A Knowledge-Driven Approach for Designing Data Analytics Platforms

Madhushi Bandara · Fethi A. Rabhi

Abstract Big data analytics technologies are rapidly expanding across all industry sectors as organizations try to make analytics an integral part of their everyday decision making. Although there are many software tools and libraries to assist analysts and software engineers in developing solutions, organizations are looking for flexible analytics platforms that can address their specific objectives and requirements. To minimize costs, such platforms also need to co-exist with existing IT infrastructures and reuse information and knowledge resources already accumulated in the organization. To assist organizations in addressing their specific needs, this paper proposes the **D**ata **A**nalytics **S**olution **E**ngineering (DASE) framework - a knowledge-driven approach supported by semantic web technologies for requirements elicitation, and for designing and developing new data analytics platforms. It includes a meta-model that captures data analytics platform requirements via a knowledge base, a set of guidelines that organizations can follow in engineering data analytics platforms and an implementation that demonstrates how to use these guidelines. We evaluate the capabilities of the proposed approach by conducting two case studies in which we develop data analytics platforms for house price prediction and time-series data processing. The results show that the DASE framework can be used to rapidly engineer flexible, user-friendly and easy-to-maintain data analytics platforms that reflect organizational analytics requirements.

Madhushi Bandara
School of Computer Science, University of Technology Sydney, Australia
E-mail: madhushi.bandara@uts,edu.au

Fethi A. Rabhi
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia.
E-mail: f.rabhi@unsw.edu.au

1 Introduction

Even though data analytics is a popular topic among industry and researchers alike, organizations are struggling to implement data analytics solutions that match well with their organization's IT infrastructure and analytics goals. In many cases, organizations outsource the design and development of data analytics solutions to external contractors. The resulting solutions carry significant technical debt and are hard to maintain, especially when new requirements emerge over time. Even a slight shift in the organizational analytics requirements, such as adding a new algorithm or integrating a new data source, requires full re-engineering of the existing data analytics platforms and processes [1]. Furthermore, the experience and knowledge accumulated by the external teams will not be available for future use in the organization. The alternative is to use open source or commercial products such as WEKA or Tensor-flow for organizational data analytics. They offer a wide range of analytics capabilities but are difficult to customize, require high technical expertise to operate and restrict organizations to use only the popular analytics methods [2, 3]. Recent surveys [3, 2] highlight these challenges and advocate the need of better technologies to manage and utilize the data analytics related knowledge and resources in an organization that can to provide the means to rapidly develop and evolve user-friendly data analytics platforms.

In an effort to address these challenges, this paper proposes the **D**ata **A**nalytics **S**olution **E**ngineering (DASE) framework, a knowledge-driven approach which utilizes semantic web technologies to improve data analytics platform design and development. It is built on the premise that building data analytics solutions is a knowledge intensive task that requires expertise of software engineers, data scientists as well as domain specialists. The use of semantic web technologies gives the opportunity to build rich information models that can integrate knowledge from different spheres and support flexible software design.

The DASE framework relies on a knowledge base that can accumulate organizational, domain, analytics and service related information as well as integrate it with open knowledge. A key component of the proposed knowledge base is the Analytics Requirements Ontology (ARO), a meta-model that can represent data analytics platform requirements in an organization. The framework includes an architecture organizations can adapt, which aims to utilize the knowledge base and elicit data analytics platform requirements, define the analytics platform design elements and support the implementation of new data analytics platforms. To evaluate the DASE framework, we conduct two real-world case studies related to house price prediction and time-series data analytics, using a prototype implementation of the proposed the DASE architecture.

The paper is structured as follows. The next section discusses the background and related work of the DASE framework, followed by a description of the main elements of the framework in section 3. Section 4 presents the prototype we developed for the DASE framework. Section 5 describes the evaluation process and the paper concludes in section 6, discussing limitations and the potential future work.

2 Background and Related Work

Literature suggests that, compared to the amount of research dedicated to designing data analytics models, there are fewer studies that focus on software engineering issues such as providing adequate software infrastructures, analytics knowledge representations and workflows to assist data analysts [4]. Among them, there are many research studies that propose new meta-learning platforms [5], Service-oriented Architecture (SOA) designs and workflow-based systems (e.g. WINGS [6], ADAGE [7], Clipper [8]) to assist the engineering of analytics solutions. Although such studies have been able to address many challenges associated with data analytics platform development, they lack a sound methodology that can consistently capture details of the different artifacts and processes produced in relation to building data analytics platforms. Multiple studies highlight the need of comprehensive knowledge repositories that can assist analysts to understand data analytics requirements, analytical models, datasets used, decision-making processes and relevant domain information that can assist future analytics operations [2, 3, 4].

We believe that the paradigm of building a knowledge base supported by semantic web technology, as proposed by Berners-Lee et al [9], can be useful in this context. Semantic web technologies have a well-developed set of standards and notations such as RDF, RDFS and OWL. These standards are supported by different tools for modelling, storing, querying and inferencing the knowledge. Using these standards and tools for modelling semantic rich information about analytics artifacts and processes can advance the way organizations design and develop their data analytics platforms.

Different communities have adapted semantic technologies to build standard ontologies related to their practices. While there are examples of leading internet companies (e.g., Google, Amazon and Facebook) beginning to exploit the power of semantic search and domain ontologies (e.g., Schema.org, DBpedia, SNOMED), many industry players are still largely unaware of the value that these approaches represent [10, 11, 12].

A recent systematic literature review [13] has studied existing research efforts related to the use of semantic technology to aid data analytics platform design and implementation. The majority of identified studies use semantic models to support isolated activities such model generation [14] or data source selection [15]. The extant literature covers well data mining techniques and knowledge discovery processes (e.g. [16]). Yet there are scarce research efforts whose primary objective is in linking multiple facets of the expert knowledge related to the different phases of the development process (ie. requirements gathering, design and implementation). The Research Variable Ontology (RVO) [17] represents a preliminary effort that has been made to address this issue. RVO is a semantic model that is capable of integrating organizational analytics knowledge with open knowledge in a way that can support the decision-making process of data analysts.

Although a semantic model such as the RVO can be used to design an organizational analytics knowledge base, its practical usage in a way that integrates well with an organization's existing IT infrastructure is not straightforward [18]. For this reason, this paper proposes the DASE framework as an approach for an organization to utilize a knowledge base in engineering an end-to-end data analytics solution from capturing requirements, to platform design and development. Data analytics platforms developed via the DASE framework can be customized
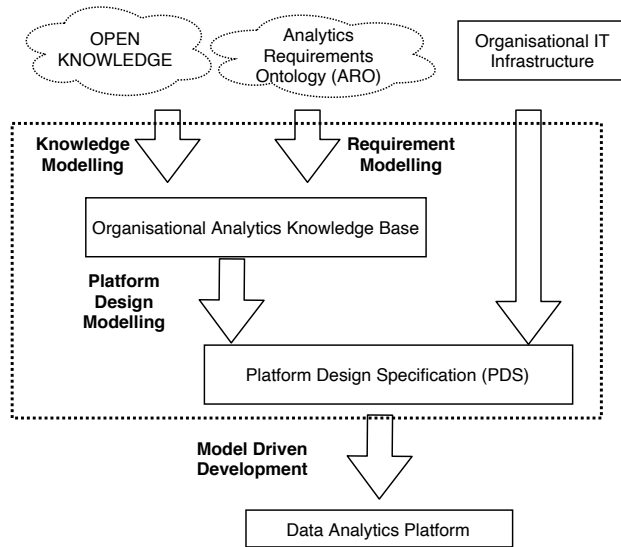
Fig. 1 Scope of the DASE Framework

for specific organization requirements, are flexible enough to be rapidly modified and can reduce the cognitive burden on data analysts when it comes to analysing data in a new domain or adapting new approaches, resulting in a shorter learning curve.

3 DASE Framework

3.1 Overview

Fig. 1 shows the scope and the main building blocks of the DASE framework and the steps of utilizing analytics knowledge, requirements and IT infrastructure to generate a Data Analytics Platform. A Data Analytics Platform in the scope of this paper is an interactive application with a GUI, backed by a knowledge base and a set of services used by the data analysts in an organizationto perform their analytics activities. The knowledge engineers and system engineers will utilise the DASE framework to develop the Data Analytics Platform, in consultation with the data scientists, as they are the end users. As the DASE framework is based on service-oriented and workflow architecture design principles,organizations are free to leverage IT components (databases, middleware and machine learning technologies) from the existing infrastructure, and expose them as services to be utilized by the Data Analytics Platform.

Within the scope of the DASE framework, we define *Open Knowledge* as all the publicly available knowledge related to data analytics represented in semantic web as linked-data through different ontologies. The *Analytics Requirements Ontology (ARO)* is a main contribution of our paper. It is a meta-model that can represent knowledge about Data Analytics Platform requirements in an organization.

As Fig. 1 shows, ARO, together with other Open Knowledge, are used in *Knowledge Modelling* and *Requirement Modelling* to creat the *Organizational Analytics Knowledge Base*: an integrated information repository reflecting the organisational analytics activities, resources and experties.

Platform Design Modelling is the step where Organizational Analytics Knowledge Base and Organizational IT Infrastructure are utilized to create *Platform Design Specification (PDS)*. As transforming a PDS into an executable platform via Model Driven Development (MDD) principles is a mature and independent field of research [19], how to automatically transform a PDS into an executable Data Analytics Platform is left out of the scope of this paper.

3.2 Knowledge Modelling and Requirement Modelling

3.2.1 Introduction

During the Knowledge Modelling step, the organization develops an analytics knowledge base, accumulating and integrating all analytics related knowledge and resources such as domain knowledge, details about past analytics experiments and findings, and information about resources and services contained in organization IT infrastructure. Organizations can reuse *Open Knowledge*; publicly available taxonomies, ontologies and linked-data. It is also possible to extend Open Knowledge and model organization specific data following semantic web and linked-data standards.

The Requirement Modelling step involves the use of the Analytics Requirement Ontology (ARO) to elicit requirements for the Data Analytics Platform, modelling of requirements in detail and linking them with Open Knowledge providing clarity and context.

3.2.2 Analytics Requirement Ontology

The ARO- Analytics Requirement Ontology, illustrated in Fig. 2, helps organizations to identify and catalogue requirements related to a particular Data Analytics Platform. We designed ARO with its three core concepts defined using UML stereotypes following the Object Management Group standards [1]. This way, organizations can easily translate and extend traditional requirement specifications into ARO. The UML concepts we extended are:

- **"UseCases"**: a means to capture the requirements of systems, i.e., what systems are supposed to do. The behavior of a **"UseCases"** can be described by a set of elements such as interactions (e.g. activity diagram/interaction diagram or communication diagram.)
- **"Activity"**: a behavior specified as sequencing of subordinate units, using a control and data flow model.
- **"Action"**: the fundamental unit of behavior specification in UML.
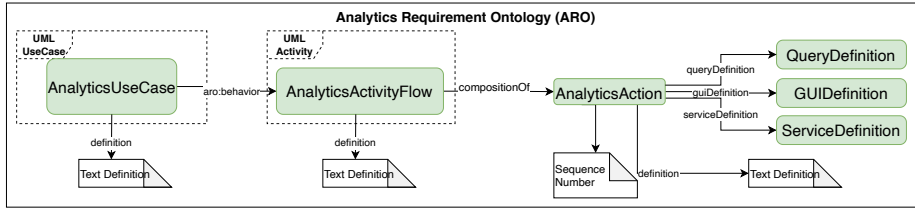
These are the main concepts of ARO (Fig. 2):

---

[1] https://www.omg.org/spec/UML/2.5.1/PDF

Fig. 2 Main concepts of ARO

- *AnalyticsUseCase* is a stereotype of UML **"UseCases"** that anchors require-ments of a specific analytics platform by identifying the specific scenario per-formed by the analyst.
- *AnalyticsActivityFlow* is a stereotype of UML **"Activity"** representing a com-mon group of steps analysts will follow to perform an *AnalyticsUseCase*.
- *AnalyticsAction* is a stereotype of UML **"Action"** that captures each unit step of *AnalyticsActivityFlow*. The resulting data analytics platform needs to support these actions via the application workflow. *AnalyticsAction* contains textual information and a sequence number to ensure the order of action within *AnalyticsActivityFlow*.

We identify three components of *AnalyticsAction* that can capture low-level requirements necessary for platform design modelling.

- *QueryDefinition* defines the nature of the query, with its inputs and out-puts, that can retrieve relevant information from the Organizational Analytics Knowledge Base to fulfill the related *AnalyticsAction*.
- *GUIDefinition* defines the type of Graphical User Interface (GUI) required to perform the related *AnalyticsAction*.
- *ServiceDefinition* defines the characteristics of the service (i.e. API) that needs to be invoked by the Data Analytics Platform to access a service available in the organizational IT infrastructure.

3.2.3 Utilizing ARO for Requirement Modelling

To demostrate how analytics requirements are modeled using ARO, we instanti-ate two scenarios as *AnalyticsUseCase* using ARO. The first *AnalyticsUseCase* in Fig. 3 is 'Conducting new forecast using an existing model'. Its behavior is rep-resented by a very simple *AnalyticsActivityFlow* named 'AAF-Fore'. It contains three *AnalyticsActions* that define the sequence of steps a data analyst will fol-low to fulfill the *AnalyticsUseCase*. Fig. 4 provides another *AnalyticsActivityFlow*
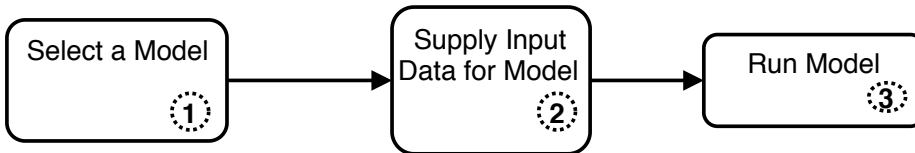

Fig. 3 AAF-Fore: An instance of ARO *AnalyticsActivityFlow* for the *AnalyticsUseCase*: 'Con-ducting new forecast using an existing model.'
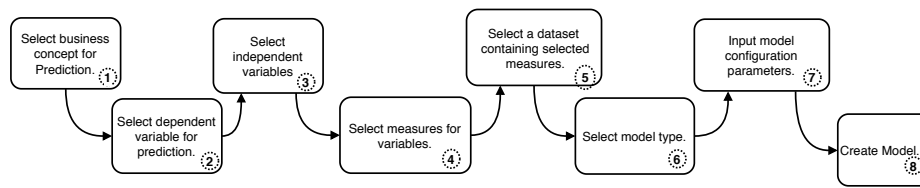
Fig. 4 AAF-Train: An instance of *AnalyticsActivityFlow* for the *AnalyticsUseCase*: 'Training a new prediction model.'

named AAF-Train to capture the behaviors of 'Training a new prediction model' *AnalyticsUseCase*. Further examples of *AnalyticsUseCases* are provided in supplementary materials.

To further illustrate the nature of information captured by ARO, in Table 1 we present how ARO is used to define all requirements related to these two *AnalyticsUseCases*. These requirements were developed and refined in consultation with the data analysts (end-users) involved in the two case studies presented in Section 5.

3.3 Platform Design Modelling

Once an *AnalyticsUseCase* is modelled in ARO, it can be utilized to model a Platform Design Specification linking with the organizational IT Infrastructure and the organizational analytics knowledge base. The Platform Design Specification has four components.

- *Query Management:* Each *QueryDefinition* requirement from ARO is used as a basis to define a SPARQL query against the Organizational Analytics Knowledge Base to access information necessary to fulfill the *AnalyticsAction*.
- *GUI Design:* Each *AnalyticsAction* needs a detailed graphical user interface (GUI) that satisfies all the *GUIDefinition* requirements. Details of the user interface such as what GUI elements to be used (e.g.- buttons, drop-down lists, sliders), their color, size etc. and how to organize GUI elements need to be defined.
- *Service Management:* If an *AnalyticsAction* has a *ServiceDefinition* requirement, service invocation details need to be modelled and managed. This includes information such as the input parameters required by the service and how to read and manage the service response.
- *Workflow Design:* All the *AnalyticsActivityFlow* and associated *AnalyticsActions* are ordered in specified sequence and composed into an implementable and machine readable workflow model.

4 Prototype Implementation

4.1 The DASE Architecture

The DASE architecture contains an architectural recommendation for organizations to implement and adapt the DASE framework. The DASE architecture is

Table 1 Example ARO based Requirement Specifications for Two *AnalyticsUseCases*

Analytics Use-Case and *AnalyticsActivityFlow*	*Analytics Action*	*Query Definition*	*GUI Definition*	*Service Definition*
Conduct new forecast using an existing model (AAF-Fore)	Select a Model	Retrieve all models available	Visualize the models available for user to inspect and capability to select one of them.	-
	Supply Input Data for Model	Retrieve the measures related to independent variables of the selected model	List all measures, with their definition, data type, data range and provide capability for user to input values or select from a possible value set.	-
	Run Model	Retrieve information about the service that can execute the selected model	List service configuration parameters user needs to input for the particular service fetched by the query. Provide capability for user to input values and a button to execute the service. Visualize results returned by the service	Invoke the particular service fetched by the query with user inputs (configuration parameters, selected Model, and input data). Retrieve results.
Train a new prediction model (AAF-Train)	Select domain concept for prediction	Retrieve all predictable domain concepts	List the domain concepts fetched by the query for user to inspect and select one or more of them.	-
	Select dependent variable for prediction	Retrieve all variables related to selected domain concepts	List the variables fetched by the query for user to inspect and select one of them.	-
	Select independent variables	Retrieve all independent variables linked to the selected dependent variable	List the variables fetched by the query for user to inspect and select one or more of them.	-
	Select measures for variables	Retrieve measures that represent selected dependent and independent variables	List the set of measures associated with each variable fetched by the query for user to inspect and select one measure for each variable.	-
	Select a dataset containing selected measures	Retrieve datasets that contain selected measures	List the datasets fetched by the query for user to inspect and select one of them.	-
	Select model type	Retrieve model types supported by the platform	List the model types and their performance details fetched by the query for user to inspect and select one of them.	-
	Input model configuration parameters	Retrieve configuration parameters for the selected model type	List all model configuration parameters, with their definition, data type, data range and provide capability for user to input values or select from a possible value set.	-
	Create Model	Retrieve service that can train selected model type	List model configuration parameters user needs to input for the particular service fetched by the query. Provide capability for user to input values and a button to execute the service. Visualize the details of the trained model returned by the service	Invoke the particular service fetched by the query with user inputs (configuration parameters, selected dataset, and model type. Retrieve trained model as results.

Fig. 5 Proposed DASE architecture

independent of the organization's application domain or technology stack it uses. Its design has the flexibility to be extended or updated with any new module independently of others. We organize this architecture into five modules (see Fig. 5).

The *Knowledge Base* module contains instance data related to the Organizational Analytics Knowledge Base created utilizing Open Knowledge and ARO as described in Section 3.2.

The *Platform Design Framework* module facilitates the creation of all four components of the Platform Design Specification described in Section 3.3, referring to organizational IT infrastructure, and analytics knowledge.

The *IT infrastructure* shown in Fig. 5 represents existing back-end modules that Data Analytics Platform can access via service calls. This may include machine learning APIs, analytics tools, data sources, workflows and authentication APIs.

The *Tools* layer provides two tools. The *Knowledge Management Interface* is for Knowledge Modelling and Requirement Modelling and will be used to manipulate the objects in the *Knowledge Base*. The *Platform Design Interface* allows us to define elements of PDS through *Platform Design Framework* and it is linked to the *Knowledge Base* through the *Query Engine*.

The *Data Analytics Platform* is the final product of the DASE framework, generated via model driven implementation of the PDS. It is connected to Knowledge Base via Query Engine and to the IT infrastructure via service invocations.

So far we defined each component in the DASE architecture, without mentioning implementation details, to maintain platform independence. For demonstration and evaluation purposes, a prototype implementation of the DASE architecture was realised with following implementation choices:

– Use of Protégé and MarkLogic as the Knowledge Management Interface and Knowledge Base
– Use of Capsifi Jalapeno as the Platform Design Interface and Framework
– Use of R Shiny Framework for Data Analytics Platform Implementation

4.2 Use of Protégé and MarkLogic as the Knowledge Management Interface and Knowledge Base

In our prototype implementation, we used Protégé[2] ontology editing tool as the Knowledge Management Interface for Knowledge Modelling and Requirement Modelling because of its simplicity and availability. Protégé provides a GUI for users to import ontologies available via a file or an URL, explore them and integrate them with local ontologies. Protégé also supports the creation of instance data that represent organizational analytics knowledge, validation of the consistency through its deductive classifiers and infering of new information.

Multiple domain ontologies were imported into Protégé from Open Knowledge to create Organizational Analytics Knowledge Instance Data. These domain knowledge was organized around RVO- The Research Variable Ontology following the approach proposed in Bandara et. al [17]. For requirement modelling we imported ARO into Protégé and modeled suitable *AnalyticsUseCase* and *AnalyticsActivityFlow* with their associated *AnalyticsActions*. Then we wrote *QueryDefinitions, GUIDefinitions and ServiceDefinitions* for all the *AnalyticsActions*, according to the schema of the organizational knowledge base.

We used MarkLogic Triple Store [3] to implement the Knowledge Base module and its inbuilt *Marklogic Query API* as a Query Engine. Marklogic Query API is a REST Client API that can execute SPARQL queries on ontologies and instance data stored in Marklogic Triple Store.

4.3 Use of Capsifi Jalapeno as the Platform Design Interface and Framework

In our prototype, we use the Capsifi[4] Jalapeno tool which provides many of required capabilities of both the Platform Design Interface and the Platform Design Framework module. Capsifi Jalapeno is a commercial tool that utilizes semantic web technologies to model organizational business architecture. It provides high extensibility and the ability to integrate with open knowledge. Because of that, organizations can easily plug-in the Knowledge Base of the DASE architecture to the Jalapeno tool and conduct Platform Modelling.

The Digital Interaction (DI) Framework within the Jalapeno tool is specially designed to support rapid design and development of software applications that reflect knowledge intensive and service-oriented processes in an organization [20, 21]. This DI framework is a dynamic composition of concrete services, set of interactions and underlying information concepts which can be easily converted into execution level workflow code that deliver value to the organization. Once those parts are in place, we can use Jalapeno tool to design all four components of the Platform Design Specification: *GUI Design*, *Service Management* and *Query Management*, *Workflow Design*.
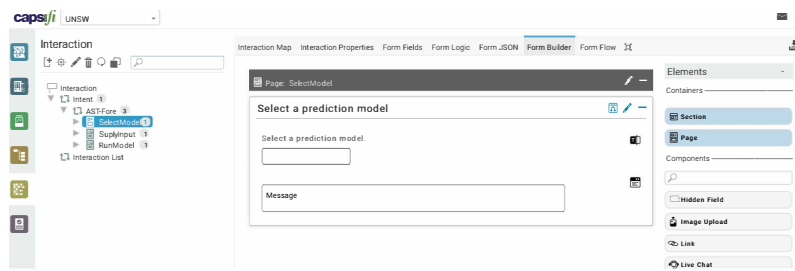
---

[2] https://protege.stanford.edu
[3] http://marklogic.com
[4] https://www.capsifi.com

Fig. 6 GUI Definition for *Select a Model AnalyticsAction* in AAF-Fore
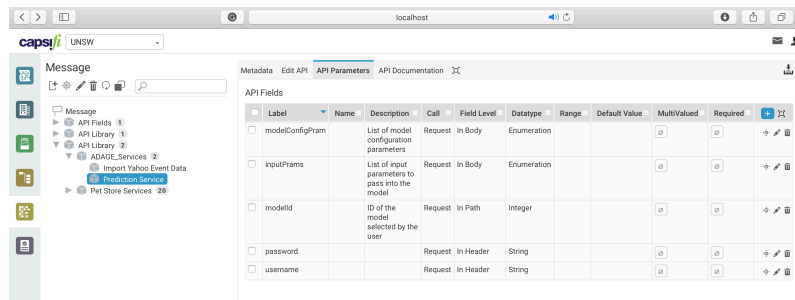

Fig. 7 Service definition for AAF-Fore

4.3.1 GUI Design

To conduct *GUI Design* for each *AnalyticsAction* related to an *AnalyticsUseCase*, we use the Form Builder Interface provided by the Jalapeno platform. This interface has a drag-and-drop feature to design web pages and forms generic component templates such as text fields, drop-down, and radio-buttons. Further, we can link GUI components to the concepts defined in the Knowledge Base, referring to their ontology URI. That way, once deployed, GUI components can be dynamically populated by information fetched via queries. Also, the parameters provided by an analyst in the GUI can be mapped automatically to knowledge base.

Fig. 6 shows a snapshot of Form Builder interface that contains GUI Design, modelled following the *GUI Definition* for 'Select a Model' *AnalyticsAction* related to AAF-Fore in Table 1. Users can drag and drop GUI elements from right hand menu, and organize them to finalize GUIs related to each *AnalyticsAction*.

4.3.2 Service Management

The Jalapeno Message interface is designed to model generic web services. We can use it to conduct *Service Management*, and model any service available in the IT Infrastructure based on ARO *Service Definition*.

Fig. 7 shows how the service related to 'Run Model' *AnalyticsAction* in AAF-Fore (Table 1) is defined in Jalapeno. As shown, the API Fields table allows user to define all input parameters, their data types, and their location in the API request.

User Requirement	Query	Description

Select Model	SELECT DISTINCT ?model ?modelLabel ?modelAttribute ?attributeValue
	WHERE{
	?model ?modelAttribute ?attributeValue.
	?modelAttribute rdf:type ?propertyType.
	OPTIONAL{?attributeValue rdfs:label ?objectLabel.}
	{
	SELECT DISTINCT ?model ?modelLabel
	WHERE {
	?model rdf:type rvo:Model.
	?model rdfs:label ?modelLabel.
	}}
	BIND(
	IF(?propertyType = owl:ObjectProperty, ?objectLabel, ?attributeValue) as ?attributeValue).
	}
		Use subquery inside select statement to identify all models from the Knowledge Base, thein their attributes and attribute values are retrieved. If an attribute value is another object, its label is returned.

Supply Input Data for Model	SELECT DISTINCT ?propertyLabel ?rangeLabel
	WHERE {
	UI1output rvo:independentVariable ?indepVar.
	?indepVar rvo:measuredBy ?measure.
	?measure x:unit ?rangeLabel.
	?measure rdfs:label ?propertyLabel.
	}
		Retrieve all measures and their acceptable value range for independent variables of the model selected by user (Uioutput).

Run Model	SELECT DISTINCT ?serviceLabel
	WHERE {
	?service rdf:type sa-rest:Service.
	?service rdfs:label ?serviceLabel.
	?service rvo:implementation **UI1output** .
	}
		Retrieve service that implements the model selected by user (Uioutput).

Fig. 8 Query written as part of the platform design specification of AAF-Fore

4.3.3 Query Management

When conducting design activities related to the *Query Management*, we refer to the *QueryDefinition* in ARO to understand Query requirement, refer to the schema of the organizational knowledge-base and then write a SPARQL query through the Query Interface provided in Jalapeno tool.

Fig. 8 lists the three queries defined in Jalapeno tool for three *AnalyticsActions* related to AAF-Fore (defined in Table 1). As we used RVO [17] in the prototype knowledge to integrate and represent organizational analytics knowledge, the queries in Fig. 8 are written to cater to the RVO schema [5] using the prefix 'rvo'.

4.3.4 Workflow Design

For *Workflow Design*, we use the Form-Flow interface of the Jalapeno tool. The Form-Flow interface, as shown in Fig. 9, provides a canvas to drag-and-drop modelled GUIs, and services and design executable workflow that reflect any *AnalyticsActivityFlow*. The workflow modelled in Fig. 9 reflects AAF-Fore *AnalyticsActivityFlow* (Fig. 3). Each node and edge in the flow diagram can be edited to map input and output parameters.

4.4 Use of R Shiny Framework for Data Analytics Platform Implementation

As previously illustrated in Fig. 5, the DASE framework's utimate goal is to allow the generation of an executable Data Analytics Platform automatically from the Platform Design Specification in a model-driven fashion. As providing such an
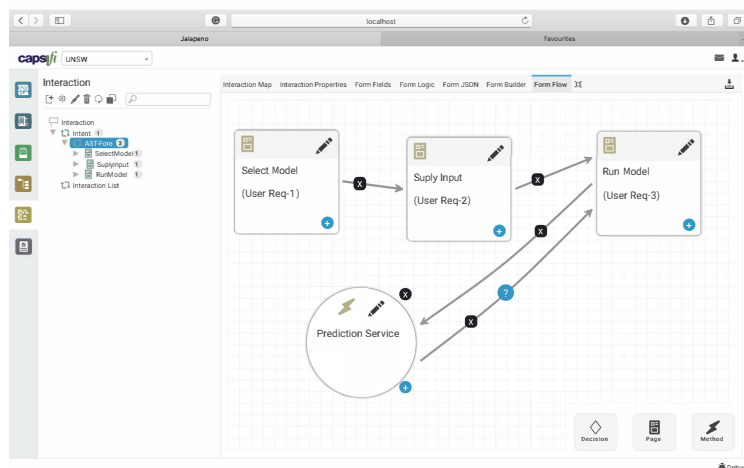
---

[5] http://adage2.cse.unsw.edu.au/rvo/

Fig. 9 Workflow definition for AAF-Fore

Table 2 Example implementations of *AnalyticsUseCases*

Case Study	Analytics Platform Description	*Analytics UseCases* and *ActivityFlow*	Platform Design Specification	Platform Implementation
House Price Prediction [17, 22, 24]	A platform to train house price prediction models and use them to forecast on new datasets	AAF-Fore (Fig. 3), AAF-Train (Fig. 4)	Fig. 6, Fig. 7, Fig. 8, Fig. 9	Fig. 10
Time-series data processing [23, 25, 24]	A platform to acquire time-series data, transform and integrate datasets and store them	Online resource 1 Fig. 1 , Fig. 2	Online resource 1 Fig. 3, Fig. 4	Fig. 11

MDD implementation is out of scope of our work, this prototype manually implements a Data Analytics Platforms using the Shiny software package that is based on R programming language.

Shiny is designed to build interactive web applications utilizing computational capabilities of R. The developers can easily host a web application or build dashboards. Shiny applications are easy to write and do not require web development skills, reducing the workload of the developer. Shiny applications are also extensible with CSS themes, HTML widgets, and Java-Script actions to provide better visualizations and usability to the data analytics platform.

Snapshots of two example platforms we developed to evaluate the framework using R Shiny package will be discussed in detail later (Fig. 10 and 11).

5 Evaluation

To evaluate the proposed framework, we used the prototype implementation described in Section 4 to design and develop two data analytics platforms in different

application domains: house price prediction and time-series data processing. Based on those two case studies we measure the effectiveness of the DASE framework in three dimensions: (1) how well the framework facilitates end-to-end requirement-driven platform design and implementation, (2) how well the knowledge base can support both data analytics platform engineering and operations; and by (3) the usability of the resulting data analytics platform.

Table 2 provides an overview of both case studies, including the figures and references related to associated artifacts. The two *AnalyticsUseCases*, their associated requirements and Platform Design Specifications presented as examples in section 3 and 4 are a part of the house price prediction case study. Please find further details of the house price prediction case study and its implementation in Bandara et. al [17] and Rabhi et. al [22]. For the details of the Time-series data processing case study and its implementation you can refer to Bandara et. al. [23].

5.1 Framework Facilitates End-to-end Requirement-driven Platform Design and Implementation

We used the prototype implementation of the DASE architecture to observe how well the DASE framework can capture and use the organizational analytics requirements related to two case studies for Data Analytics Platform design and implementation. First step in developing Data Analytics Platforms in both case studies was identifying *AnalyticsUseCases* that represent end-user requirements, and modelling related *AnalyticsActivityFlow* and *AnalyticsActions*. Then all the components of the Platform Design Specification necessary to realize those user requirements were modelled in a way that incorporates existing IT infrastructure components. Then we (manually) implemented the two Data Analytics Platforms based on the Platform Design Specifications using R Shiny framework.

When evaluated from the end-user perspective, these platforms were able to support all the *AnalyticsUseCase* functionalities desired by the analyst, confirming that the requirements captured by the DASE framework are sufficient for Data Analytics Platform completion.

One limitation in the DASE framework is that the *AnalyticsActions* supported by ARO are limited by the information available in the Knowledge Base and the services provided by the organization IT infrastructure. In addition, only very simple requirements can be captured by the framework. The framework cannot model complex user requirements or enforce non-functional requirements. Furthermore, in ARO, requirement definitions are limited to text, and *AnalyticsActivityFlow* is represented as a sequence of steps without branching or loops. In future, we plan to extend ARO with more attributes and complex workflow definitions (possibly from an existing workflow language or notation).

5.2 The Knowledge Base Supports Analytics Platform Engineering and Analytics Operations

First we look at how well the Platform Design Specification (PDS) is driven by organizational analytics knowledge accumulated in the Knowledge Base. In both case studies, when developing a PDS, the knowledge base is accessed via the query
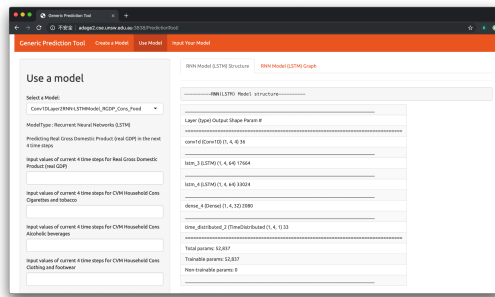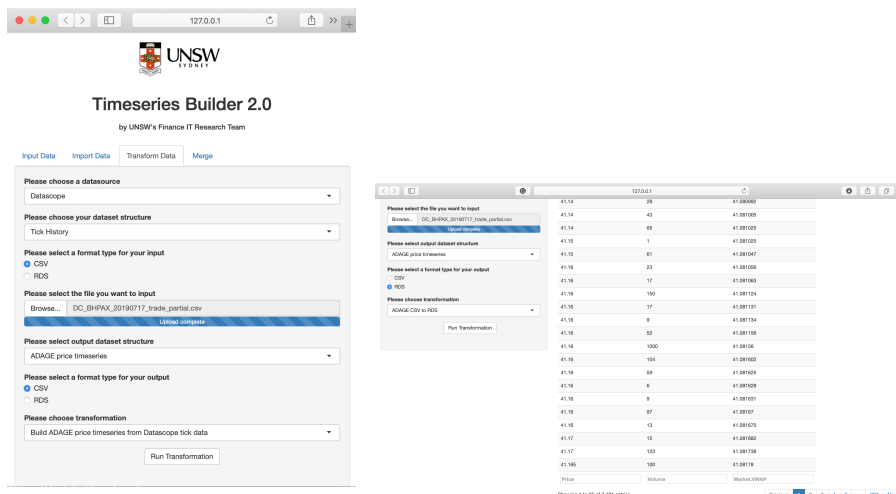
(a)
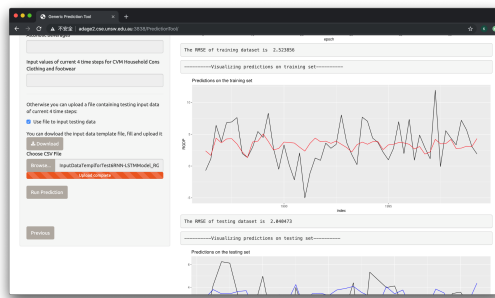
(b)

Fig. 10 Snapshots of the analytics platform developed for house price prediction case study

(a)

(b)

Fig. 11 Snapshots of the analytics platform developed for time series building case study

engine, to map GUI components with the concepts defined in Open Knowledge, and to identify suitable services to fulfill Service Management. When implementing the platforms, the developer integrates the Query Engine with the platform so that each analytics activity conducted by the analyst is assisted by the recommendations provided by queries.

Secondly we observe that once the data analytics platform is implemented, majority of *AnalyticsActions* are assisted by the knowledge base. For example, *Use a Model* step in AAF-Fore implementation (Fig. 10) is supported by a dropdown list populated via a query to show all available models to the analyst. Certain GUI components, such as text boxes to input data for all independent variables in AAF-Fore implementation (Fig. 10), are dynamically created via queries, fetching most relevant information from the knowledge base, based on previous decisions made by the analysts. This ensures that many activities of the data analytics process can be supported by the semantic models in the DASE framework.

We also evaluated how organizations can rapidly update the analytics platform or create a new one for different requirements, with the help of the Knowledge Base. To observe how the DASE framework can handle a change of requirements in the analytics platform, we conducted few hypothetical changes as described below:

– Changing the Knowledge Base: to reflect a change at the organizational knowledge base, we look at how the platform behaves when there is a requirement to add a new prediction model to the House Price Prediction platform. This can be done by updating the Knowledge Base with new model via Protégé. As the platform fetches information via queries dynamically, the new model will be available for the analyst to use immediately in the 'Select a Model' step of the GUI in Fig. 10.
– Changing the IT infrastructure: to study this, we look at implications on the Data Analytics Platform when there is a requirement to add an improved prediction service. To realize it, Service Management and Workflow Design components of the Platform Design Specification have to be updated in Jalapeno to use a new service instead of the current one. Then the platform source code should be updated to help new service invocations.
– Changing the *AnalyticsActivityFlow*: to study this, we add a new step into AAF-Fore for analyst to define expected prediction accuracy, before selecting a model. Then the Knowledge Base needs to be updated with new *AnalyticsActivityFlow* in the place of the current AAF-Fore. The Platform Design Specification needs to be updated for this new step, and the query associated with *Select a Model AnalyticsAction* needs to be updated to filter results by prediction accuracy value. All other components of the Platform Design Specification and the platform software code can be reused.

The main limitation in utilizing the DASE framework for platform engineering is that converting a PDS into a software implementation is conducted manually. The Digital Interaction Framework in Jalapeno tool has sufficient information captured about each artifact of the data analytics platform to be automatically converted into an executable platform in model-driven fashion [20]. Yet the automatic model driven implementation of the Data Analytics Platforms was not explored within the scope of this study.

5.3 The usability of the resulting data analytics platform

We evaluated the usability of the analytics platforms generated by the DASE framework, including its flexibility and ease of maintenance qualities using two Data Analytics Platforms implemented.

First we look into how these platforms reuse existing IT infrastructure and knowledge. As we saw in both case studies, once the analytics platform is generated, as it was designed to follow SOA and workflow principles, it uses existing services and software packages in the organization that are linked to the GUI elements via Platform Design Specification. When the organization updates its underlying services, this modular design allowed the developers to upgrade the analytics platform with minimum effort, reducing the technical debt.

Secondly, with the DASE framework, there is no need for the analyst to identify what process to follow manually. The platforms generated by the DASE framework are designed to support predefined *AnalyticsActivityFlows* linked to a rich Knowledge Base. This means that the analysts are provided with a structured process within the application and directions to follow it, saving their time spent in establishing a manual process.

Lastly, the analytics platform is designed to reduce the cognitive burden of the analysts in the organization. To use the Data Analytics Platforms in both case studies, the analyst doesn't need prior knowledge of existing prediction models, data sources or other resources in the organization. The analyst is able to learn and explore alternatives while conducting the analysis. As the platform interacts seamlessly with the knowledge base and provide relevant information for the analyst to support his decision making, analyst can save time spent on background research or domain understanding. For example, in AAF-Fore implementation, when the analyst selected a model, its performance and structure details are visualized dynamically through the associated query. Also, the platform can filter the alternatives presented for each activity, based on the previous inputs of the analyst. Furthermore, the DASE framework makes many technical details transparent to the analyst. The analysts do not need any programming skills to use the resulting analytics platforms in both case studies.

One drawback we observed is that the data analytics platforms generated by the DASE framework can limit the freedom of data analysts. They would not be able to develop and execute custom scripts or workflows when necessary. That means the DASE framework will be mostly useful for organisations that have frequently used analytics processes that can be modelled in a generic way.

6 Conclusion

This paper presents the DASE framework, a novel approach that enables knowledge-driven data analytics platform engineering. The framework is capable of designing and assisting the implementation of user friendly and easy-to-maintain data analytics platforms that align well with organizational analytics requirements and IT infrastructure. This framework combines semantic web technology with service-oriented and workflow design principles to offer flexibility and extendibility when designing the data analytics platforms, while reducing the technical intensity and cognitive burden of the data analysts.

By implementing and evaluating the DASE framework, we identified multiple limitations and future work we can conduct to improve it. How to generate the Data Analytics Platform utilizing Model-Driven Development paradigm was not explored in this paper. The flow-logic definition in Jalapeno DI Framework can be used as the basis for such work in future, combining with the existing research that explore workflow automations utilizing ontologies [6]. The resulting workflow definitions can be used with an appropriate Model-Driven Development engine to fully automate the Data Analytics Platform software generation.

Furthermore, the DASE framework needs improvements, so that it can harness semantic inference capability. Then the resulting data analytics platforms can provide intelligent recommendations to the analysts, reducing the cognitive burden of analysis further.

Once the DASE framework is implemented within an organization, this organization is going to accumulate a repository of analytics knowledge over time, together with meta-data such as how this knowledge is utilized by the analysts. For example, organizations can generate statistics on how same analytical model performs on different datasets, and what are the best performing analytics algorithms. This information can be used to advance the organizational data analytics platforms by employing them to build value added applications such as meta-learning and eXplainable Artificial Intelligence (XAI) systems specialized for organizational context. Meta-learning is a novel research direction where data analytics platforms are designed to learn and adapt with experience gained by exploiting meta-knowledge extracted in previous analytics tasks or from different domains or problems [26, 27]. XAI is about creating a suite of analytics techniques that can produce more explainable models with high performance, and enable humans to understand, trust, and manage the AI systems [28]. By stressing the importance of having a knowledge base and a service-oriented modular architecture as integral parts of the analytics and AI platforms, the DASE framework is offering a practical solution on how to manage meta-knowledge and semantics related to such complex systems.

References

1. D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, in *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)* (2014)
2. M. Kim, T. Zimmermann, R. DeLine, A. Begel, IEEE Transactions on Software Engineering **44**(11), 1024 (2017)
3. H. Khalajzadeh, M. Abdelrazek, J. Grundy, J.G. Hosking, Q. He, IEEE Transactions on Big Data (2019)
4. D. Crankshaw, J. Gonzalez, P. Bailis, Communications of the ACM **61**(8), 45 (2018)
5. P. Brazdil, C.G. Carrier, C. Soares, R. Vilalta, *Metalearning: Applications to data mining* (Springer Science & Business Media, 2008)
6. Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, J. Kim, in *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2* (AAAI Press, 2007), pp. 1767–1774

7. L. Yao, F.A. Rabhi, Concurrency and Computation: Practice and Experience **27**, 1188 (2015)

8. D. Crankshaw, X. Wang, G. Zhou, M.J. Franklin, J.E. Gonzalez, I. Stoica, in *NSDI* (USENIX Association, 2017), pp. 613–627

9. T. Berners-Lee, J. Hendler, O. Lassila, et al., Scientific American **284**, 28 (2001)

10. J. Cardoso, M. Hepp, M.D. Lytras, *The semantic web: Real-world applications from industry*, vol. 6 (Springer Science & Business Media, 2007)

11. M.D. Lytras, R. García, International Journal of Knowledge and Learning **4**(1), 93 (2008)

12. V.R. Benjamins, J. Davies, R. Baeza-Yates, P. Mika, H. Zaragoza, M. Greaves, J.M. Gomez-Perez, J. Contreras, J. Domingue, D. Fensel, IEEE Intelligent Systems **23** (2008)

13. M. Bandara, F.A. Rabhi, Semantic Web **11**(3), 525 (2020)

14. T.W. Wlodarczyk, C. Rong, B. Jia, L. Cocanu, C.I. Nyulas, M.A. Musen, in *Services (SERVICES-1), 2010 6th World Congress on* (IEEE, 2010), pp. 123–127

15. J.U. Kietz, F. Serban, S. Fischer, B. A., in *European Semantic Web Conference,* (Springer, 2014), pp. 706–720

16. P. Panov, S. Džeroski, L. Soldatova, in *2008 IEEE International Conference on Data Mining Workshops* (IEEE, 2008), pp. 752–760

17. M. Bandara, A. Behnaz, F.A. Rabhi, in *European Semantic Web Conference* (Springer, 2019), pp. 412–426

18. F. Rabhi, M. Bandara, A. Namvar, O. Demirors, in *Service Research and Innovation* (Springer, 2017), pp. 3–17

19. D. Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, M.P. Papazoglou, Information and Software Technology **62**, 42 (2015)

20. M. Bandara, F.A. Rabhi, R. Meymandpour, in *International Conference on Software Process Improvement and Capability Determination* (Springer, 2018), pp. 215–229

21. M. Bandara, F.A. Rabhi, R. Meymandpour, O. Demirors, in *Service Research and Innovation* (Springer, 2018), pp. 108–122

22. F.A. Rabhi, M. Bandara, , K. Lu, S. Dewan, Future Generation Computer Systems **116**, 209 (2021)

23. M. Bandara, F.A. Rabhi, 24th International Conference in Information Visualization, inprint (2020)

24. M. Bandara, A knowledge-driven data analytics solution engineering (dase) framework. Ph.D. thesis, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia (2020)

25. M. Bandara, A. Behnaz, F.A. Rabhi, O. Demirors, in *International Conference on Business Process Management* (Springer, 2018), pp. 543–552

26. C. Lemke, M. Budka, B. Gabrys, Artificial intelligence review **44**, 117 (2015)

27. J. Vanschoren, L. Soldatova, International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010) pp. 31–46 (2010)

28. A.B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Information Fusion **58**, 82 (2020)

Click here to access/download
Supplementary Material
Supplementary.pdf