DRain: An Engine for Quality-of-Result Driven Process-Based Data Analytics

Aitor Murguzur[1], Johannes M. Schleicher[2], Hong-Linh Truong[2], Salvador Trujillo[1], and Schahram Dustdar[2]

[1] Software Production Area, IK4-Ikerlan Research Center, Spain
`{amurguzur,strujillo}@ikerlan.es`
[2] Distributed System Group, Vienna University of Technology, Austria
`{j.schleicher,truong,dustdar}@infosys.tuwien.ac.at`

Abstract. The analysis of massive amounts of diverse data provided by large cities, combined with the requirements from multiple domain experts and users, is becoming a challenging trend. Although current process-based solutions rise in data awareness, there is less coverage of approaches dealing with the Quality-of-Result (QoR) to assist data analytics in distributed data-intensive environments. In this paper, we present the fundamental building blocks of a framework for enabling process selection and configuration through user-defined QoR at runtime. These building blocks form the basis to support modeling, execution and configuration of data-aware process variants in order to perform analytics. They can be integrated with different underlying APIs, promoting abstraction, QoR-driven data interaction and configuration. Finally, we carry out a preliminary evaluation on the URBEM scenario, concluding that our framework spends little time on QoR-driven selection and configuration of data-aware processes.

Keywords: Data-aware Processes, Runtime Configuration, Data Analytics, Smart Cities.

1 Introduction

The emergence of the smart city paradigm has created a plethora of new challenges for ICT [1]. Specifically, the analysis of large volumes of diverse data (referred to as Big Data) provided by large cities, combined with disperse requirements from multiple domain experts and stakeholders is becoming challenging [2]. For instance, the task of urban planning in the smart city context needs to collect data from all areas of significance ranging from energy consumption, construction and mobility systems to sociological factors, to just name a few.

Although workflows have been used to compose and execute a series of computational or data manipulation steps, such as scientific workflows [3,4], a few discussions have been focused on the utilization of runtime mechanisms to select and configure data-aware processes based on user-defined Quality-of-Result (QoR) to perform distributed data analytics. This is required in our URBEM[1]

[1] `http://urbem.tuwien.ac.at/`

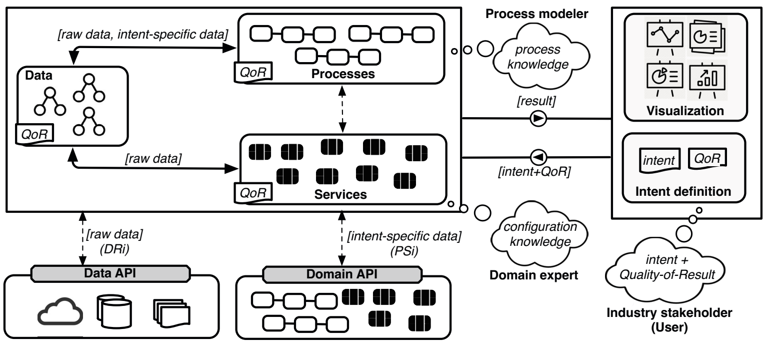


Fig. 1. Artifacts and interactions in process-based data analytics

scenario where large sets of data-aware process variants interact with data services, each with particular quality constraints. Hence, due to the high variability of data-aware analytics processes and data endpoints, it is crucial to provide means of quality-driven process selection and customization at runtime.

1.1 Motivation, Contributions and Paper Structure

In process-based data analytics the data needed to actually execute process activities is much broader than the typical process-related data (see Fig. 1). Although *raw data* (e.g. data from *Data APIs*) can be relevant to several artifacts, such as services and process activities, it is not bound to any specific intent and thus represents general information. On the other hand, the results of remote analytics processes and available services can be offered as *intent-specific data*, exposing the expected result as Data as a Service (DaaS).

Services include computational models (e.g. MATLAB model) from domain experts which are required by activities in a process execution. An data-aware analytics *process*, referred to as Workflow as a Service (WFaaS), represents a particular intent for an industry stakeholder. Such process logic is represented in form of process models (e.g. BPMN2, BPEL), which stands for a particular analytics type, consisting of a number of activities to be executed. Hence, process instances are created on user *intent* request which may indicate a desired QoR.

In this scenario and due to the high variability of related processes and data variety, we need to defer WFaaS selection and configuration to runtime, where process variants are customized and executed based on QoR. This would reduce the complexity of managing large sets of process variants, as well as binding suitable data endpoints and processes ensuring required QoR for analytics. However, although a number of approaches have been focused on data analytics processes, such as scientific workflows [3,4], Quality-of-Service (QoS) based service selection [5,6], and process variant re-configuration [7,8,9], none of them are capable of selecting and configuring data-aware process variants based on QoR at runtime.

In this paper, we therefore present some of the fundamental building blocks of a framework (called `DRain`) for QoR-driven selection and configuration of data-aware processes. The main contributions (C) are: **C1** - we propose an approach and a prototype framework to select, configure and execute data-aware analytics processes at runtime; and **C2** - we demonstrate through an evaluation on a real example from URBEM that our framework spends little time for selecting analytics processes and data endpoints, as well as configuring variation points (`DRi` activities) based on data from the data realm.

The rest of the paper is structured as follows: In Section 2 related work is summarized. We present the overall architecture and detail individual framework building blocks in Section 3. Section 4 evaluates the functionality and usefulness of the presented approach by encoding a realistic example in URBEM. Lastly, we conclude the paper and present the direction of future work in Section 5.

2 Related Work

Alternative approaches have been focused on employing workflows for data analytics, such as in scientific workflows [3,4], but without considering QoR, a term originally coined for data analytics [10], or Quality-of-Service (QoS) to drive process selection and configuration. The term QoS is mainly used in the area of service composition. In this light, the Discorso framework [5] facilitates late binding of services by the subsequent selection of applicable Web services based on supervision rules and QoS constrains at runtime. Canfora et al. [6] provide a QoS-aware composite service binding and re-binding approach based on Genetic Algorithms. These latter provide useful methods for QoS-based service (re-)binding; however, they are focused on service selection, rather than enabling process configuration at runtime through QoR-driven data interactions.

On the other hand, process re-configuration [7,8,9] capabilities have been promoted by other authors. For instance, the CEVICHE framework [7] enables BPEL process schema level re-configuration by means of monitoring QoS (service availability and service performance). In a similar vein, Xiao et al. [8] present a constraint-based framework to enable re-configuration (changing relationships among fragments through constraints) and adaptation through adaptation policies to select fragments at runtime. Additionally, in [9] autonomic mechanisms are used to guide the self-adaptation of service compositions according to context changes and variability specification. With respect to the mentioned work, we are not focused on re-configuration, otherwise our approach defers QoR-driven process selection and configuration to runtime.

Last but not least, process configuration abstractions have also been proved by other authors. For instance, a requirements-driven approach [11] enables the configuration of BPEL processes based on quality constraints. Similarly, a questionnaire-driven approach [12] enables a step-wise configuration of reference processes at design-time. However, to the best of our knowledge, no framework is capable of customizing QoR-driven data-aware process variants at runtime.

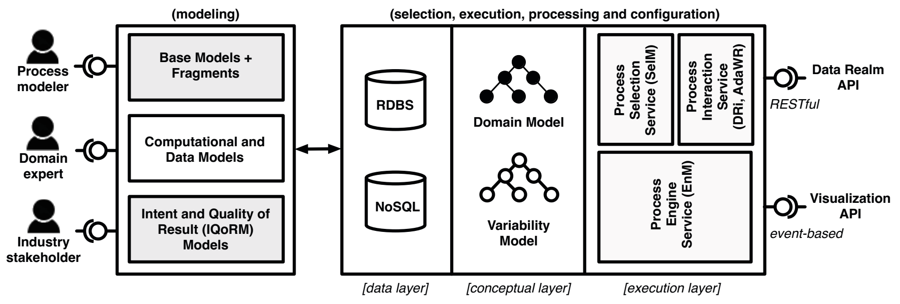

(modeling)
(selection, execution, processing and configuration)
Process modeler
Domain expert
Industry stakeholder
Base Models + Fragments
Computational and Data Models
Intent and Quality of Result (IQoRM) Models
RDBS
NoSQL
Domain Model
Variability Model
Process Selection Service (SelM)
Process Interaction Service (DRi, AdaWR)
Process Engine Service (EnM)
Data Realm API
RESTful
Visualization API
event-based
[data layer]
[conceptual layer]
[execution layer]

Fig. 2. The DRain framework overview

3 DRain Building Blocks

The `DRain` framework (see Fig. 2) allows for modeling, configuration, processing and execution of data-aware analytics processes based on user-defined QoR.

3.1 Modeling

Base Models and Fragments. In order to correspond to different user-defined QoR needs (e.g. time and cost constraints), the process modeler may create *base models* and *fragments* using BPMN2 elements (e.g. Service Tasks for invoking available *computational model* services) and custom variation points (the so-called `DRi` activities). In essence, a base model represents the commonality shared by a process family and variation points that are subjected to change. Variation points identify specific parts in a base model where data interaction and fragment selection occur (in `DRi` activities). A process fragment, or fragment for short, describes a particular configuration option for each variation point within a base model. For a more detailed discussion of our foundations regarding process variability modeling, we refer to our previous work [13].

IQoRM. The *Intent and Quality of Result Model (IQoRM)* is reflected through an UML diagram, containing intents representing user requests, constraints (QoR) representing user restrictions and analytics scope (see Fig. 3). Those abstractions are used to construct a data analytics task and its strategy, and thus represent constraints for the desired behavior. The analytics range is limited by the `Scope` class, which delimits the range of an analytics `Intent`. For example, a user might want to perform the analysis: "determine the energy consumption for the specific district X" (X can be any district of the city). In this case, "energy consumption" is the desired intent which contains "for the specific district X" as a delimiter. Two sub-classes are differentiated: `ConfigurationScope` which demarcates between different configuration alternatives (e.g. the `value` variable may determine selection (`SelM`), configuration (`ConCM`), discovery (`DiscM`) or composition (`CompM`) alternatives, as distinguished

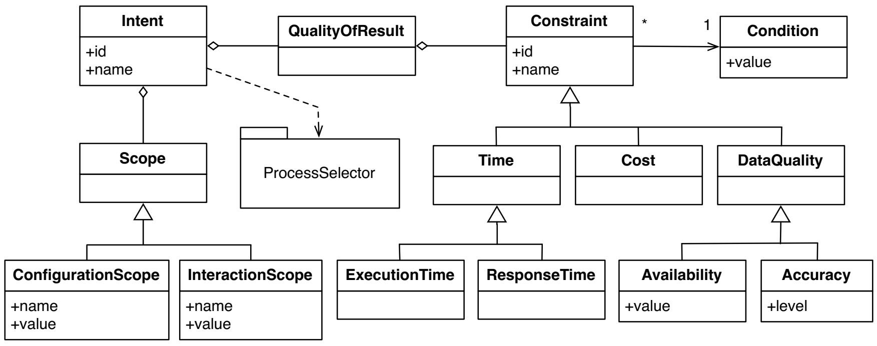


Fig. 3. IQoRM model

by [14]), and `InteractionScope` for limiting available data endpoints (e.g. the `value` variable delimits the range of data endpoints, such as check `ALL` available resources from the data realm). Such parameters delimit data interactions during a base model execution which considers both interaction and configuration scopes, i.e., data endpoints that should be considered for a particular function.

`QualityOfResult` can determine not only the selection of a particular analytics process, but also the binding of inherent data endpoints. A `Constraint` is a basic aspect of our framework which represents some condition, restriction or assertion related to the analytics artifacts. It includes a set of `Conditions` that are atomic formulae or implications (see the code snippet below) for driving process selection and customization. Conditions (`lessThan`, `greaterThan`, `inBetween`) may be applied to different constraint types. In Fig. 3, three constraints are considered: (i) `Time` for the entire time that a process, computational service or data service takes for execution (`ExecutionTime`) and its network channel to ping (`ResponseTime`), (ii) `Cost` which represents the cumulative expected cost of performing action, and (iii) `DataQuality` to exhibit the `Availability` (refers to the availability of data) and `Accuracy` (refers to the level of provided data values confidence) of provided data endpoints. In a simplified version, we consider {`True,False`} for the former, and {`Green,Orange,Red`} for the latter. Once intent, scope and QoR are specified, `ProcessSelector` initializes the search algorithm for finding a relevant analytics process.

Excerpt of a QoR condition

```
quality.addResponseTime(new ResponseTime("responseTime",
    Condition.lessThan(400)));
```

3.2 Selection, Execution, Processing and Configuration

SelM. In `DRain`, the domain model is defined using ontologies. This type of representation has been widely accepted as a conducive method for domain modeling (knowledge vocabulary) and reasoning, with low impact on scalability and performance. Our domain model defines six types of primitive classes which include several individuals and object/data properties as follow: (i) `Intent` individuals with *hasIntentName* data property mapped to the *name* parameter in Fig. 3, (ii) `Scope` individuals with *hasConfigurationScope* and *hasInteractionScope* data properties, (iii) `Time` , `Cost` and `DataQuality` subclasses of `QualityOfResult` class, (iv) `BaseModel` and `Fragment` as subclasses of `Process` individuals, (v) `DataEndpoint` individuals which contain *hasURI*, *hasServiceName* and *hasDataModel* data properties, and (vi) `ConfigurationModel` individuals with *hasFileName* property to point a particular variability model. Hence, the process selection service is capable of retrieving base models (WFaaS) for a given petition. The first suitable base model that meets user-defined QoR is then instantiated.

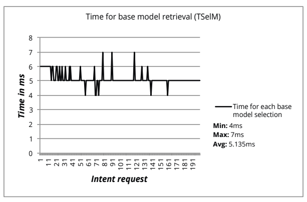
EnM, DRi, AdaWR. Once an analytics process (base model) execution reaches a `DRi` activity, the process engine follows several steps. If there is no fragment assignment for the current `DRi` activity execution, this activity throws an event to select a suitable fragment based on context data. Such selection requires two types of processing. In the first *interaction* task, the event coming from a `DRi` execution is triggered by the process interaction service to find a single data endpoint URI that satisfies pre-established QoR constraints (by running a SPARQL query). Data collected from a REST resource (in JSON) is mapped to a data model object (by a *hasDataModel* data property) to automatically perform the base model instance *configuration.* For the latter, the context values gathered from the REST service are mapped to placed attributes in a variability model, in order to get a preferred fragment choice considering pre-established constraints and fragments for each variation point. Once a suitable fragment is resolved using a Solver, `DRain` signals the particular `DRi` activity execution which executes the preferred fragment and then continues its control-flow.

We adopt feature models [15] to model all configuration options for each analytics base model and surrounding `DRi` activities (i.e. variation points) in a variability model. The mapping between a domain model and a variability model is realized by naming compounds as follows: (i) feature names are mapped to `BaseModel` and `Fragment` individuals *hasProcessKey* data property, (ii) variation point features are linked to *hasServiceName* data property of a `DataEndpoint` individual, and (iii) variability model attributes are related to data model variables from each `DataEndpoint`. The latter relation correlates data model variable/-value pairs with feature model attributes.

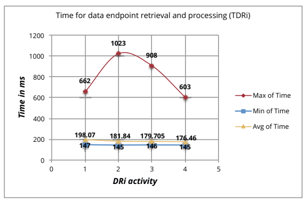
4 Evaluation

In the following, we briefly describe the evaluation scenario in URBEM and present the results of the evaluation runs on `DRain`. The `DRain` framework was developed in Java and Clojure based on open source technologies.

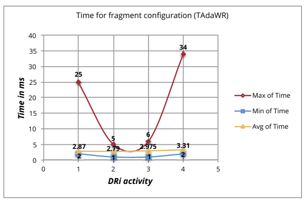
Provided Models. For the evaluation, we created 30 base model variants (individuals) with different time and cost QoR constraints for an energy consumption intent in URBEM. This analytics process consists of four `DRi` activities (variation points) and two service tasks. Each `DRi` activity contained 2 fragment



(a) Metric 1: TSelM



(b) Metric 2: TDRi



(c) Metric 3 : TAdaWR

Fig. 4. Evaluation results

alternatives, providing each distinct data quality (availability and accuracy), so we get $2^4 = 16$ variant customizations for the outlined base model. Moreover, we created 24 data endpoints with different QoR and five data values were parsed in a configuration model in each data interaction.

Metrics. In order to obtain a reliable evaluation, we processed the base model 200 times and evaluated the results against three performance-related metrics:

– *Time for base model retrieval (TSelM)*: This metric measures the time required for intent-driven and QoR-based base model searching.
– *Time for data endpoint retrieval (TDRi)*: This metric defines the timespan from `DRi` activity initialization to the moment when the process interaction service finds a suitable data endpoint for the given QoR and invoked the particular REST resource to collect data.
– *Time for fragment solving (TAdaWR)*: This metrics measures the time required to establish context values and find a suitable fragment once data is gathered from a REST resource.

Results. The results of our evaluation in terms of the average of all evaluation runs are provided as graphics in Fig 4.[2] Overall, we can state that our engine operates with little impact on performance, and slightly affects the execution time required by each analytics process instance. This allows for QoR-driven selection and configuration of data-aware analytics processes that involve a considerable number of variants and data endpoints (i.e. 30 and 24 respectively in the evaluation), offering greater flexibility and abstraction. As shown in Fig. 4 (a), the difference between the minimum and maximum time required for a base model retrieval (`TSelM`) based on a user-defined QoR is about **3ms**. In a similar vein, the average time for data endpoint selection and processing (`TDRi`) is reasonable at **184.019ms**, considering both sequential, such as `BuildingSpecification` and `EnergyDemand`, and parallel activities, such as `ElectricalGridUtilisation` and `ThermalGridUtilisation` (check supplement file). Finally, it is also important to note the overall average time required to complete the runtime configuration, e.g., for (`TAdaWR`) an average time of **2.986ms** is necessary for putting five context values in the variability model to set a particular fragment for a given `DRi`.

[2] All datasets, a detailed description of the example and additional files are available at: `https://github.com/amurguzur/drain`

5 Conclusion and Future Work

In this paper, we presented the main building blocks of a framework (`DRain`) to automatically perform a Quality-of-Result (QoR) driven selection and configuration of data-aware processes. Specifically, our approach enables abstractions to select relevant analytic processes (exposed as WFaaS) and data endpoints based on user-defined QoR, and provides flexibility in terms of runtime process variants configuration. A preliminary evaluation concluded that our framework is capable of high-performance selection, processing and configuration of data-aware processes in subsequent QoR-driven data interactions. For future work, we plan to extend the associated framework and test it against industrial case studies, and adapt the QoR model for a more domain-specific environment. Further, we will explore ranking and selection algorithms/dimensions using QoR.

References

1. Naphade, M., Banavar, G., Harrison, C., Paraszczak, J., Morris, R.: Smarter cities and their innovation challenges. Computer 44(6), 32–39 (2011)
2. Khan, Z., Kiani, A.A.,, S.L.: Cloud based big data analytics for smart future cities. In: UCC Workshops (2013)
3. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: an extensible system for design and execution of scientific workflows. In: SSDBM, pp. 423–424 (2004)
4. Hauder, M., Gil, Y., Liu, Y.: A framework for efficient data analytics through automatic configuration and customization of scientific workflows. In: e-Science, pp. 379–386 (2011)
5. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering 33(6), 369–384 (2007)
6. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: A framework for qos-aware binding and re-binding of composite web services. J. Syst. Softw. 81(10) (2008)
7. Hermosillo, G., Seinturier, L., Duchien, L.: Using complex event processing for dynamic business process adaptation. In: SCC, pp. 466–473 (2010)
8. Xiao, Z., Cao, D., You, C., Mei, H.: Towards a constraint-based framework for dynamic business process adaptation. In: SCC, pp. 685–692 (2011)
9. Alférez, G., Pelechano, V., Mazo, R., Salinesi, C., Diaz, D.: Dynamic adaptation of service compositions with variability models. In: JSS (2013)
10. Truong, H.L., Dustdar, S.: Principles of software-defined elastic systems for big data analytics. In: MIE, pp. 10–14 (2014)
11. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
12. La Rosa, M., Lux, J., Seidel, S., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-driven configuration of reference process models. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 424–438. Springer, Heidelberg (2007)
13. Murguzur, A., De Carlos, X., Trujillo, S., Sagardui, G.: Context-aware staged configuration of process variants@Runtime. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 241–255. Springer, Heidelberg (2014)
14. van der Aalst, W.M.P.: Business process management: A comprehensive survey. ISRN Software Engineering, 37 (2013)
15. Batory, D.: Feature models, grammars, and propositional formulas. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 7–20. Springer, Heidelberg (2005)