to appear in Journal of Computer Languages, Elsevier, (c) 2020.

An End-to-End Model-based Approach to Support Big Data Analytics Development

Hourieh Khalajzadeh[1], Andrew Simmons[2], Mohamed Abdelrazek[2], John Grundy[1], John Hosking[3], Qiang He[4]
[1]Faculty of Information Technology, Monash University, Australia
[2]School of Information Technology, Deakin University, Australia
[3]Faculty of Science, University of Auckland, New Zealand
[4]School of Software and Electrical Engineering, Swinburne University of Technology, Australia
{hourieh.khalajzadeh, john.grundy}@monash.edu, {a.simmons, mohamed.abdelrazek}@deakin.edu.au, j.hosking@auckland.ac.nz, qhe@swin.edu.au

Abstract—We present BiDaML, an integrated suite of visual languages and supporting tool to help multidisciplinary teams with the design of big data analytics solutions. BiDaML tool support provides a platform for efficiently producing BiDaML diagrams and facilitating their design, creation, report and code generation. We evaluate BiDaML using two types of evaluations, a theoretical analysis using the "physics of notations", and an empirical study with 1) a group of 12 target end-users and 2) five individual end-users. Participants mostly agreed that BiDaML was straightforward to understand/learn, and prefer BiDaML for supporting complex data analytics solution modeling than other modeling languages.

Keywords—big data analytics, big data modeling, big data toolkits, domain-specific visual languages, multidisciplinary teams, end-user tools

1. Introduction

Using big data analytics to improve decision-making has become a highly active research and practice area [1, 2]. Gartner's technical professional advice [3] recommends six stages for machine learning applications development: classifying the problem, acquiring data, processing data, modeling the problem, validation and execution, and finally deployment. Traditionally, advanced machine learning knowledge and experience of complex data science toolsets were required for data analytics applications. Emerging analytics approaches seek to automate many of these steps in model building and its application, making machine learning technology more accessible to those who lack deep quantitative analysis and tool building skills [4].

Recently, a number of data analytics and machine learning tools have become popular, providing packaged data sourcing, integration, analysis and visualization toolkits oriented towards end-users. These include Azure ML Studio [5], Amazon AWS ML [6], Google Cloud ML [7], and BigML [8], as reviewed in [9]. Many of these tools do not require programming language knowledge and are based on simple drag-and-drop interfaces. However, they mostly focus on the machine learning algorithms and sometimes one-click deployment, but lack domain knowledge and business problem capture, modeling, traceability to the solution and validation of the solution against the problem. They also lack an explanation of the model from an end-user perspective.

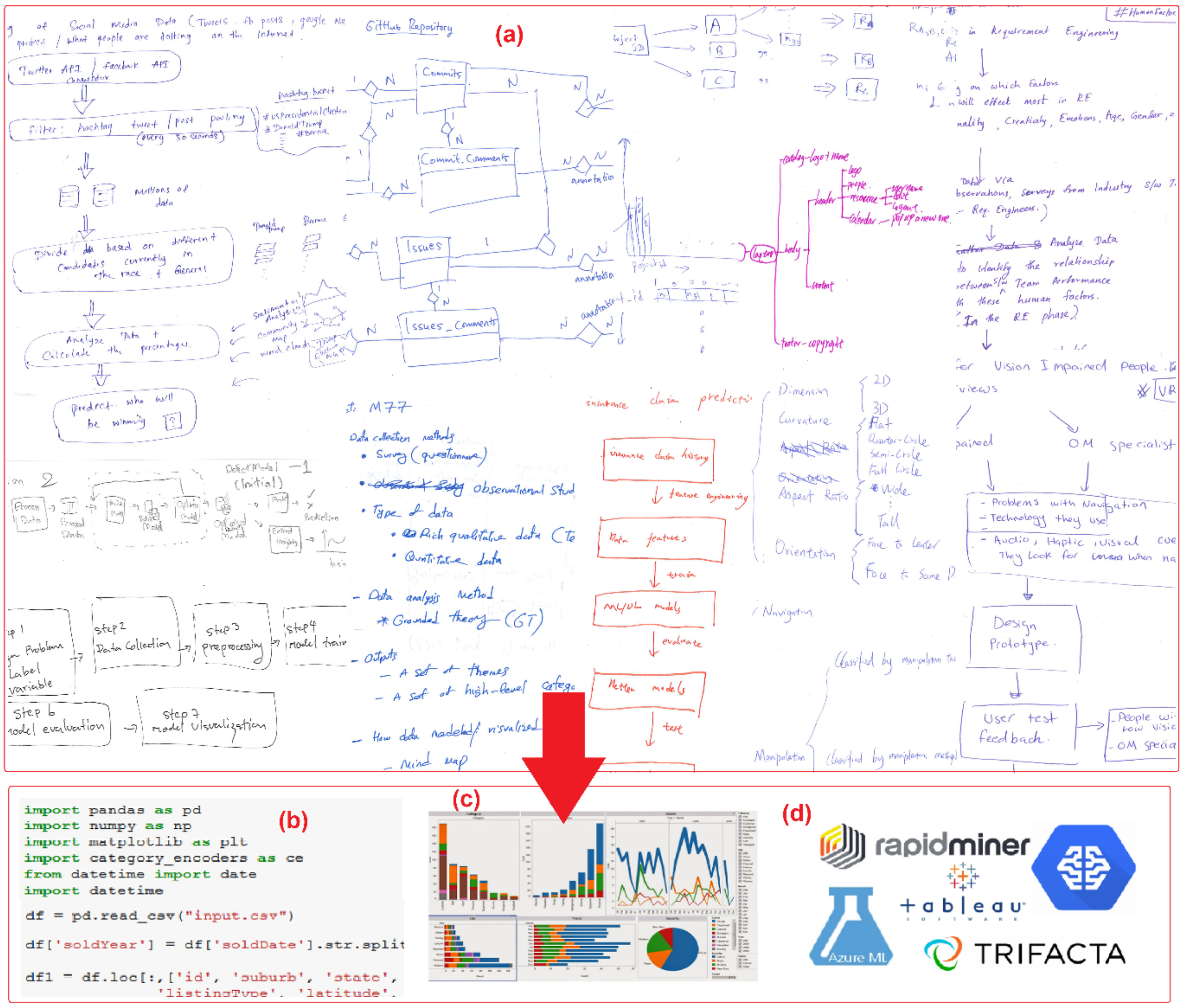
Data analytics and machine learning steps need to be more tightly connected to the control and management of business and requirements engineering processes [10]. However, the primary focus of most current big data analytics tools and technologies is on storage, processing, and in particular data analysis and machine learning tasks. Current data analytics tools rarely focus on the improvement of end-to-end processes [11]. To address this issue, better integration of data science, data technology, and process science is in demand [11]. Data science approaches tend to be process-agonistic whereas process science approaches tend to be model-driven without considering the evidence hidden in the data. Scalable process mining analytics need to be brought to big data toolkits while enabling them to be easily tailored to accommodate domain-specific requirements [11]. Tools filling these gaps would be very useful for multidisciplinary teams where a range of complicated steps and users with different skillsets are involved. E.g. to empower domain experts and business managers during the discovery and exploration phase with the tools to model the problem, extract insights/patterns, and design predictive/clustering models (if feasible) before they involve other stakeholders such as data scientists and software engineers.

We present our novel approach to addressing this problem - Big Data Analytics Modeling Languages (BiDaML). BiDaML, extended from [10], is a set of domain-specific visual models at differing levels of abstraction, to capture and refine requirements and specify different parts of the data analytics process. Through these domain-specific visual languages (DSVLs), we aim to make data analytics design more accessible to multidisciplinary teams and facilitate dialogues with expert data scientists and software engineers. BiDaML provides better tool support and collaboration between different users while improving the speed of implementing data analytics solutions. This work is extended from [10] mainly in three ways: 1) The diagrams and notations are extended and are significantly different in this new version, i.e., the number and type of diagrams, number and type of notations, the generated outputs, the evaluations, etc. 2) The Physics of Notations (PoN) evaluation done in [10] is significantly different from the comprehensive PoN conducted for this paper that led to essential changes to the notations. 3) A completely new user study with a group of target end-users including data scientists and software engineers was conducted and the results were analyzed and included in the extended version. In the group study, the notations and models are compared with the existing modeling frameworks. We follow two objectives in this paper: 1) The major objective is to present domain specific modeling languages and a support tool for designing, documenting, and communicating data analytics solutions. 2) The secondary objective that has not been elaborated comprehensively in this paper is generating code, recommending solutions, and reusing the existing solutions.

The rest of the paper is organized as follows. In Section 2, the background and motivation of this research are described with a real-world example of a property price prediction system development. Our approach is discussed in Section 3 and evaluated in Section 4. Limitations and future directions are discussed in Section 5. A comprehensive comparison to related work is presented in Section 6. Finally, we draw conclusions and discuss key future directions in Section 7. BiDaML tool, data and evaluation results can be accessed publicly from [18].

2. Motivating Example

In this section, we discuss data analytics steps as well as different types of communication between users in a data analytics project. A real property price prediction example is then demonstrated to reflect the issues and some key challenges in the process of data analytics.



```
import pandas as pd
import numpy as np
import matplotlib as plt
import category_encoders as ce
from datetime import date
import datetime

df = pd.read_csv("input.csv")

df['soldYear'] = df['soldDate'].str.split

df1 = df.loc[:,['id', 'suburb', 'state',
        'listingType', 'latitude',
```

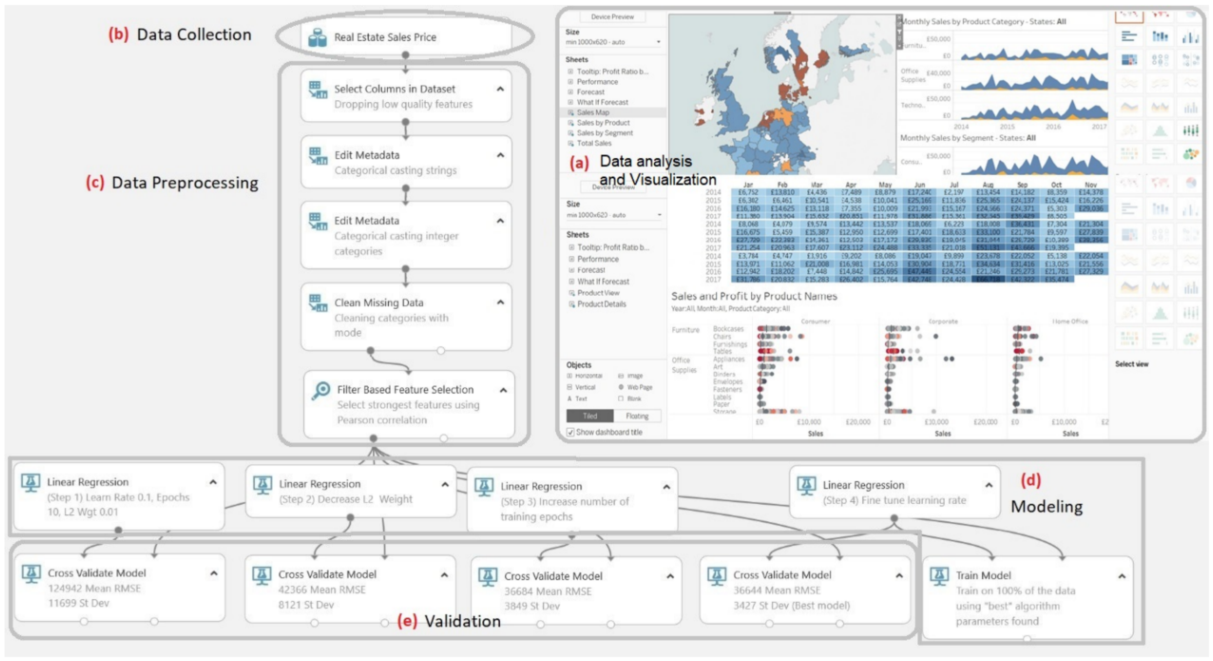
Figure 1. How data scientists currently design their big data analytics solutions. Examples (top) captured during our group user study.

2.1 *Data Analytics Process Steps*

The key steps that data scientists take to design their solutions are illustrated in Figure 1. Business owners, business analysts, data analysts and data scientists need to conduct meetings and interviews with domain experts and users and therefore, generate many handwritten/ad-hoc notations and documentations and meeting notes. Then, they acquire required datasets from different resources, obtain access to government information, integrate all datasets with different formats by communicating with each other to discuss the analyses (b) writing scripts, (c) analyzing and visualizing data, and (d) finally use different tools to design their approaches and develop their models. As shown in Figure 1(a), based on the data we collected during our group user study, there is, in fact, no unified language for these stakeholders (including non-technical domain experts and business managers through to technical data scientists and software engineers with mostly no domain knowledge) to facilitate communication throughout the project and also no unified tool to record, capture and document all the steps involved. Therefore, the higher-level stakeholders need to wait for the technical team to derive usable and deployable models to obtain updates on their progress. In this section, we show an example of a property price prediction system development to discuss some of the problems the users involved in the project face during the solution design process.

2.2 *Example: Property Price Prediction*

Consider a situation where data analytics is intended to be used for predicting property prices, as shown in Figure 2. In this scenario, a real estate agency wants to improve agents' focus and outcomes by looking for patterns in a large amount of real estate, government and financial data. In order to solve this problem using conventional approaches, such end-users in multidisciplinary teams would need to have a basic knowledge of data analytics programming languages such as Python and R and also a basic knowledge of data science. Due to the lack of necessary knowledge, the company employs a technical team of software engineers and data scientists to work on the problem. The user needs to upload a dataset, choose features based on the quality of the features, apply data type casting and data cleansing, and finally filter the features to build the dataset as an input for the prediction model. Then the user chooses the best model, algorithm, and validation method and adapts the characteristics based on problem requirements.



(a) Data analysis and Visualization
(b) Data Collection
Real Estate Sales Price
(c) Data Preprocessing
Select Columns in Dataset
Dropping low quality features
Edit Metadata
Categorical casting strings
Edit Metadata
Categorical casting integer categories
Clean Missing Data
Cleaning categories with mode
Filter Based Feature Selection
Select strongest features using Pearson correlation
Linear Regression
(Step 1) Learn Rate 0.1, Epochs 10, L2 Wgt 0.01
Linear Regression
(Step 2) Decrease L2 Weight
Linear Regression
(Step 3) Increase number of training epochs
Linear Regression
(Step 4) Fine tune learning rate
(d) Modeling
Cross Validate Model
124942 Mean RMSE
11699 St Dev
Cross Validate Model
42366 Mean RMSE
8121 St Dev
Cross Validate Model
36684 Mean RMSE
3849 St Dev
Cross Validate Model
36644 Mean RMSE
3427 St Dev (Best model)
Train Model
Train on 100% of the data using "best" algorithm parameters found
(e) Validation
Sales and Profit by Product Names
Monthly Sales by Product Category
Monthly Sales by Segment

Figure 2. Property price prediction using Tableau and Azure ML Studio

The team decides to use Azure ML Studio to do some initial analyses on the Ames Housing dataset [12] before purchasing data from a data provider. In Figure 2, (a) the user analyses and visualizes data in Tableau, then using ML Azure Studio (b) drags the "Real Estate Sales Price" module to upload input data, (c) adds "Select Columns in Dataset", "Edit Metadata", "Clean Missing Data", and "Filter Based Feature Selection" modules to prepare and clean data, (d) "Linear Regression" and "Train Model" modules to apply linear regression and train the model, and finally (e) "Cross Validate Model" module to validate the model. These steps are low-level machine learning operations and the user needs to have data science knowledge to choose the modules properly and change their parameters. If the user wants to use a pre-processing method or apply a model that does not exist in the list of modules, knowledge of programming languages such as Python and R is required to embed code in order to improve steps and add features.

The new challenge that arises is that the company realizes that the team lacks sufficient understanding of important domain knowledge; therefore, appoints a senior real estate agent, highly experienced with the nuances in the domain, to help the team build the analytics solution. The team struggles with a lack of a common dialect between engineers, scientists and domain experts. Eventually, the team realizes that the solution is not ready due to issues in the available dataset that need to be rectified and it takes a long time for the team to finally design and build a working analytics solution. A few months later, after the technical team has been disbanded, the company realizes that adding new features, e.g. public transports in the area, or building a new model to predict who would be willing to sell their property in a given suburb might be beneficial. In addition, the company observes degradation in the performance of the existing property price prediction model, which needs to be updated. The company decides to recruit a new team to apply the new capability, however, they struggle to reuse and update the existing solution due to lack of business knowledge and lack of clear documentation or an integrated framework to allow them to reuse or integrate models.

2.3 Key Challenges

As illustrated above in Figure 2, there is no traceability back to the business needs/requirements that triggered the project. Furthermore, communicating and reusing existing big data analytics information and models is a challenge for many companies new to data analytics. Users need to be able to collaborate with each other through different views and aspects of the problem and possible solutions. Current practices and tools do not cover most activities of data analytics analysis and design, especially the critical business requirements. Most current tools focus on low-level data analytics process design, coding and visualization of results and they mostly assume data is in a form amenable to processing. In reality, most data items are in different formats and are not clean, integrated or usable by the machine learning models and great effort is needed to source the data, integrate, harmonize, pre-process and cleanse it. Moreover, only a few of the machine learning tools offer the ability for the data science expert to embed new code and expand algorithms and provide visualizations for their needs.

Data processing and machine learning tasks are only a small component in the building blocks necessary to build real-world deployable data analytics systems [13]. As Figure 3 illustrates, these tasks cover a small part of data and machine learning operations and model deployment. At the same time, conventional business and management modeling tools do not usually support many key data analytics steps including data pre-processing and machine learning steps. There is a need to capture the high-level goals and requirements for different users such as domain expert, business analyst, data analyst, data scientist, software engineer, and customers and relate them to low-level steps and capture details such as different tasks for different users, requirements, objectives, etc. Finally, embedding code and changing features based on the users' requirements require data science and programming knowledge.

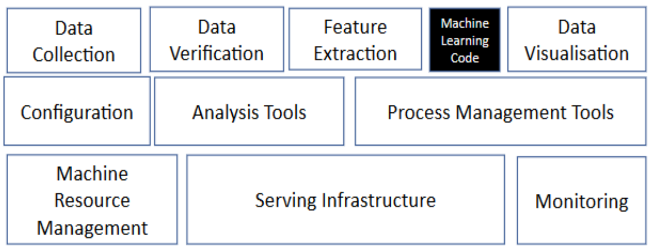


Figure 3. Data analytics steps (adapted from [13])

3 OUR APPROACH

As the example in Section 2 shows, many current big data analytics tools provide only low-level data science solution design, despite many other steps being involved in solution development. Therefore, a high-level presentation of the steps to capture, represent and communicate the business requirements analysis and design, data pre-processing, high-level data analysis process, solution deployment, and data visualization is required.

3.1 BiDaML Visual Language

We present BiDaML, a set of domain-specific visual languages using five diagram types at different levels of abstraction to support key aspects of big data analytics. These five diagram types cover the whole data analytics software development life cycle from higher-level requirement analysis and problem definition through the low-level deployment of the final product. These five diagrammatic types are:

- ***Brainstorming diagram*** which provides an overview of a data analytics project and all the tasks and sub-tasks that are involved in designing the solution at a very high level. Users can include comments and extra information for the other stakeholders;
- ***Process diagram*** which specifies the analytics processes/steps including key details related to the participants (individuals and organizations), operations, and conditions in a data analytics project capturing details from a high-level to a lower-level;
- ***Technique diagrams*** which show the step by step procedures and processes for each sub-task in the brainstorming and process diagrams at a low level of abstraction. They show what techniques have been used or are planned to be used and whether they were successful or there were any issues;

- ***Data diagrams*** which document the data and artifacts that are produced in each of the above diagrams at a low level, i.e. the technical AI-based layer. They also define in detail the outputs associated with different tasks, e.g. output information, reports, results, visualizations, and outcomes;
- ***Deployment diagram*** which depicts the run-time configuration, i.e. the system hardware, the software installed on it, and the middleware connecting different machines to each other for development related tasks.

Figure 4 shows an overview of the BiDaML approach on the left side, and how these diagrams are created and connected from high level to low level for the Property Price Problem, in the right side. Different users i.e., domain experts, business owners, business analysts, data analysts/scientists, and software engineers create and modify the diagrams in a collaborative way via the BiDaML tool. In the property price prediction example, a brainstorming diagram is defined for the project to assist all the stakeholders to communicate and specify the high-level objectives, targets, tasks, etc. Then, at a lower level to include more details and specify the organizations and participants, we use a process diagram, which is also a unique diagram for each project. Every task in brainstorming and process diagrams can be further extended by technique and data diagrams at different levels. As many technique and data diagrams as is necessary can be defined by data scientists and software engineers to communicate their findings and progress with the other members of the team in a more communicable language than the informality of the notes of Figure 1. Finally, a deployment diagram, defined for development related tasks in the data analytics problem, models the deployment related details at a low level. A deployment diagram is unique for each project, and reuses the notations and elements defined in the previous diagrams to communicate the deployment strategies used by software engineers. With the users having defined this set of diagrams, the tool can then generate and inform users of reports, documentations, Python code, and API access code.

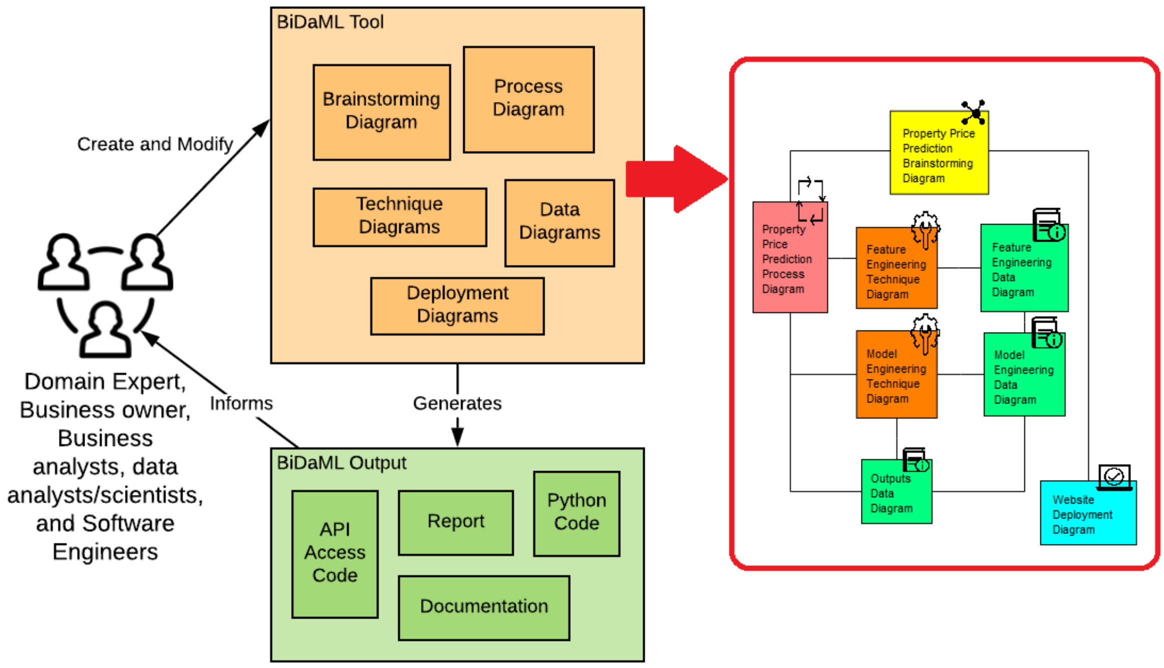

BiDaML Tool
Brainstorming Diagram
Process Diagram
Technique Diagrams
Data Diagrams
Deployment Diagrams
Create and Modify
Domain Expert, Business owner, Business analysts, data analysts/scientists, and Software Engineers
Informs
Generates
BiDaML Output
API Access Code
Report
Python Code
Documentation
Property Price Prediction Brainstorming Diagram
Property Price Prediction Process Diagram
Feature Engineering Technique Diagram
Feature Engineering Data Diagram
Model Engineering Technique Diagram
Model Engineering Data Diagram
Outputs Data Diagram
Website Deployment Diagram

Figure 4. BiDaML overview (left), and overview of the diagrams created for the property price problem (right).

3.1.1 Brainstorming Diagram

A data analytics brainstorming diagram covers the entirety of a data analytics project expressed at a high-level. There are no rules as to how abstractly or explicitly a context is expanded. The diagram overviews a data analytics project in terms of the specific problem it is associated with, and the tasks and subtasks to solve the specific problem. It supports interactive brainstorming and collaboration between interdisciplinary team members to identify key aspects of a data analytics project such as its requirements implications, analytical methodologies, and specific tasks.

Figure 5 shows the visual notation we use for a brainstorming diagram. It comprises a red hexagon icon representing the data analytics problem and yellow ellipses to show to-do tasks with which the problem is associated. The red hexagon is intended to emphasize the problem, e.g "Property Price Prediction" in our example, while the yellow ellipses provide a notational association with Post-it notes. High-level tasks connected directly to the problem are automatically changed to bright orange post-it notes with fixed pins to highlight their importance. For our motivating example, "price prediction feature engineering" is one of the high-level tasks which is further broken down to “clean data”, “impute missing value”, “add new features”, “choose features”, etc, as lower level to-do tasks. Specific information to share between the members is captured as comment icons. Finally, input icons are used in case specific data items need to be connected to any of the tasks.

We characterize the building blocks of a data analytics software system into four groups: Domain and business-related activities (BusinessOps); data-related activities (DataOps); artificial intelligence and ML-related activities (AIOps); and development and deployment activities (DevOps). BusinessOps covers domain and business knowledge and requirement gathering, modeling, and analysis. DataOps includes data collection/ingestion, data validation cleansing, wrangling, filtering, union, merge, etc. AIOps covers feature engineering and model selection, model training and tuning. Finally, DevOps covers model integration and deployment, monitoring and serving infrastructure. Blue dashed ellipses (operation separators) are used to group these operations in order to make it easier for the different types of stakeholders to focus on their specific aspects of the problem and to see how this fits in with the overall system. An undirected line is then used to connect the problem to the tasks and tasks to the other tasks, comments, and inputs.

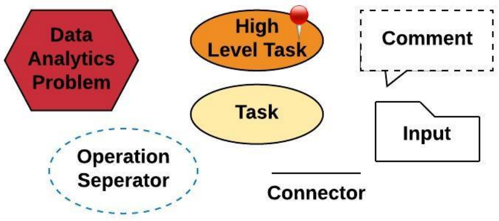
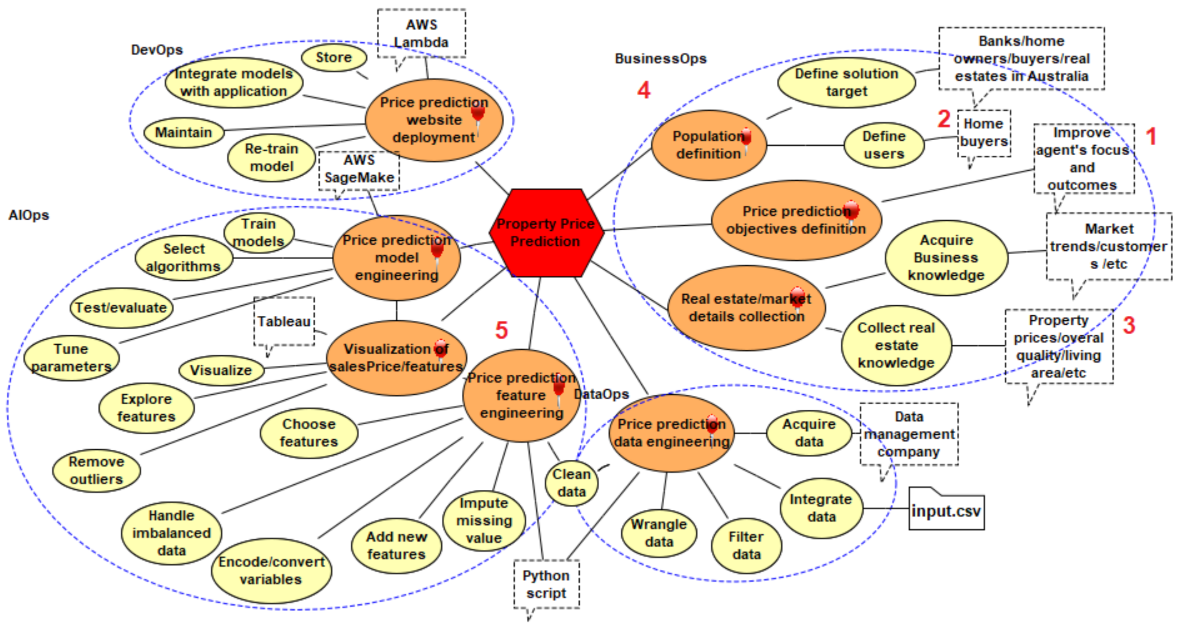


Figure 5. Brainstorming diagram notational elements.



DevOps
AWS Lambda
Store
Integrate models with application
Price prediction website deployment
Maintain
Re-train model
AWS SageMake
BusinessOps
Banks/home owners/buyers/real estates in Australia
Define solution target
4
Population definition
Define users
2
Home buyers
Improve agent's focus and outcomes
1
Price prediction objectives definition
Acquire Business knowledge
Market trends/customers /etc
Real estate/market details collection
Collect real estate knowledge
Property prices/overal quality/living area/etc
3
AIOps
Train models
Select algorithms
Price prediction model engineering
Property Price Prediction
Test/evaluate
Tableau
Tune parameters
Visualize
Visualization of salesPrice/features
5
Price prediction feature engineering
DataOps
Explore features
Choose features
Price prediction data engineering
Acquire data
Data management company
Remove outliers
Clean data
Integrate data
input.csv
Handle imbalanced data
Impute missing value
Wrangle data
Filter data
Add new features
Encode/convert variables
Python script

Figure 6. A brainstorming diagram example for the property price problem

Figure 6 depicts a high-level brainstorming diagram for our property price prediction example from Section 2. As some insights, from this figure we can see that:

1. The objective of the data analytics problem is to improve the agent’s focus and outcomes;
2. The project’s target population is home buyers;
3. Its implications are the identification of relationships between the sales price and features such as overall quality, living area, etc;
4. Population definition, price prediction objectives definition, and real estate/market details collection are the high level tasks to be conducted as BusinessOps before acquiring and accessing data; and
5. In order to conduct price prediction feature engineering, clean data, impute missing value, add new features, and choose features, are some of the tasks to complete.

3.1.2 Process Diagram

The key business processes, organizations, and stakeholders involved as well as the project flow in a data analytics application are shown in a process diagram, whose basic notation is shown in Figure 7. We have adapted and simplified the Business Process Modeling Notation (BPMN) [14] to specify big data analytics processes at several levels of abstraction. Process diagrams support business process management, for technical users such as data analysts, data scientists, and software engineers as well as non-technical users such as domain experts, business users and customers, by providing a notation that is intuitive to business users, yet able to represent complex process semantics.

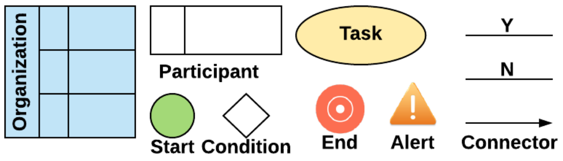


Figure 7. Process diagram notational elements.

In this diagram type, we use different "pools" for different organizations and different "swim lanes" for the people involved in the process within the same organization. For example, "Business analyst/owner", "Data analyst", "Data scientist", and "Software engineer" are the users in "data analytics solution provider", as one of the organizations involved in our motivating example. To facilitate cognitive integration of the different BiDaML diagram types, tasks are reused from the brainstorming diagram and any changes to tasks in one diagram are automatically propagated by the tool to the others. There might be some tasks not being used in the process diagram or some tasks being created in the process diagram that were not initially in the brainstorming diagram. The reason some of the tasks in the brainstorming diagram might not be followed in the process diagram is that they are not related to the other participants or organizations or they are high level tasks and have already been resolved in the previous steps. Also, some new lower-level tasks might be expanded from high level tasks and added to the process diagram that were not initially considered in the high-level brainstorming diagram. A green circle is used to show the start of the project and a target sign to show the ending point of the project. For instance, a data analytics project starts for a real estate manager when they have some data to be analyzed, and it finishes when users get predictions from the website and follow up with the business manager. Tasks, conditions, and alerts trigger other events and redirect the process to the other users in the same or different pool. Diamonds show different decision points that can adjust the path based on conditions, and the unexpected events that can change the process at any step. Lines with yes (Y) and no (N) signs are used to connect from conditions to the appropriate tasks and directed arrows are used to connect the tasks to other tasks, conditions, alerts and the ending point based on the order and priorities. Except from the task notation that is re-used from the brainstorming diagram, the other process diagram notations reuse symbols and syntax from BPMN where possible, to let users who are familiar with BPMN symbols and syntax to more easily remember and re-use the symbols.

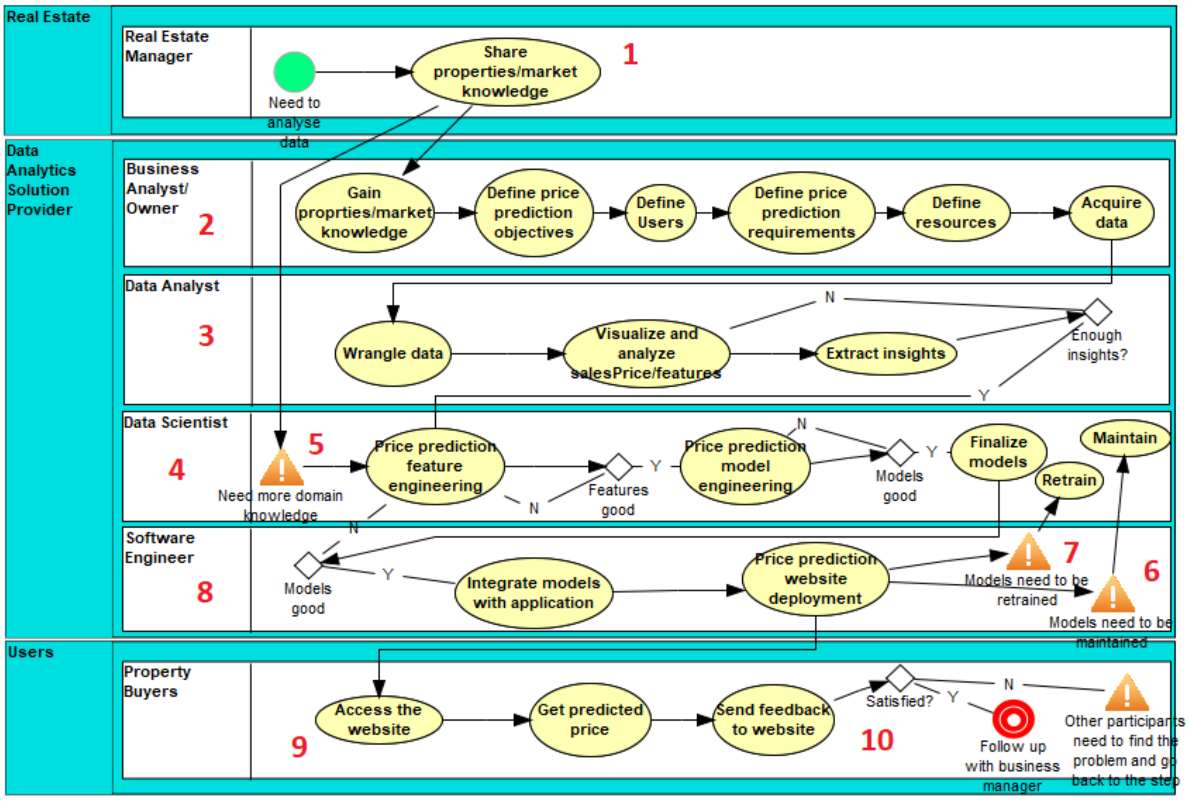

Real Estate
Real Estate Manager
Need to analyse data
Share properties/market knowledge
1
Data Analytics Solution Provider
Business Analyst/ Owner
2
Gain proprties/market knowledge
Define price prediction objectives
Define Users
Define price prediction requirements
Define resources
Acquire data
Data Analyst
3
Wrangle data
Visualize and analyze salesPrice/features
Extract insights
Enough insights?
N
Y
Data Scientist
4
5
Need more domain knowledge
Price prediction feature engineering
Features good
Price prediction model engineering
Models good
Finalize models
Maintain
Retrain
Software Engineer
8
Models good
Integrate models with application
Price prediction website deployment
7
Models need to be retrained
6
Models need to be maintained
Users
Property Buyers
9
Access the website
Get predicted price
Send feedback to website
10
Satisfied?
Follow up with business manager
Other participants need to find the problem and go back to the step

Figure 8. A process diagram example for the property price problem

A high-level process diagram for our property price prediction example is shown in Figure 8. In this, 1) a real estate manager from a real estate organization shares background knowledge with the business manager/analyst. 2) The business analyst /owner then defines the price prediction objectives, requirements, users, etc and acquires data from the related companies and organizations based on the budget and requirements and passes the dataset along with the domain knowledge to the data analysts/scientists team. 3) A data analyst applies DataOps, extracts insights, and communicates the insights with the 4) data scientists to allow them to apply AIOps and share the finalized models with software engineers. These collaborations between domain expert, business manager/analysts, data analysts/scientists, and software engineers do not happen only once; they might 5) return to previous steps to collect more domain knowledge and insights based on an unexpected event, 6) maintain and 7) retrain models, in case the condition is not fulfilled. Once models are finalized, 8) software engineers can deploy the models and 9) users can access the property price prediction website to send requests. Participants need to be involved later to retrain, maintain, and improve the models based on the 10) users' feedback. In this case, an alert needs to be triggered and the business managers must find the responsible participant to improve the process and consequently the outcome.

3.1.3 Technique Diagram

Data analytics technique diagrams extend the brainstorming diagram to low-level details specific to different big data analytics tasks. For every task, the process is broken down to the specific stages and the technique used to solve a specific task is specified. Figure 9 shows the technique diagram notation. Tasks and alert symbols are reused from the brainstorming and process diagrams; a green hexagon sign is used to specify the techniques used and a tick sign to specify that a technique leads to successful results while alerts rectify the issues/challenges faced that need to be taken into account. The green color is associated with the solution and the reason for using a hexagon as the shape is to include a variety of different shapes based on the physics of notations recommendations. Tick and alert signs are automatically associated with success and issues in the users' minds with no need to further remember new signs. Directed arrows are used to connect tasks to the techniques used/planned to be used for them, and the techniques to the other techniques and outcomes.

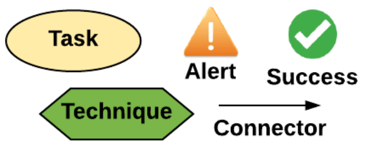


Figure 9. Technique diagram notational elements

Two different technique diagrams for data wrangling and price prediction model engineering are shown in Figure 10. In Figure 10 (a), (a1) sold properties data needs to be purchased from a data provider, and some additional datasets collected from government websites, or (a2) Python pandas library can be used for parsing input file, data reshaping, and data integration, however, it needs basic data science and programming knowledge. Moreover, in Figure 10 (b), (b1) XGBoost and (b2) Lasso regression algorithms are used to train initial models and finally (b3) the average of the predictions from these two models is used as the final prediction. A (b4) cross-validation technique is used for the test/evaluate task. We can create such diagrams for every task and sub-task in brainstorming and process diagrams.

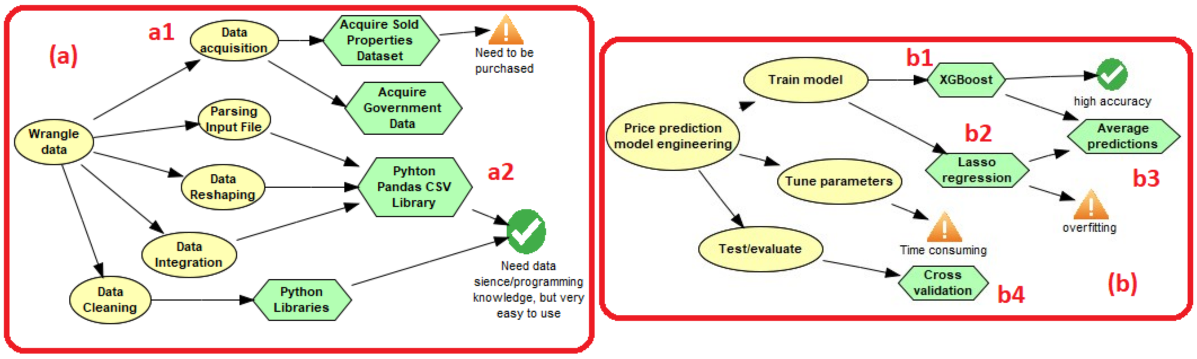


Figure 10. Technique diagram examples for a) data wrangling and b) price prediction model engineering tasks in the property price problem

3.1.4 *Data Diagram*

To document the data and artifacts consumed and produced in different phases described by each of the above diagrams, a low-level data diagram is presented, using the notation in Figure 11. Data diagrams support the design of data and artifacts collection processes. They represent the structured and semi-structured data items involved in the data analytics project in different steps and the data items, reports, models and parameters generated throughout the project. Keeping track of data helps the explainability of black-box machine learning models. A high-level data diagram can be represented by connecting the low-level diagrams for different BusinessOps, DataOps, AIOps, and DevOps.

In Figure 11, tasks are reused from brainstorming and process diagrams and new notations are defined for different types of data and artifacts, e.g. clipboards to represent reports and dashed grey rectangles to represent information. Data items are shown by green file types to reflect excel/datasets and blue file types to reflect source code/models. Directed arrows are used to connect tasks to the data items and artifacts and vice versa.

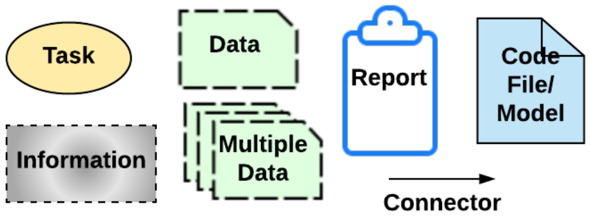


Figure 11. Data diagram notational elements.

Three different data diagrams for "price prediction feature engineering", "property price model engineering" and "integrate models with applications" tasks for our price prediction problem are created and shown in Figure 12. Here, data and artifacts related to all tasks in brainstorming and process diagrams are connected to different data entities. In this case, different data items, features, outliers, the algorithms used, parameters related to these algorithms, models created based on different algorithms, the evaluation metrics used for the models, and finally, the expected outputs are captured. Data and artifacts produced for all the other tasks can be detailed and depicted with different data diagrams.

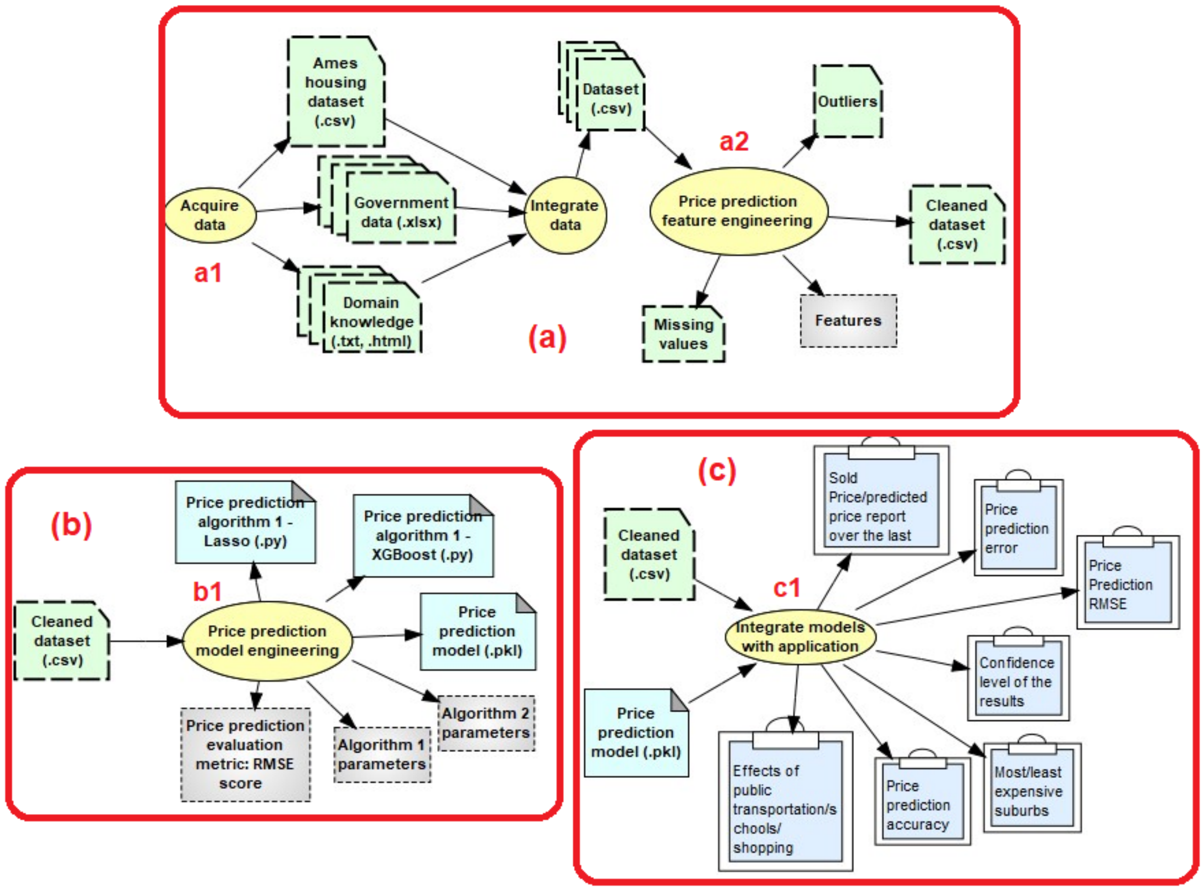

(a)
a1
Acquire data
Ames housing dataset (.csv)
Government data (.xlsx)
Domain knowledge (.txt, .html)
Integrate data
Dataset (.csv)
a2
Price prediction feature engineering
Outliers
Cleaned dataset (.csv)
Missing values
Features
(b)
b1
Cleaned dataset (.csv)
Price prediction model engineering
Price prediction algorithm 1 - Lasso (.py)
Price prediction algorithm 1 - XGBoost (.py)
Price prediction model (.pkl)
Price prediction evaluation metric: RMSE score
Algorithm 1 parameters
Algorithm 2 parameters
(c)
c1
Cleaned dataset (.csv)
Price prediction model (.pkl)
Integrate models with application
Sold Price/predicted price report over the last
Price prediction error
Price Prediction RMSE
Confidence level of the results
Most/least expensive suburbs
Price prediction accuracy
Effects of public transportation/schools/ shopping

Figure 12. Data diagram examples for a) price prediction feature engineering, b) model engineering and c) integrate models with application tasks in the property price problem

For example, in Figure 12 (a), (a1) Ames housing dataset in CSV format, government data in XLSX format, and domain knowledge in TXT and HTML formats are collected during the "acquire data" task and need to be integrated through the "integrate data" task. Also, (a2) the "price prediction feature engineering" task receives a dataset in CSV format and generates outliers, a cleaned dataset, features, and missing values. In Figure 12 (b), the "price prediction model engineering" task receives the cleaned dataset and generates Python codes, models, parameters, and scores. Finally, in Figure 12 (c), we can see (c1) the outputs and reports that can be extracted using current techniques and data items, such as predicted price versus sale price report, and accuracy, as well as analysis of property prices in different areas based on nearby schools, public transport, and shopping centers.

3.1.5 Deployment Diagram

Since the deployment phase follows the same rules as the deployment process in software development, we had initially adopted the deployment diagram concepts from the context of Unified Modeling Language (UML) [15]. However, as big data analysis requires a focus on distributed cloud platforms, services, and frameworks rather than individual nodes/devices, the visual notation itself draws inspiration from system layer diagrams to visualize the technology stack. The BiDaML deployment diagram notation is shown in Figure 13. Most of the notations are reused from the other BiDaML diagrams.

In a deployment diagram, rectangles indicate the software artifacts and the deployment components and an application logo is used to specify the website and application. Undirected lines from one item to another item specify the relationships between elements.

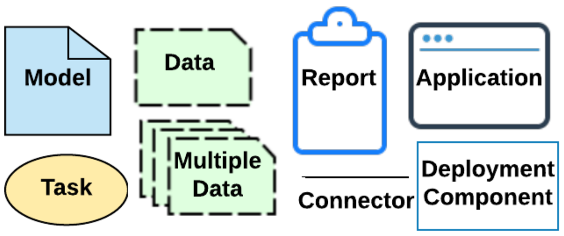


Figure 13. Deployment diagram notational elements.

Figure 14 shows an example deployment diagram for our property price prediction problem. In this diagram we can see that 1) the browser is running on the client system, 2) the sold properties/suburbs reports and prediction price results will be displayed in the browser using HTML5, 3) AWS Lambda is used to respond to website price prediction requests from clients and run the model, 4) government suburbs data is stored in the real estate MySQL DB, 5) model training tasks are running on AWS Spot instances, and so on. Note the use of vertical stacking in BiDaML to denote layers of the technology stack, e.g. AWS Cloud is used to run AWS Spot Instances that support the Model Training environment. This is because visual containment (e.g. nested execution environments in UML deployment diagrams) can lead to diagrams that are hard to read and use when there are many levels of containment involved [16]. Figure 14 also displays tasks directly on the deployment diagram, an optional feature of BiDaML deployment notation designed to help clarify how the infrastructure will support key tasks, e.g. the Model Training environment exists to support the tasks of training/re-training models on demand, and to make it easier to relate the deployment diagram back to the other diagrams.

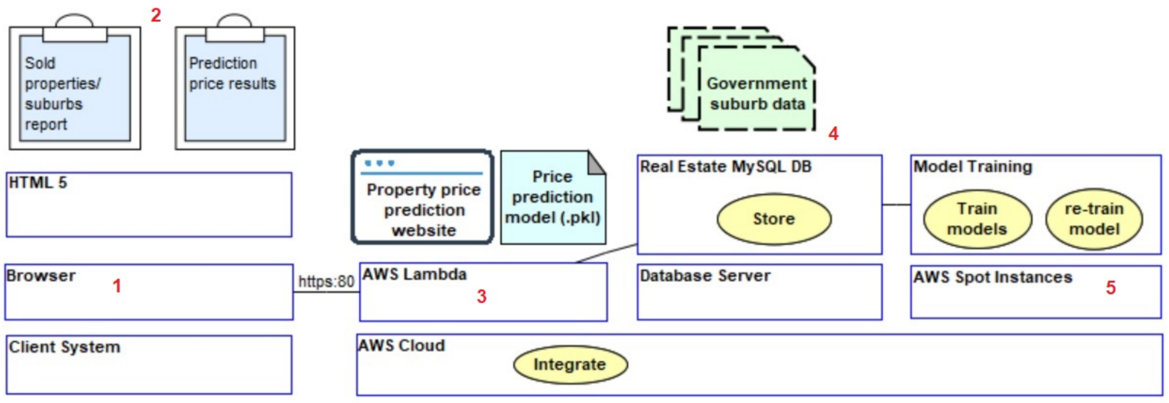
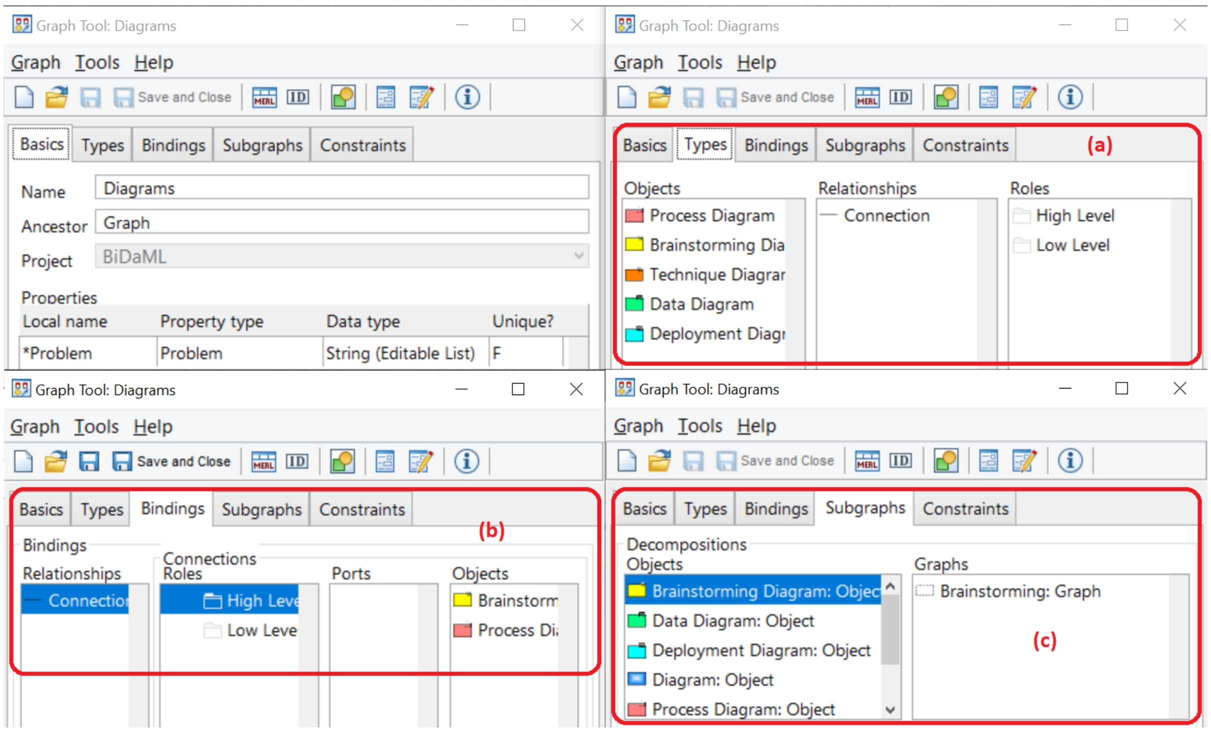


Figure 14. A deployment diagram example for the property price problem utilizing BiDaML deployment notation to show layers of the technology stack and how they support key tasks.

3.2 *BiDaML Support Tool*

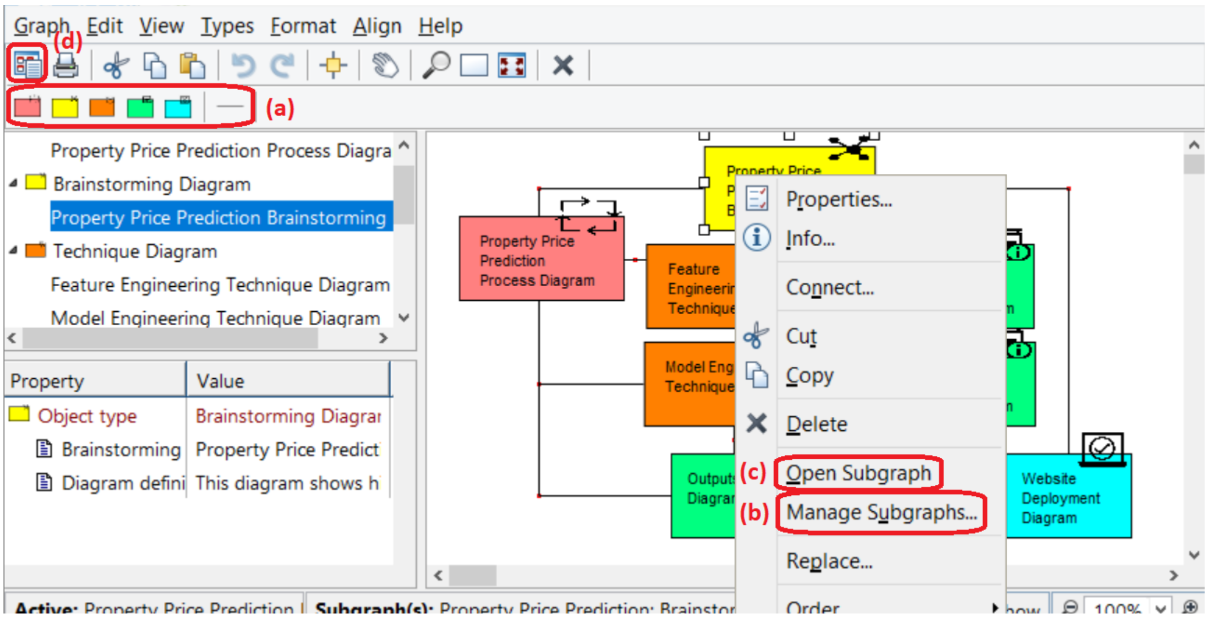
We have developed an integrated design environment for creating BiDaML diagrams. BiDaML tool support aims to provide a platform for efficiently producing BiDaML visual models and to facilitate their creation, display, editing, storage, code generation and integration with other tools. We used MetaEdit+ Workbench [17] to implement our tool. Using MetaEdit+, we created the objects and relationships defined as the notational elements for all of the diagrams, different rules on how to connect the objects using the relationships, and finally how to define low-level sub-graphs for the high-level diagrams. Figure 15 shows (a) how the objects, relationships, and roles are defined, (b) how the rules for connecting objects through relationships are defined, and (c) how the sub-graphs are connected to different objects of a graph, for the overview diagram. These are defined in a similar way for all other BiDaML diagrams.



Local name	Property type	Data type	Unique?
*Problem	Problem	String (Editable List)	F

Figure 15. Defining notational elements in MetaEdit+

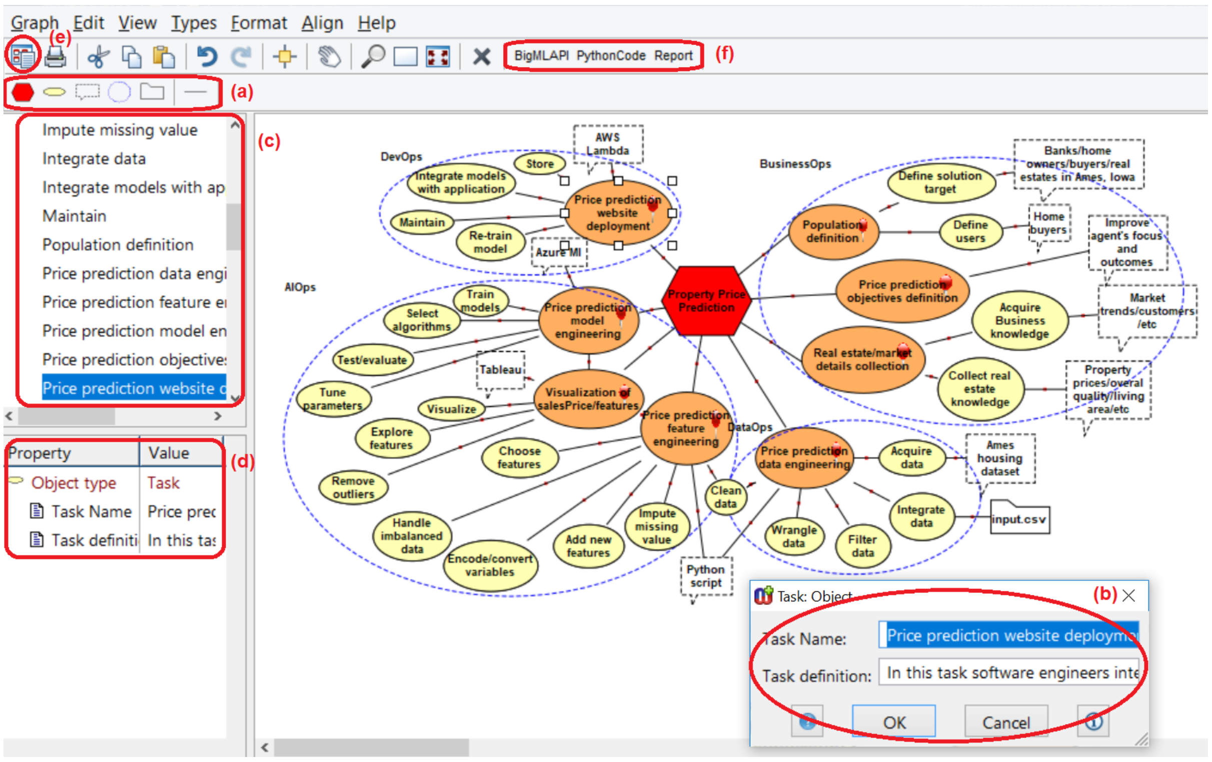
Once the notational elements are defined and bound and different diagrams are connected together, users can create different diagrams. Figure 16 shows our tool used to create an overview of all the diagrams for the price prediction example we discussed throughout this paper. Starting from an overview of all the diagrams, figure 16 shows that (a) notations can be dragged and dropped to the drawing area and connected together, then (b) different diagrams can be created and connected to each of the notations, and (c) accessed later. Once all the diagrams are created and connected to the appropriate notations, users can (d) generate different types of documentation and reports from the overview diagrams.


Graph Edit View Types Format Align Help
(d)
(a)
Property Price Prediction Process Diagra
Brainstorming Diagram
Property Price Prediction Brainstorming
Technique Diagram
Feature Engineering Technique Diagram
Model Engineering Technique Diagram
Property Price
Property Price Prediction Process Diagram
Feature Engineerin Technique
Model Eng Technique
Output Diagrar
Website Deployment Diagram
Properties...
Info...
Connect...
Cut
Copy
Delete
(c) Open Subgraph
(b) Manage Subgraphs...
Replace...
Order
Active: Property Price Prediction
Subgraph(s): Property Price Prediction: Brainstor
100%

Property	Value
Object type	Brainstorming Diagrar
Brainstorming	Property Price Predict
Diagram defini	This diagram shows h

Figure 16. How to create an overview of BiDaML diagrams, connect them together and generate reports

Figure 17 shows the brainstorming created for the same problem, as a sub-graph of the brainstorming diagram notation in the overview diagram. Here, users (a) choose the notations of objects/relationships and (b) modify the properties of the object/relationship. Notations added to the diagram are all listed (c) and details are shown by clicking on the notations (d). Users can click on any of the objects and create a sub-graph i.e., data and technique diagrams for them. Finally, once completed, (e) code generation features can be embedded and modified and (f) finally Python code, BigML API recommendations and reports can be generated for our property price prediction example in this designed brainstorming diagram. We will explain these in more detail in sections 3.2.1 and 3.2.2 below.


Graph Edit View Types Format Align Help
(e)
BigMLAPI PythonCode Report
(f)
(a)
(c)
Impute missing value
Integrate data
Integrate models with ap
Maintain
Population definition
Price prediction data engi
Price prediction feature e
Price prediction model en
Price prediction objective
Price prediction website d
(d)
DevOps
AWS Lambda
Store
Integrate models with application
Price prediction website deployment
Maintain
Re-train model
Azure MI
BusinessOps
Banks/home owners/buyers/real estates in Ames, Iowa
Define solution target
Population definition
Define users
Home buyers
Improve agent's focus and outcomes
Property Price Prediction
Price prediction objectives definition
Acquire Business knowledge
Market trends/customers/etc
Real estate/market details collection
Collect real estate knowledge
Property prices/overal quality/living area/etc
AIOps
Train models
Select algorithms
Price prediction model engineering
Test/evaluate
Tableau
Tune parameters
Visualization of salesPrice/features
Visualize
Explore features
Price prediction feature engineering
DataOps
Price prediction data engineering
Acquire data
Ames housing dataset
Choose features
Remove outliers
Clean data
Handle imbalanced data
Impute missing value
Add new features
Encode/convert variables
Wrangle data
Filter data
Integrate data
input.csv
Python script
(b)
OK
Cancel

Property	Value
Object type	Task
Task Name	Price prec
Task definiti	In this tas

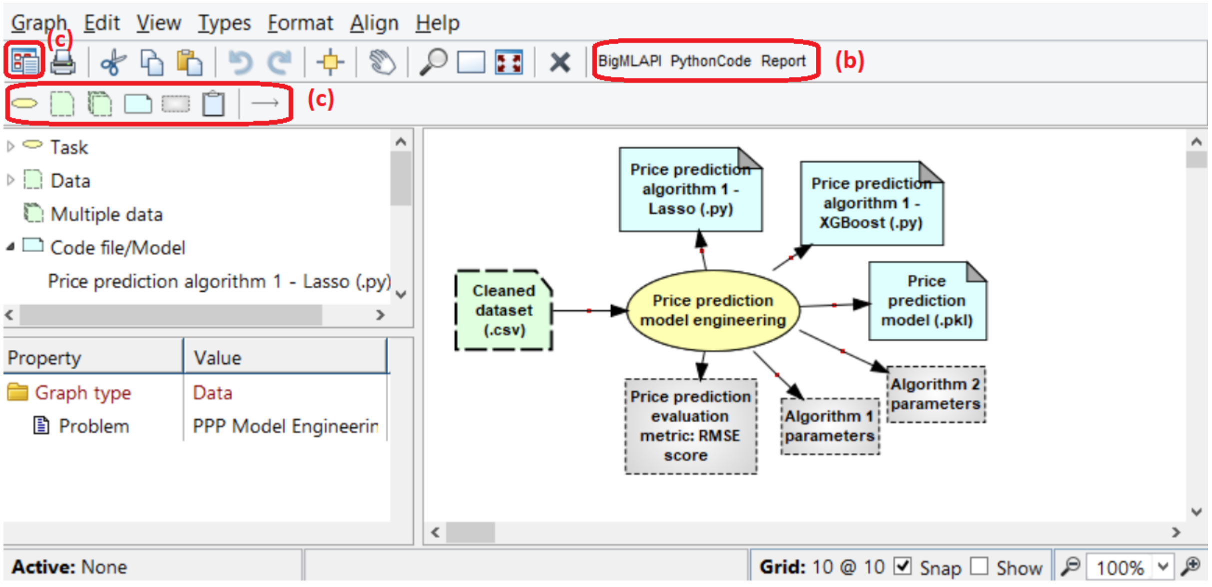
Task: Object

Task Name: Price prediction website deployme

Task definition: In this task software engineers inte

Figure 17. An example of BiDaML tool for creating brainstorming diagram for the property price problem

Figure 18 shows one of the technique diagrams created for the “price prediction model engineering” task in the brainstorming diagram, as a sub-graph of the technique diagram notation in the overview diagram. In this figure, users (a) choose the notations of objects/relationships and create the diagram by dragging, dropping and choosing a name/explanation for the objects. Technique diagrams also have (b) code generation features that can be embedded and modified and (f) reports can also be generated for the technique diagram.


Graph Edit View Types Format Align Help
(c)
BigMLAPI PythonCode Report
(b)
(c)
Task
Data
Multiple data
Code file/Model
Price prediction algorithm 1 - Lasso (.py)
Price prediction algorithm 1 - Lasso (.py)
Price prediction algorithm 1 - XGBoost (.py)
Cleaned dataset (.csv)
Price prediction model engineering
Price prediction model (.pkl)
Price prediction evaluation metric: RMSE score
Algorithm 1 parameters
Algorithm 2 parameters
Grid: 10 @ 10 Snap Show 100%

Property	Value
Graph type	Data
Problem	PPP Model Engineerir

Active: None

Figure 18. An example of BiDaML tool for creating technique diagram for the property price problem

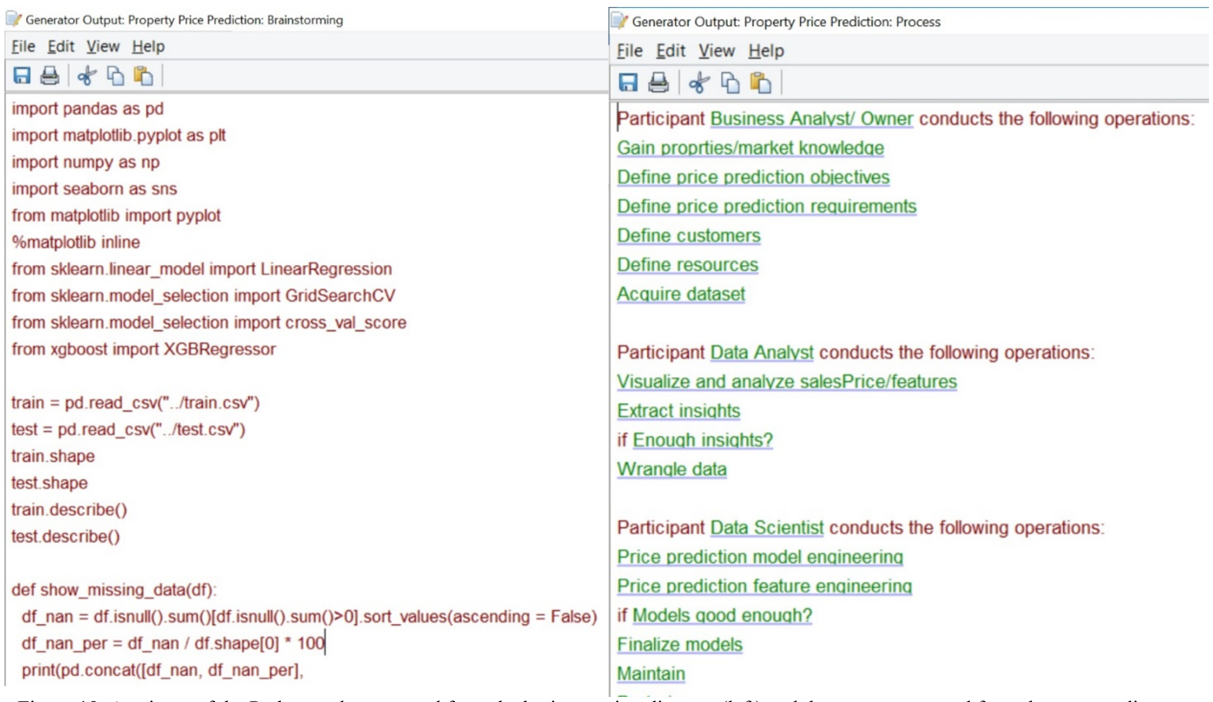
Once all the diagrams are created and connected, users can obtain outputs and share them with other stakeholders. There are currently two sets of outputs generated from the diagrams. First, a hierarchy of the graphs can be exported to Word and HTML from any of the diagrams. However, since all the sub-graphs are connected together in the overview diagram, the most comprehensive report can be generated and exported to Word/HTML through the overview diagram. The second set of outputs are Python code/BigML API, and reports that are embedded in the tool and can be traced back.

3.2.1 *BiDaML Report Generation*

Users can generate a hierarchical report from the overview diagrams, which covers all the high level to low level diagrams from brainstorming to deployment diagram, and also includes any optional metadata associated with objects in the diagrams. For example, task objects have an optional "task definition" field that can be included in the Word document without being presented in the diagrams. This will help to automatically aggregate all the diagrams and information, generate detailed reports and share them with the other stakeholders involved in the project. Some of the reports generated from different industrial projects can be found in [18].

3.2.2 *BiDaML Code Generation*

For the code generation part, we have currently embedded the common data analytics tasks including "importing libraries", "read/write files", "impute missing value", "visualize", "wrangle data", etc and well-known machine learning algorithms such as "XGBoost", "SVM", and "linear regression". The tool iterates through the tasks and techniques and if any of the existing methods are planned for the project as a task or technique, it includes the code in the generated Python file. Then users can modify the source code and work on it instead of writing the code from scratch. Also, users can embed their final code in the tool to reuse for future projects or retraining the existing models. Python code can be generated from brainstorming and technique diagrams. If generated from the brainstorming diagram, the code is more comprehensive since it iterates through the techniques associated with any of the tasks. Regarding API recommendation, we have currently included BigML API recommendation which helps users create their source code, dataset, model, and predictions more easily in order to get started with BigML.io. Setting up the environment variable, i.e., BIGML_AUTH, storing username and API key, etc are all included in the BigML API source code. This can be extended to any other tools and users can add their own API accesses. Figure 19 shows a snippet of the Python code generated from the brainstorming diagram and the report generated from the process diagram. All the diagrams in a project and the revision histories are stored in a Git repository. Any of the stakeholders can access the tool from their computer, modify different diagrams, commit changes to Git, and then push them (e.g. to GitHub) for the other collaborators to access the latest version.


Generator Output: Property Price Prediction: Brainstorming
File Edit View Help
Generator Output: Property Price Prediction: Process
File Edit View Help

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from matplotlib import pyplot
%matplotlib inline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from xgboost import XGBRegressor

train = pd.read_csv("../train.csv")
test = pd.read_csv("../test.csv")
train.shape
test.shape
train.describe()
test.describe()

def show_missing_data(df):
  df_nan = df.isnull().sum()[df.isnull().sum()>0].sort_values(ascending = False)
  df_nan_per = df_nan / df.shape[0] * 100
  print(pd.concat([df_nan, df_nan_per],
```

Participant Business Analyst/ Owner conducts the following operations:
Gain proprties/market knowledge
Define price prediction objectives
Define price prediction requirements
Define customers
Define resources
Acquire dataset

Participant Data Analyst conducts the following operations:
Visualize and analyze salesPrice/features
Extract insights
if Enough insights?
Wrangle data

Participant Data Scientist conducts the following operations:
Price prediction model engineering
Price prediction feature engineering
if Models good enough?
Finalize models
Maintain

Figure 19. A snippet of the Python code generated from the brainstorming diagram (left) and the report generated from the process diagram (right)

4 EVALUATION

We have evaluated the usability and suitability of BiDaML in two ways. The first was an extensive Physics of Notations evaluation of the visual notation [19]. This allowed a rigorous symbol-by-symbol evaluation of the usability and effectiveness of the visual notation against established theoretical principles without having to involve a large-scale usability trial. The second was an empirical evaluation with a range of target users. To test whether the set of five BiDaML diagram types were sufficient to cover a wide range of real-world data analytic scenarios, we ran a group user study in which participants applied BiDaML to model a data analytics solution of their choosing based on their own experience. To test our hypothesis that BiDaML diagrams are suitable as a common form of communication, we provided each participant with a BiDaML diagram created by another participant and asked them to compare it to a non-BiDaML diagram used as a control. To understand how easy BiDaML diagrams are to learn and use for specific projects, we also conducted a cognitive walkthrough of the tool with individual target end-users including data scientists and software engineers. Finally, we summarize our findings in using BiDaML in three industrial use cases.

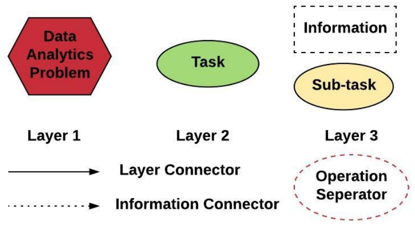
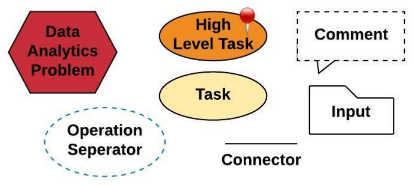
4.1 Physics of notations evaluation

Physics of Notations analysis involves a detailed symbol-by-symbol analysis against design principles. There are two aspects to evaluate: the semantics of the language, and the visual notations used to represent these semantics. Moody states that "the principles [of PoN] define desirable properties of visual notations, which can be used for evaluation and comparison" and that they have "been successfully used to evaluate and improve three leading SE notations" [20]. In our case, we apply PoN to both improve and evaluate the cognitive effectiveness of the BiDaML visual notations. PoN-based visual language evaluation and/or improvement has been used in many studies, such as [21-24]. For purposes of space, only the results of the analysis will be provided here.

Semiotic clarity specifies that a diagram should not have symbol redundancy, overload, excess and deficit. All visual symbols in BiDaML have 1:1 correspondence to their referred concepts. A partial exception to this principle in BiDaML is tasks and subtasks, which are conceptually identical but use different symbol variants in the brainstorming diagram to ensure that the high-level tasks are visually salient. ***Perceptual discriminability*** is primarily determined by the visual distance between symbols. All symbols in BiDaML use different shapes as their main visual variable, plus redundant coding such as color and/or textual annotation. As color is only used redundantly, BiDaML remains suitable for handwritten diagrams and users with color blindness. ***Semantic transparency*** identifies the extent to which the meaning of a symbol can be inferred from its appearance. In BiDaML, icons are used to represent visual symbols and minimize the use of abstract geometrical shapes. ***Complexity management*** restricts a diagram to have as few visual elements as possible to reduce its diagrammatic complexity. We used hierarchical views for representation in BiDaML. For example, the brainstorming diagram supports recursive decomposition of tasks into subtasks, with low-level details of subtasks presented using technique diagrams. As future work, we will add a feature for users to be able to hide visual details for complex diagrams. ***Cognitive integration*** identifies that the notations should support the user to assemble information from separate diagrams into a coherent mental representation of a system; and it should be as simple as possible to navigate between diagrams. All the diagrams in BiDaML have a hierarchical tree-based structure relationship as explained in Section 3. Furthermore, the five BiDaML diagram types share symbols with each other (e.g. the same symbol for tasks) in order to facilitate cognitive integration of the different diagrams.

Visual expressiveness defines a range of visual variables (position, shape, size, brightness, color, orientation texture) to be used, resulting in a perceptually enriched representation that exploits multiple visual communication channels and maximizes computational offloading. BiDaML diagrams utilize position, shape, size, brightness and color to distinguish symbols and convey meaning. Orientation was not utilized due to the rotational symmetry of shapes used by BiDaML. Texture was only partially utilized (e.g. different line styles) due to limited tool support for texture in MetaEdit+ Workbench and existing diagram editing tools. ***Dual coding*** means that textual encoding should also be used, as it is most effective when used in a supporting role. In BiDaML, all visual symbols have a textual annotation. ***Graphic economy*** discusses that the number of different visual symbols should be cognitively manageable. As few visual symbols as possible are used in BiDaML; the most complex type of BiDaML diagram is the process diagram which defines 10 symbols in contrast to the 171 possible symbols [25] in BPMN 2.0 process diagrams. ***Cognitive fit*** means that the diagram needs to have different visual dialects for different tasks or users. However, as BiDaML borrows concepts and symbols from common notations where possible, such as mindmaps (brainstorming), BPMN (process diagram), and dataflow diagrams (data diagram), the BiDaML notation is expected to be immediately usable and familiar to a broad range of users without the need to introduce dialects. Nevertheless, in the future, we plan to provide different views for different users in our BiDaML support tool, and users will be able to navigate between views based on their requirements.

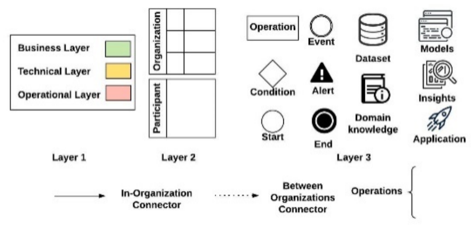
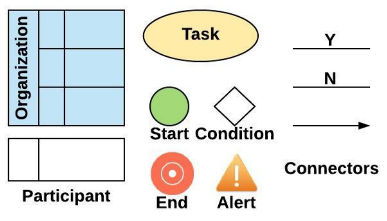
Table 1. BiDaML brainstorming diagram notations evolution: before and after several rounds of conducting physics of notations.

Diagram	**Initial Notations**	**Final Notations**
Brainstorming	 Data Analytics Problem Task Information Sub-task Layer 1 Layer 2 Layer 3 Layer Connector Information Connector Operation Seperator	 Data Analytics Problem High Level Task Comment Task Input Operation Seperator Connector

The initial and final notations designed for the brainstorming diagram after several rounds of conducting physics of notations are shown in table 1. The initial brainstorming notation was based on using two connectors, one to connect tasks and one to connect tasks to information. However, since it was not necessary to differentiate between different connectors, we consider the same connector type for both data and information in the updated version, to reduce the number of symbols, as recommended by physics of notations guidelines. Moreover, in the initial notation, the same shapes with different colors were used to differentiate high-level tasks from low-level tasks. This was potentially a problem for people with certain forms of color vision deficiency (color-blindness), and therefore, in order to avoid using many symbols and at the same time differentiate between high-level and low-level tasks, we automatically change the task symbol to a darker color with a pin icon if the task is directly connected to the problem symbol. High-level tasks are designed to resemble pinned todo-notes in order to improve semantic transparency. Tasks and subtasks both use an ellipse shape to show that they are conceptually similar.

The initial and final notations designed for the process diagram after several rounds of conducting physics of notations are shown in table 2. Our initial process diagram included 21 symbols and connectors, that contradicted PoN recommendation of around six symbols for novice users. We have now removed the unnecessary symbols and cut down the number of symbols and connectors to 10. The unnecessary items that are removed in the final version include different layers, operations, and icons to reflect datasets, models, domain knowledge, insights, and applications. Different layers correspond to the operations (BusinessOps, DataOps, AIOps, and DevOps) and since they are already specified in the brainstorming diagram, they were not adding any values to the process diagram. Moreover, the data items and artifacts are all specified in the data diagrams, and therefore using different icons to show how and where they are transmitted was only leading to complications in the process diagram. This greatly helped to reduce the number of notations and simplifying the process diagram. Given that the task icon is reused from the brainstorming diagram, and that the start, end, and condition icons are completely problem independent and shared between many diagrams, it is not a problem for novice users to learn these symbols. Moreover, the process diagram reuses symbols and syntax from BPMN where possible, so users who already have BPMN symbols and syntax stored in their long-term memory will only need to memorize the BiDaML specific aspects. The operation (task) symbol is also changed to an ellipse so that it matches and re-uses the task symbol in the brainstorming diagram. Furthermore, in the initial notation, the only difference between In-Organization and Out-Organization connectors was whether or not they span different organizations. However, since this can already be read off the process diagram by checking whether the connector spans different organization pools, in the final version, the notation is simplified by removing different types of organization connectors to avoid contradicting semiotic clarity.

Table 2. BiDaML process diagram notations evolution: before and after several rounds of conducting physics of notations.

Diagram	Initial Notations	Final Notations
Process	 Business Layer Technical Layer Operational Layer Organization Participant Layer 1 Layer 2 Operation Event Dataset Models Condition Alert Domain knowledge Insights Start End Application Layer 3 In-Organization Connector Between Organizations Connector Operations	 Organization Task Y N Start Condition Connectors Participant End Alert

The initial and final notations designed for technique diagram after several rounds of conducting physics of notations are shown in table 3. The technique diagram initially consisted of only four symbols that left some room for us to extend it further with additional concepts/symbols. e.g. which techniques worked, issues faced (overfitting, etc.), workarounds (possibly leading to further issues), etc. Therefore, we added two new symbols to reflect whether the technique was successful or needs further work.

Table 3. BiDaML technique diagram notations evolution: before and after several rounds of conducting physics of notations.

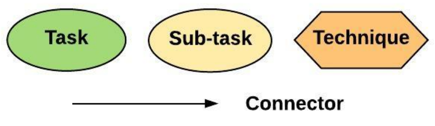
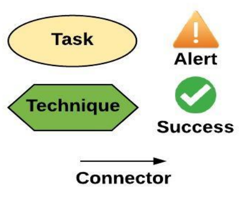
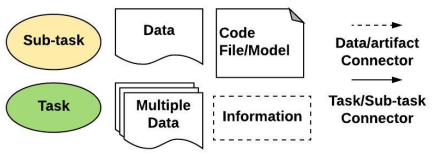
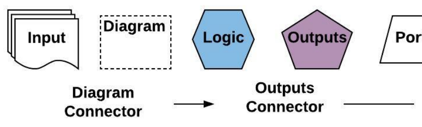
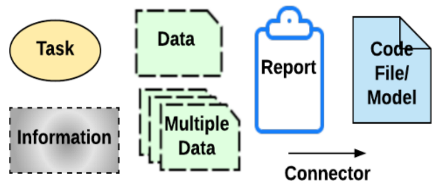
Diagram	Initial Notations	Final Notations
Technique		

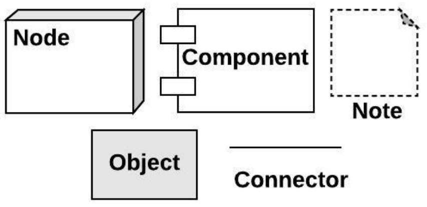
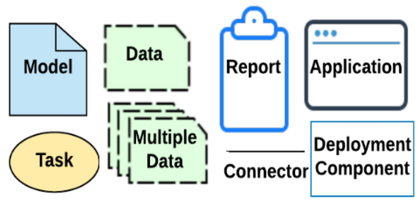
Table 4. BiDaML data diagram notations evolution: before and after several rounds of conducting physics of notations.

Diagram	Initial Notations	Final Notations
Data	Data  Sub-task Data Code File/Model Data/artifact Connector Task Multiple Data Information Task/Sub-task Connector Output  Input Diagram Logic Outputs Port Diagram Connector Outputs Connector	 Task Data Report Code File/ Model Information Multiple Data Connector

The initial and final notations designed for the data diagram after several rounds of conducting physics of notations are shown in table 4. BiDaML initially had two sets of diagrams, one to reflect the set of data items and artifacts used or generated throughout the process, and the other for showing the outputs and reports to be generated as results of the technique and data diagrams. We have now merged these two as one single data diagram and added a new notation for "report", to use for designing the expected reports and outputs. Also, the initial symbols were mostly rectangular and in the final version, we used a greater range of colors and shapes to distinguish between symbols. Moreover, data and task connectors are now considered as one connector. Furthermore, notations are defined in a more semantically transparent way, e.g. clipboards to represent reports.

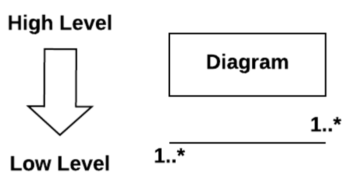
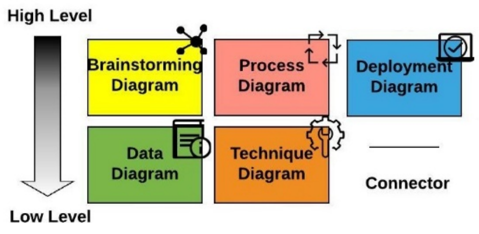
The initial and final notations designed for the deployment diagram after several rounds of conducting physics of notations are shown in table 5. The initial deployment diagram, borrowed from the UML deployment diagram, used shades of grey to represent different symbols; however, the PoN guidelines advocate for utilizing color (so long as it is not the only means to distinguish symbols). In the updated version, we used a system layer diagram to show layers of the technology stack on the vertical y-axis rather than using a UML deployment diagram (as big data requires thinking in terms of distributed cloud platforms, services, and frameworks rather than individual nodes/devices). The new deployment diagram reuses symbols from other BiDaML diagrams to improve perceptual integration with the other diagrams. BiDaML data diagram notations are reused to help more precisely distinguishing the type of object without the need to learn new symbols. The updated version also shows how the services and infrastructure support (DevOps) tasks over the lifespan of the project, to answer questions such as "what infrastructure will the Data Science team use to (re-)train their models?" and "where will this model be deployed?"

Table 5. BiDaML deployment diagram notations evolution: before and after several rounds of conducting physics of notations.

Diagram	Initial Notations	Final Notations
Deployment		 Model Data Report Application Task Multiple Data Connector Deployment Component

Finally, the initial and final notations designed for showing the overview of all the diagrams are shown in table 6. Moody describes a "summary (long shot) diagram" as an "important mechanism to support conceptual integration" of diagrams. BiDaML's initial overview diagram was at a meta-level (i.e. how diagrams connect and their cardinalities) rather than conveying the important problem specific information. Therefore, each diagram symbol was only capable of linking to a single diagram and was not sufficient to depict multiple diagrams of the same type. In the updated version, individual diagram instances are shown in the overview diagram to provide a "helicopter view" of the solution. Different colors and icons are used to represent the type of specific diagrams.

Table 6. BiDaML overview notations evolution: before and after several rounds of conducting physics of notations.

Diagram	Initial Notations	Final Notations
Overview		 High Level Low Level Brainstorming Diagram Process Diagram Deployment Diagram Data Diagram Technique Diagram Connector

4.2 *Cognitive walkthrough*

We conducted two sets of user studies, the first aiming at evaluating the BiDaML notations and comparing them with existing practices, with the second one evaluating the tool for specific projects.

4.2.1 *Group user study*

In this study, we asked a group of 12 data analysts, data scientists and software engineers to use any modeling language including UML, BPMN, data flow, entity relationship, mindmap, textual description, or their own ad hoc notation to model and describe a project of their choice on a piece of paper. We then introduced the BiDaML concept, notations, and diagrams and asked them to use one of the five BiDaML diagrams most related to the diagram they chose in the initial step to model the same problem on another piece of blank sheet. We then collected the handwritten diagrams and distributed them randomly between participants while making sure no one received their own handwritten diagram[1]. In this step, we asked participants to fill in a questionnaire and decide which diagram they prefer for a business owner, business analyst, a data scientist, a software engineer, or themselves. They were also asked to rate whether BiDaML is easy to understand in this step. Finally, a demo of the BiDaML tool was given to the audience and they were asked to rate whether they felt the tool was easy to learn.

The group study consisted of 9 PhD students, 2 academic staff, and 1 participant from industry. 8 participants categorized themselves as software engineers, 5 as data analysts/scientists, and 1 as "other" (2 of the participants described themselves as both software engineers and data analysts/scientists). The distribution of data analytics/data science experience was: 6 participants with less than 1 year; 3 participants with 2 years; 1 participant with 4 years; and 2 participants with 5 to 9 years. The distribution of programming experience was: 1 participant with 2 years, 1 participant with 3 years, 2 participants with 4 years, 6 participants with 5 to 9 years, and 2 participants with 10 or more years.

Surprisingly, none of the participants' initial diagrams used UML or BPMN. The majority of participants either made up their own ad-hoc notation, or combined multiple notations such as mindmaps, data flow diagrams, entity relationship, and textual descriptions.

Figure 20 shows participants' responses for which of the two diagrams (handwritten initial diagram or other participant's handwritten BiDaML diagram) they preferred. For the purposes of comparison, we excluded cases where participants responded "either", "neither" or left the question blank.

Of the 9 participants that stated a preference, all responded that they would prefer BiDaML over the alternative to communicate a data analytics solution to a business owner. Despite the small number of responses, we can statistically generalize that BiDaML is preferable (>50%) to alternatives for communicating data analytics solutions to business owners, as confirmed by a two-sided binomial test ($p<0.005$). A preference for BiDaML over alternatives was also weakly significant for communicating with software engineers ($p<0.05$). While the small number of responses prevents us from drawing conclusions about the use of BiDaML for the other audiences, in all cases at least 6 of the 12 participants preferred the BiDaML diagram over the alternative. A Wilcoxon signed-rank test confirms that participants prefer BiDaML for a greater number of audiences than the alternative ($p<0.005$), thus suggesting that BiDaML is suited for communication between different types of users in a data analytics project.

[1] As one participant returned from a break after diagrams were redistributed, they were allocated a pair of handwritten diagrams created prior to the study as a special case.

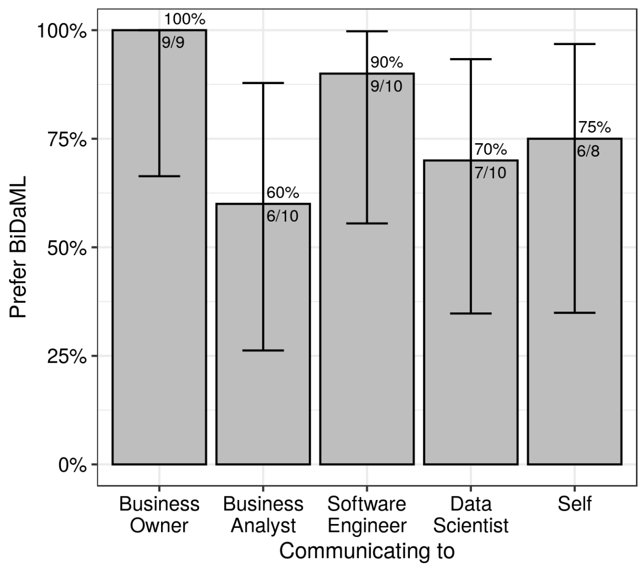


Figure 20. Percentage of participants that preferred handwritten BiDaML diagram over the handwritten alternative "to communicate a data analytics solution" for each audience. Error bars denote 95% confidence interval.

Figure 21 shows that 11 of 12 participants agreed that BiDaML was straightforward to understand. The one disagreement was likely unintentional, as the participant contradicted their response in a comment, stating that the diagrams "are easily understandable and communicable" and that BiDaML "includes a wide range of diagrams to cover almost all aspects of data and software engineering".

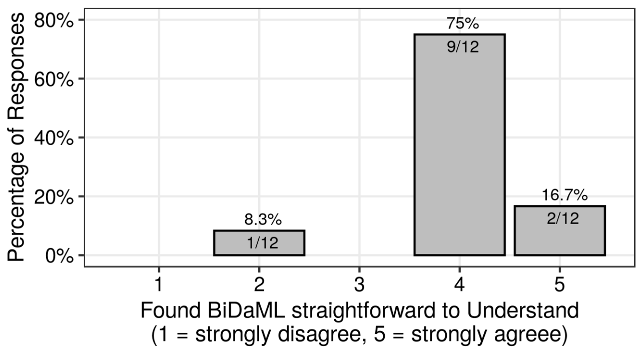


Figure 21. Distribution of participant agreement with the prompt "I found the BiDaML notation and diagrams straightforward to understand".

Figure 22 shows that 10 of 11 participants agreed that the BiDaML tool was straightforward to learn. The reason for lack of stronger agreement appears to be due to the unpolished nature of the current BiDaML user interface implemented within MetaEdit+, with explanations including "a bit hard to use the MetaEdit+ [user interface], but it is easy to know how to use" and that it was "unclear in what way automated codes are generated".

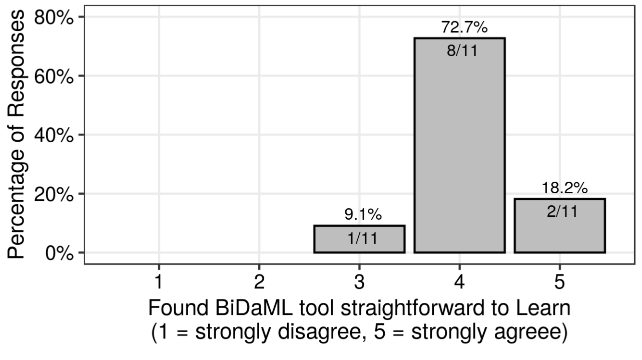


Figure 22. Distribution of participant agreement with the prompt "I found the BiDaML tool straightforward to learn".

4.2.2 Individual user studies

The individual study [10] consisted of 1 research fellow in data science, 3 academic staff, 1 in data science and 2 in software engineering, and 1 data scientist from industry. We asked the three data scientists and two software engineers (all experienced in big data analytics tasks) to carry out a task-based end-user evaluation of BiDaML. The objective was to assess how easy it is to learn to use BiDaML and how efficiently it can solve the diagram complexity problem. BiDaML diagrams were briefly introduced to the participants who were then asked to perform three predefined modeling tasks. The first was to design BusinessOps, DataOps, AIOps, or DevOps parts of a brainstorming diagram for a data analytics problem of their choice from scratch. In the second, the subject was given a process diagram and asked to explain it, comment on the information represented and provide suggestions to improve it. The third involved subjects designing a technique diagram related to a specific task of the data analytics problem they chose for the first part of the evaluation. Figure 23 shows (a) devOps part of one of the brainstorming diagrams and (b and c) two of the technique diagrams that the data scientists/software engineers drew to help explain their current work tasks of (a) automated test-case generation (b) data wrangling and (c) data augmentation.

Overall, user feedback indicated that BiDaML is very straightforward to use and understand. Users felt they could easily communicate with other team members and managers and present their ideas, techniques, expected outcomes and progress in a common language during the project and before the final solution. They liked how different layers and operations are differentiated. Moreover, they could capture and understand business requirements and expectations and make agreements on requirements, outcomes and results through the project. These could then be linked clearly to lower-level data, technique and output diagrams. Using this feedback we have made some minor changes to our diagrams such as the shape and order of some notations, and the relationships between different objects.

However, several limitations and potential improvements were also identified in our evaluations. Some users would prefer to see technique and data diagrams components together in a single diagram, while others would prefer to have these separate. In the process diagram, some users would prefer to only see the operations related to their tasks and directly related tasks. Finally, one of the users wanted to differentiate between tasks/operations that are done by humans versus by a tool. In future tool updates, we will provide different views for different users and will allow users to hide/unhide different components of the diagrams based on their preference. Moreover, in our future code generation plan, we will separate different tasks based on whether they are conducted by human or tool. We will then run a larger user evaluation with business and domain expert end-users.

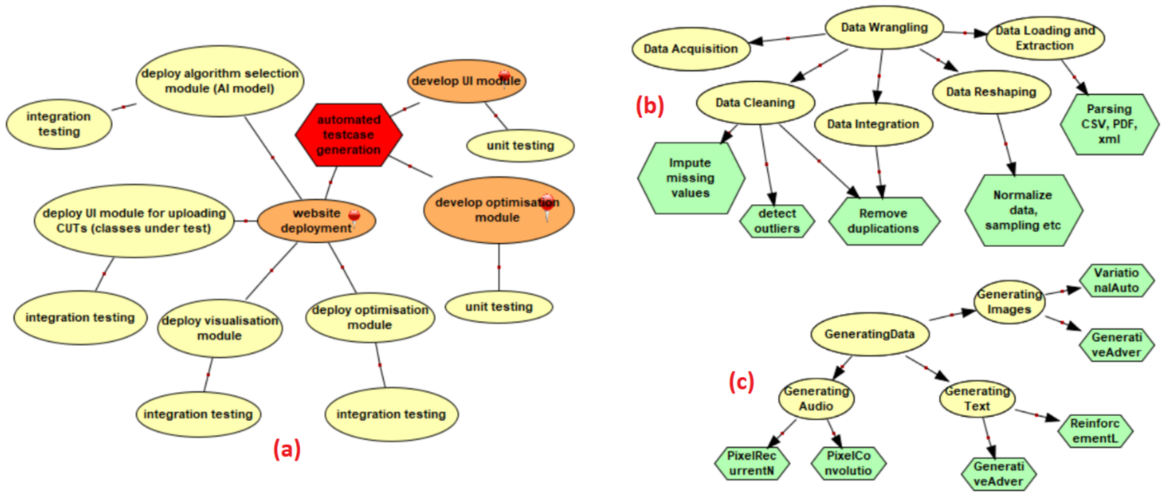


Figure 23. Example brainstorming and technique diagrams from our evaluators.

4.3 Industry Experiences with BiDaML

In this Section, we describe our experiences working with three different industry partners to model and capture the requirements of their big data analytics applications using BiDaML. A full list of the diagrams and the generated reports for the three projects are available in [18]. Based on the feedback from our industry users, as stated below, BiDaML can help understanding and communicating the project and its requirements within the teams, that can ultimately lead to reducing the time of the project's development and increasing the reusability of the projects through a common platform. However, since the code generation part was the secondary contribution of this paper, a comprehensive evaluation of the code generation part remains the focus of our future work.

4.3.1 ANZ REALas (realas.com)

REALas is a property price prediction website owned by the Australia and New Zealand Banking Group (ANZ). Launched in 2011, REALas claims to provide Australia's most accurate price predictions on properties listed for sale2. In this use-case, a complex new model needed to be developed to improve accuracy and coverage of the property price prediction model. The project team originally comprised a project leader, a business manager, a product owner, three software engineers, and one data scientist. There was an existing working website and ML model, as well as a dataset purchased from a third party. Two new data analyst/scientists were appointed to this project in order to create new models and integrate them with the existing website. The solution had initially been developed without the use of our BiDaML tool, and the challenges the team faced to communicate and collaborate through the process, was a key motivation for our research. The new data scientists initially lacked an understanding of the existing dataset and solution as well as domain knowledge. Therefore, it took them some time to be able to start the project. Moreover, communicating progress to the business manager and other members of the team was another problem. Together with the REALas team, we have used BiDaML to document the process from business analysis and domain knowledge collection through to the deployment of the final models in software applications.

We used BiDaML to redesign the whole project, in order to communicate and collaborate through the project, as well as automatically document all the above development steps. A brainstorming diagram was designed in the first place to help all the stakeholders fully understand and communicate the steps and existing solutions through a visualized drag-and-drop based platform. Once agreements were made on all the steps, a process diagram was created to specify the tasks to the existing stakeholders. These two steps only took a few hours from the whole team, and they could also incrementally modify these two during the process as needed. Data scientists were now fully aware of the domain knowledge, background of the project and existing models and could further leave comments and ask questions if further information was required. In the next step, data scientists worked on the data and ML parts, however this time, needed to visually record and keep track of the data items, artefacts, models, reports, etc that they tried, whether they were successful, failed, or were just being planned. This made it easy for them to communicate their progress and also reach agreement on the results. Our BiDaML tool code generation feature helped data scientists start from a template to work on instead of starting from scratch. The tool gives data scientists the ability to embed their code templates for future usage, as well as for documentation.

They gradually developed new models, and finally, worked with software engineers on a deployment diagram to define where and how to deploy the items generated in different steps. The new method was efficient in the way it took less time for the stakeholders to communicate and collaborate, and the step-by-step automatic documentation made the solution reusable for future reference. Based on the product owner's feedback *"this tool would have been helpful to understand and communicate the complexity of a new ML project within an organisation. It would assist the wider team to collaborate with data scientists and improve the outputs of the process"*.

4.3.2 Monash/VicRoads

VicRoads, the Victorian road traffic authority, utilizes the Sydney Coordinated Adaptive Traffic System (SCATS) to monitor, control and optimize traffic intersections. The Civil Engineering department at Monash University sought to build a traffic data platform that would ingest a real-time feed of SCATS data and integrate it with other transport datasets such as public transport travel history and traffic incidents reported through social media. Initially, the Civil Engineering department consulted with a software outsourcing company, who proposed a platform composed of industry standard big data tools. However, the software outsourcing company lacked understanding of the datasets and intended use of the platform, thus were unable to begin work on the project. Furthermore, it was unclear who would maintain the computing infrastructure, monitor data quality, and integrate new data sources after the initial phase of the project. We have worked with transport researchers and used BiDaML to document the intended software solution workflow from data ingestion through to traffic simulation and visualization. This allowed us to assist in formation of an alternative software solution that made better use of the systems and services already available.

We performed in-depth interviews with the project leader and traffic modeling expert, then used BiDaML to document the entire data analytics workflow including data ingestion, transport modeling, and result visualization. The traffic prediction brainstorming diagram was initially created as a handwritten sketch on paper, then later recreated using the BiDaML tool. The process diagram, technique diagram, data diagram, and deployment diagram were created directly using the BiDaML tool. While most BiDaML diagram types took only 15-30 minutes to create, the BiDaML process diagram proved the most time consuming, taking almost 3 hours due to the need to detail tasks to integrate each system and determine roles of individuals (we have since simplified the BiDaML process diagram notation in order to streamline the process). As BiDaML forces the user to consider all phases of the project, the modeling process helped reveal gaps in planning that required attention. Notably, no budget or personnel had been assigned to maintain the system after initial deployment, integrate new data sources, and monitor data quality/security. Indeed, in the BiDaML process diagram, we were forced to label both the organization and participant for these tasks as To Be Determined (TBD).

We presented the diagrams to the traffic modeling expert for feedback. This took place over a course of an hour session, in which we presented each diagram in the BiDaML tool. The BiDaML tool supported live corrections to the diagrams such as creation, modification or re-assignment of tasks as we discussed the diagrams with the traffic

2 https://realas.com/realas-science

modeling expert. Feedback from the expert was positive: "*I think you have a good understanding of the business... how do you know about all of this? I think this is very interesting, very impressive what you are proposing. It covers a lot of work that needs to be done.*" While the expert stated that the BiDaML diagrams were helpful to "*figure out all the processes and what tasks need to be done*" they were reluctant to use BiDaML to communicate with external stakeholders in other organizations: "*to use this tool, it will be likely not possible, because they [the other organizations involved] have their own process, they don't want to follow a new one.*" We subsequently presented printouts of the diagrams to the project leader. The project leader expressed some uncertainty about the purpose of the notation; however, noted that an adaptation of the BiDaML data diagrams as a means to document data provenance (i.e. the ability to trace the origins of data analysis results back to the raw data used) would be "*very useful*".

4.3.3 The Alfred Hospital (alfredhealth.org.au)

In this project, a group of radiologists, researchers and executives from the Alfred Hospital planned to use AI for predicting Intracranial hemorrhage (ICH) through CT Scans, work traditionally done by radiologists. The AI platform would enable them to prioritize the CT Scans based on the results and forward them to the radiologist for an urgent double check and follow up. Hence, a CT Scan with positive outcome could be reported in a few minutes instead of a few days. The team wanted to analyze the data before and after using the AI platform and based on the turnaround time (TAT) and cost analysis decide whether to continue using the AI platform or not. However, due to the diversity of the team it was difficult to communicate the medical terms to the data analysts and software engineers, and the analysis methods and software requirements and solution choices to the radiologists and the executive team. Thus, we used BiDaML with clinical, data science and software team members to model and document the steps and further plan for the next stages of the project.

We briefly introduced BiDaML to the team and since it seemed to be a well-designed fit for the project due to the diverse nature of the stakeholders, we decided to use the tool to model and analyze the requirements and capture the details. We had an initial one-hour meeting with one of the radiologists and started developing the models and collecting a deep understanding of the project, requirements, concepts, and objectives through the brainstorming diagram. Then we spent almost 30 minutes to document the entire data analytics workflow including data collection and wrangling, comparing the methods, making the final decision and deploying the final product through BiDaML's process, technique, data and deployment diagrams. Since we needed to deeply think about all the details and plans, BiDaML forced us to consider all phases and details of the project. We then organized a follow-up meeting with the team from the Alfred hospital, including two radiologists and the team leader and presented the diagrams for their feedback. The meeting took 30 minutes. During the meeting we modified the organizations and users involved as well as the expected reports and outcomes and the infrastructure in the deployment diagram. Going through the diagrams made us think about these and plan for them. We then shared the automatic report generated from the BiDaML tool with the team for their feedback. Feedback from the team was that *"BiDaML offered a simplified visual on different components of the project. These diagrams could be circulated to the project team and would clarify the workflow, requirements, aims and endpoints of each role and the entire project. In large-scale projects, BiDaML would be of even greater benefit, with involvement of multiple teams all working towards a common goal."* However, *"The user interface seemed quite challenging to navigate. However, this could be easily negated with appropriate training and instructional material."*

4.4 Threats to Validity

Due to the small number of participants involved in the group user study, only strong effects where nearly all participants agreed could be confirmed with any statistical significance. The group study questionnaire was delivered through an anonymous online form in order to encourage truthful feedback; however, it is possible that participants may have been biased to favor BiDaML diagrams due to familiarity with the researcher conducting the study. A further limitation is that time-bound user studies are unable to assess BiDaML at scale as a tool for communication within large projects. The authors have successfully utilized BiDaML to model mid-sized industrial projects such as property price prediction, transport data management, and CT Brain project communication. However, future work is needed to trial BiDaML in large long-term industrial projects. Moreover, the participants involved in the group user study were primarily PhD students and academics, so their attitudes to BiDaML may not reflect those of practitioners working in industry. This was the same for most of the participants (4 out of 5) in the individual study. However, it is not uncommon for industry to seek out PhD students and academic researchers to work on industry data science problems. Furthermore, the rise of powerful open source data analytics tools (e.g. TensorFlow) means that data scientists in academia will share much of the same software as data scientists in industry.

Another limitation of the group user study is that participants created their BiDaML diagram after their initial non-BiDaML diagram, so it is possible that the process of creating the initial diagram contributed to a better second diagram rather than the BiDaML notation itself. To avoid this, we could employ two separate evaluator groups to lessen this effect, one group to use only other modeling tools while the second group utilize BiDaML and to compare the achieved models. The challenge here would have been if we ran separate groups for each language, it would be difficult to control for the complexity of the diagrams during the comparison part of the study. By running a single session where participants design both a BiDaML diagram and a diagram using another modeling language, it ensures that both diagrams have the same complexity. Ideally, we would counterbalance the design through switching the order that the participants create their diagrams. However, the study places an extensive time burden on participants, with many commenting that the study was too long. Due to the burden the study places on participants, we chose not to repeat the study (future studies could trial a reversed/randomized ordering). Furthermore, many participants have had extensive prior formal training in formal languages (such as UML, etc.), but not BiDaML, so even if the ordering was reversed, it would not be a fair comparison. Moreover, participants selected problems that they already had familiarity with, for example, one participant based their initial diagram on a figure they had previously crafted for a research paper. One might argue that the complexity of models differs according to the selected projects, and therefore, how to manage this variability in the interpretation of the models and getting the results? We indeed compared BiDaML/Non-BiDaML models created by each of the participants for the same project by randomly distributing the BiDaML/Non-BiDaML models to the other participants. We believe that including a variety of projects has helped to compare BiDaML/Non-BiDaML for the same projects in different levels of complexity. Moreover, since the participants were from different fields and domains, by providing the participants with the freedom to choose their own project, we could make sure that the participants would only focus on the modeling part. If we were to choose the project for the participants to work on, the results could be biased toward our chosen project depending on its complexity and participants' familiarity.

In contrast to the group study, the individual user studies were conducted in a less structured manner, and as feedback was collected directly, it was not anonymous. For the individual user studies, the researcher sat with participants for 30 minutes to 1 hour to introduce the tool and create the diagrams. This enabled valuable feedback on the BiDaML notations themselves and the overall tooling approach to be gathered. However, due to the researcher's intervention, the usability of the current MetaEdit+ interface for new users of the BiDaML tool remains uncertain.

5 Discussion

Our experiences gained from discussions with the teams involved in data analytics projects, domain specific visual languages development, the PoN evaluation, and user studies indicate the lack of a prior modeling tool and collaboration framework in data analytics application development. The fact that none of our participants initially used UML or BPMN diagrams, as shown in Figure 1, as their way of communicating their data analytics projects with the other participants, as well as struggling to communicate their ideas indicates the lack of a common language or framework to collaborate through interdisciplinary teams. One of the initial questions we received from some of the participants was how to model, communicate and explain their problems to the other participants given they routinely work with a group of people in the same field, which is far from the interdisciplinary data analytics software development teams had in mind. Interestingly, based on our comparison in the group study, none of the participants, even though they were mostly software engineers and data scientists, used UML or BPMN as their initial diagrams, despite their familiarity with these modeling languages. The majority of participants finally made up their own ad-hoc notation or used a combination of multiple notations i.e., mindmaps, data flow diagrams, entity relationship, and textual descriptions. The decision of participants to use ad-hoc notations or a combination of notations, despite their formal training in software engineering and data science, further suggests the lack of existing standardized notations that support modeling of real-world data analytics problems. Also, our PoN evaluations and user studies, have provided evidence for the efficacy and usability of our BiDaML visual notations and languages as a standard way of communicating and modeling data analytics projects. Although we are expecting a wider gap in communications between software engineers, data scientists, and business managers, a recent online survey of business analysts and software architects by Linden et al. [26] suggested that UML, BPMN, SysML, and ArchiMate are the main notations used by practitioners for conceptual modeling. As our study primarily surveyed software engineers and data scientists, future work is needed to determine how business analysts respond to BiDaML. Conducting similar studies on business managers and analysts remains future work.

Limitations and deficiencies predominantly related to the maturity of the tool and its implementation. The current prototype tool has been developed using the bespoke MetaEdit+ domain-specific modeling development tool, which has limitations in fulfilling our desired end state. Some of the notable challenges we faced during evaluations and user studies were that although BiDaML can be accessed by all the stakeholders in different geographical locations, our intervention has been required so far, since for end users to use the tool, they would require a MetaEdit+ trial license. It also lacks a web-based user interface that would allow users to more easily access the tool without installing the software. Moreover, using MetaEdit+ limited us in generating documentation, reports and source code. It also limited us in easily giving access to users and evaluating the tool. We are currently looking to reimplement BiDaML as a stand-alone web-based tool and equip it with a recommender system to recommend suitable resources, models, techniques and solutions to the users based on the modeled problem and objectives. Another challenge we faced was that while BiDaML is ready for the initial problem definition and requirement analysis part, users continue to use existing tools or programming language to develop the ML and application development parts once they have completed the requirement analysis, modeling and planning part of the project. We aim to further develop BiDaML integrations for well-known existing tools to encourage users to continue using BiDaML through the entire development of the final product.

6 Related Work

There are many data analytics tools now available, such as Azure ML Studio [5], Amazon AWS ML [6], Google Cloud ML [7], and BigMl [8] as reviewed in [9, 27]. However, these tools only cover a restricted set of phases of DataOps, AIOps, and DevOps and none cover business problem description, requirements analysis and design. Moreover, most end-users in multidisciplinary teams have limited technical knowledge of data science and programming and thus usually struggle to use these tools.

Some DSVLs have been developed for supporting enterprise service modeling and generation using end-user friendly metaphors. An integrated visual notation (EML) for business process modeling is presented and developed in [28] using a novel tree-based overlay structure that effectively mitigates complexity problems. MaramaAIC [29] provides end-to-end support between requirements engineers and their clients for the validation and improvement of the inconsistencies in requirements. SDLTool [30] provides statistician end-users with a visual language environment for complex statistical survey design and implementation. These tools provide environments supporting end-users in different domains. However, they do not support data analytics processes, techniques, data, and requirements specifications, and do not target end-users for such applications.

Scientific workflows are widely recognized as useful models to describe, manage, and share complex scientific analyses and tools have been designed and developed for designing, reusing, and sharing such workflows. Kepler [31] and Taverna [32] are Java-based open-source software systems for designing, executing, reusing, evolving, archiving, and sharing scientific workflows to help scientists, analysts, and computer programmers. VisTrails [33] is a Python/Qt-based open-source scientific workflow and provenance management system supporting simulation, data exploration, and visualization. It can be combined with existing systems and libraries as well as user developed packages/modules. Finally, Workspace [34], built on the Qt toolkit, is a powerful, cross-platform scientific workflow framework enabling collaboration and software reuse and streamlining delivery of software for commercial and research purposes. Users can easily create, collaborate and reproduce scientific workflows, develop custom user interfaces for different customers, write their own specialized plug-ins, and scale their computation using Workspace's remote/parallel task scheduling engine.

Different projects can be built on top of these drag-and-drop based graphical tools and these tools are used in a variety of applications and domains. However, they are specifically designed for data analytics applications, and also make it hard for end-users to use data analytics and ML capabilities and libraries. Toreador [35] is a project aiming to overcome hurdles preventing companies from reaping the full benefits of big data analytics by delivering an architectural framework and components for model-driven set-up and management of big data analytics processes. Toreador is a "big data as a service" tool that helps to set up the pipeline and later execute it with other tools. However, what makes Toreador different from BiDaML is that Toreador users are software engineers lacking big data expertise who use Toreador as a replacement for NoSQL databases such as Cassandra or HBase, data preparation utilities such as Paxata, and distributed, parallel computing systems such as Apache Hadoop, Stark or Flink. Toreador needs a data set to be uploaded and a list of the processes to be chosen to be able to generate a workflow; that is where BiDaML can help end-users to make decisions and agreements, allowing them to be clear on the datasets to use or the methods to choose.

We aimed to provide a suite of visual languages to cover all aspects of data analytics projects with integration between different diagram concepts, instead of the user using different diagrams with different notations for various aspects. Having a suite of visual notations would help reusing and sharing the notations between diagrams as well as the participants. Most existing modeling notations such as UML, BPMN, etc provide a general purpose modeling framework and none of the modeling tools provide a high level to low level specific and goal driven means for the users to be able to easily memorize and model all the steps including brainstorming, technique, deployment, etc. However, any of the other modeling languages, i.e., iStar 2.0 [36] can be adapted and merged with BiDaML in case they cover an aspect of the data analytics application development that has not been covered in BiDaML. The main novelty of this paper is to present modeling languages to cover, model, and document all different aspects of data analytics applications in order to generate, recommend and reuse solutions. We aim to incorporate and synthesize the best existing practices instead of proposing completely new visual languages that do not follow any principles. Therefore, this does not preclude using iStar or integrating iStar or other goal-directed requirements modeling notations into BiDaML in the future.

Finally, some software tools implement algorithms specific to a given graphical model such as Infer.NET [37]. This approach for implementing data analytics techniques is called a model-based approach to ML [38]. An initial conceptualization of a domain-specific modeling language supporting code generation from visual representations of probabilistic models for big data analytics is presented in [39] by extending the analysis of Infer.NET. However, it is in very early stages and does not cover many of the data analytics steps of real-world problems.

7 Conclusions

We have described a set of visual notations for specifying data analytics projects. Our set of DSVLs, BiDaML, aims to provide a similar modeling framework for data analytics solution design as UML does for software design. It is comprised of five high- to low-level diagrammatic types. These diagrams represent both data- and technique-oriented components of a data analytics solution design. A Physics of Notations analysis and a cognitive walkthrough with several end-users were undertaken to evaluate the usability of BiDaML. We have also used our diagrams to model several complex big data analytics problems. Key findings include 1) nearly all participants agreed that the BiDaML notations/tool was straightforward to understand and learn, and 2) participants prefer BiDaML for supporting complex data analytics solution modeling than other existing modeling languages.

Our intended future work includes providing multiple view/elision support for large diagrams in our BiDaML modeling tool. In addition, we see considerable scope for providing back end integration with other data analytics tools, such as Azure ML Studio, Our tool can then be used at an abstract level during requirements analysis and design, and then connected to different tools at a low level from say Google, Microsoft or Amazon. Therefore, our DSVLs could be used to design, implement and control a data analytics solution. Our BiDaML tool will also support modeling and code generation, together with collaborative work support in the future. Since big data analysis has the same steps, the code generation feature of our tool will provide a set of templates for handling different classes of systems in data analytics projects. These will be leveraged to integrate our tool with other data analytics packages.

Acknowledgments

This work was supported by ARC Discovery grant DP170101932 and ARC Laureate Fellowship FL190100035.

References:

[1] I. Portugal, P. Alencar, and D. Cowan, "A Preliminary Survey on Domain-Specific Languages for Machine Learning in Big Data," presented at the IEEE International Conference on Software Science, Technology and Engineering (SWSTE), Beer-Sheva, Israel, 2016.

[2] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A Survey of Open Source Tools for Machine Learning with Big Data in the Hadoop Ecosystem," *Journal of Big Data,* vol. 2, no. 24, 2015.

[3] C. E. Sapp, "Preparing and Architecting for Machine Learning," Gartner Technical Professional Advice2017.

[4] J. B. Rollins, "Foundational Methodology for Data Science," IBM Analytics2015.

[5] *Microsoft Azure Machine Learning Studio*. Available: https://studio.azureml.net/

[6] *Machine Learning at AWS - Amazon AWS*. Available: https://aws.amazon.com/machine-learning/

[7] *Predictive Analytics - Cloud Machine Learning Engine | Google Cloud*. Available: https://cloud.google.com/products/machine-learning/

[8] *BigML.com is Machine Learning Made Easy*. Available: https://bigml.com/

[9] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "Survey and Analysis of Current End-user Data Analytics Tool Support," *IEEE Transactions on Big Data,* vol. 5, 2019.

[10] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "BiDaML: A Suite of Visual Languages for Supporting End-user Data Analytics," in *IEEE Big Data Congress*, Milan, Italy, 2019, pp. 93-97.

[11] W. v. d. Aalst and E. Damiani, "Processes Meet Big Data: Connecting Data Science with Process Science," *IEEE Transactions on Services Computing,* vol. 8, no. 6, pp. 810-819, 2015.

[12] D. D. Cock, "Ames, Lowa: Alternative to the Boston Housing Data as an End of Semester Regression Project," *Journal of Statistics Education,* vol. 19, no. 3, 2011.

[13] D. Sculley *et al.*, "Hidden Technical Debt in Machine Learning Systems," presented at the 28th International Conference on Neural Information Processing Systems (NIPS), Montreal, Canada, 2015.

[14] OMG. (2011). *Business Process Model And Notation (BPMN)*. Available: https://www.omg.org/spec/BPMN/2.0/

[15] S. Ambler, *The Object Primer: Agile Model-Driven Development With Uml 2.0 3rd Edition*. Cambridge University Press 2004.

[16] S. Kelly and R. Pohjonen, "Worst Practices for Domain-Specific Modeling," *IEEE Software,* vol. 26, no. 4, pp. 22-29, 2009.

[17] *MetaEdit+ Domain-Specific Modeling tools – MetaCase*. Available: https://www.metacase.com/products.html

[18] *BiDaML Case Studies*. Available: http://bidaml.visualmodel.org/

[19] D. Moody, "The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering," *IEEE Transactions on Software Engineering,* vol. 35, no. 6, pp. 756-779, 2009.

[20] D. Moody and J. van Hillegersberg, "Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams," in *Software Language Engineering*, Berlin, Heidelberg, 2009, pp. 16-34: Springer Berlin Heidelberg.

[21] M. Famelis and M. Chechik, "Managing design-time uncertainty," *Software & Systems Modeling,* vol. 18, no. 2, pp. 1249-1284, 2019/04/01 2019.

[22] H. Henriques, H. Lourenço, V. Amaral, and M. Goulão, "Improving the Developer Experience with a Low-Code Process Modelling Language," presented at the Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Copenhagen, Denmark, 2018. Available: https://doi.org/10.1145/3239372.3239387

[23] T. Miranda *et al.*, "Improving the Usability of a MAS DSML," in *Engineering Multi-Agent Systems*, Cham, 2019, pp. 55-75: Springer International Publishing.

[24] E. Gonçalves, C. Almendra, M. Goulão, J. Araújo, and J. Castro, "Using empirical studies to mitigate symbol overload in iStar extensions," *Software and Systems Modeling,* 2019/12/12 2019.

[25] N. Genon, P. Heymans, and D. Amyot, "Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation," in *Software Language Engineering*, Berlin, Heidelberg, 2011, pp. 377-396: Springer Berlin Heidelberg.

[26] D. van der Linden, I. Hadar, and A. Zamansky, "What practitioners really want: requirements for visual notations in conceptual modeling," *Software & Systems Modeling,* vol. 18, no. 3, pp. 1813-1831, 2019.

[27] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "A Survey of Current End-user Data Analytics Tool Support," in *IEEE International Congress on Big Data 2018*, San Francisco, USA, 2018, pp. 41-48.

[28] L. Li, J. Grundy, and J. Hosking, "A Visual Language and Environment for Enterprise System Modelling and Automation," *Journal of Visual Languages & Computing,* vol. 25, no. 4, pp. 253-277, 2014.

[29] M. Kamalrudin, J. Hosking, and J. Grundy, "MaramaAIC: Tool Support for Consistency Management and Validation of Requirements," *Automated Software Engineering,* vol. 24, no. 1, pp. 1-45, 2017.

[30] C. H. Kim, J. Grundy, and J. Hosking, "A Suite of Visual Languages for Model-Driven Development of Statistical Surveys and Services," *Journal of Visual Languages and Computing,* vol. 26, no. C, pp. 99-125, 2015.

[31] B. Ludäscher *et al.*, "Scientific Workflow Management and the Kepler System," *Concurrency and Computation: Practice and Experience,* vol. 18, no. 10, pp. 1039-1065, 2005.

[32] K. Wolstencroft *et al.*, "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud," *Nucleic acids research,* vol. 41, no. 1, pp. 557-561, 2013.

[33] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "VisTrails: Visualization Meets Data Management," in *ACM SIGMOD international conference on Management of data*, 2006, pp. 745-747.

[34] P. W. Cleary, D. Thomas, M. Bolger, L. Hetherton, C. Rucinski, and D. Watkins, "Using Workspace to Automate Workflow Processes for Modelling and Simulation in Engineering," presented at the 21st International Congress on Modelling and Simulation, 2015. Available: https://research.csiro.au/workspace/

[35] E. Damiani, C. Ardagna, P. Ceravolo, and N. Scarabottolo, "Toward Model-Based Big Data-as-a-Service: The TOREADOR Approach," in *Advances in Databases and Information Systems*, Cham, 2017, pp. 3-9: Springer International Publishing.

[36] F. Dalpiaz, X. Franch, and J. Horkoff. (2016). *iStar 2.0 Language Guide*. Available: https://sites.google.com/site/istarlanguage/home#h.p_ID_44

[37] T. Minka, J. Winn, J. Guiver, and D. Knowles, "Infer .NET 2.4, 2010. Microsoft Research Cambridge," 2010.

[38] C. M. Bishop, "Model-based Machine Learning," *Philosophical Transactions of the Royal Society A, Mathematical, Physical and Engineering Sciences,* vol. 371, no. 1984, 2012.

[39] D. Breuker, "Towards Model-Driven Engineering for Big Data Analytics – An Exploratory Analysis of Domain-Specific Languages for Machine Learning," presented at the 47th Hawaii International Conference on System Science, 2014.