portability across BPM tools will be in demand in industry.
- Support for ad-hoc queries and model navigation—the enterprise portal was popular among stakeholders. However, as implemented, it was limited in its ability to render dynamically-generated model views on the fly in response to ad-hoc queries. Such queries support exploration and discovery of interconnected models and are valuable to stakeholders.
- Model design drift and compliance—just as process instances drift during execution, so too does design intent drift when subject matter experts design and maintain large, complex process models over years. There was interest in archiving model changes for corporate history and in measuring the cost of model drift [12]. A big driver is tracking process compliance and alignment to industry standards, particularly the APQC standard for automotive manufacturing processes.

- The biggest lesson learned from stakeholder feedback was that PQL techniques are applicable to all TOGAF enterprise architecture layers, not just the process layer. Thus, an open problem and area for future research is whether and how to apply PQL techniques to performing queries in the broader context of enterprise architecture models.

References

1. Business Process Model and Notation Specification, version 2.0.2, sect. 7.3.1. <https://www.omg.org/spec/BPMN/2.0/>. Accessed 30 July 2018
2. TOGAF version 9.2, Part IV. <http://pubs.opengroup.org/architecture/togaf92-doc/arch/>. Accessed 30 July 2018
3. ProVision product data sheet. https://www.opentext.com/file_source/OpenText/en_US/PDF/ProVision%20Enterprise%20and%20Product%20Architecture%20Software%20Product%20Overview%20.pdf. Accessed 30 July 2018
4. Dettmer, H.W.: Goldratt's theory of constraints: a systems approach to continuous improvement. ASQC Quality Press, Milwaukee (1997). Section: The categories of legitimate reservation
5. Kumar, A., Sabbella, S.R., Barton, R.R.: Managing controlled violation of temporal process constraints. In: BPM 2015, pp. 280–296 (2015)
6. APQC Process Classification Framework (PCF) - Automotive (OEM) - Excel Version 7.0.5. <https://www.apqc.org/knowledge-base/documents/apqc-process-classification-framework-pcf-automotive-oem-excel-version-705>. Accessed 13 July 2018
7. <http://processquerying.com/pql-grammar/>. Accessed 01 Aug 2018
8. Polyvyanyy, A., Ouyang, C., Barros, A., van der Aalst, W.: Process Querying: Enabling Business Intelligence Through Query-Based Process Analytics, p. 3. Queensland University of Technology, Brisbane. <https://doi.org/10.1016/j.dss.2017.04.011>
9. Transact-SQL, xp_cmdshell(). <docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/xp-cmdshell-transact-sql>. Accessed 30 July 2018
10. Wei, Z.: Efficient XML stream processing and searching, pp. 1, 18–21. Florida State University (2012). <http://ww2.cs.fsu.edu/~wzhang/dissertation.pdf>. Accessed 11 July 2018
11. Github site for this project. <https://github.com/curiouskurt/pq2018>
12. Alizadeh, M., de Leoni, M., Zannone, N.: History-based Construction of Log-Process Alignments for Conformance Checking: Discovering What Really Went Wrong? p. 9, Eindhoven University of Technology, 19–21 November 2014
13. Katuu, S.: Enterprise content management and digital curation applications—maturity model connections. UNESCO Memory of the World in the Digital Age Conference, p. 1029 (2012)