Model-Driven Technologies for Data Mining Democratisation

Alfonso de la Vega and Pablo Sánchez

Software Engineering and Real-Time, University of Cantabria, Santander (Spain)
{alfonso.delavega, p.sanchez}@unican.es

Abstract. Data mining techniques allow discovering insights previously hidden in data from a domain. However, these techniques demand very specialised skills. People often lack these skills, which hinders *data mining democratisation*. To alleviate this situation, we defined a model-driven framework and some domain-specific languages that contribute to the democratisation of data mining. Here we summarise these contributions.

Keywords: Model-Driven Engineering · Domain-Specific Languages · Data Mining · Data Mining Democratisation

1 Introduction

Currently, computer systems gather large amounts of data that, when properly analysed, can be of great help for different purposes [9]. For instance, data collected by Uber is being used by different city halls to improve public transport networks, whereas Netflix is using their data to determine its next productions.

Nevertheless, data mining techniques, which can find valuable facts hidden in data, require very specialised skills. For instance, before grouping some data by their similarities, we must decide which one of the dozens of available clustering algorithms best fits with our needs. Then, some preprocessing is necessary to adapt the input data to the requirements of the selected algorithm, such as converting categorical values to a numerical representation; or normalising numbers into the range [0, 1]. People willing to analyse data often lack the technical skills to achieve these tasks, which hampers *data mining democratisation*.

As a first step to address these issues, we analysed the state of the art of the data mining democratisation field by means of a systematic literature review [4]. In this review, more than 700 works were considered, including both research articles and industrial tools. Some conclusions of this review are: (1) generic solutions, which are completely domain-independent, might exhibit accuracy problems, since they do not take into account the particularities of each domain to configure their algorithms or to preprocess input data; and (2) the issue of facilitating the data selection and data formatting stages is scarcely addressed in the literature.

Model-Driven Engineering (MDE) and *Domain-Specific Languages (DSLs)* have demonstrated to be effective methods to provide domain-adapted solutions that are easy to use and feel familiar to experts in an application domain. Therefore, we explored whether these benefits can be applied to the data mining area. Our initial idea was to create a DSL with a high-level syntax that hid low-level details of the applied mining techniques, so that it could be used for people without expertise on these. This DSL was initially devised to work with data coming from any domain, but ignoring domain details quickly turned into an unfeasible option, as the first conclusion of our review states. Thus, we opted to develop *FLANDM*: a model-driven framework for the rapid generation of DSLs for data mining [7], where generated DSLs are adapted to the specificities of each concrete context.

Additionally, this framework uses two DSLs, *Lavoisier* [6] and *Pinset* [8], to support its customisation. These DSLs address the second conclusion of our review by helping with the data transformation steps, i.e., making data conform to the requirements imposed by the applied data mining algorithms.

Our approach has been validated by generating DSLs for several domains, with a special focus in the analysis of data extracted from e-learning platforms, web systems, and data from model-driven artefacts [8]. Moreover, we performed a set of empirical experiments to state whether the generated DSLs might be actually used by people without knowledge on data mining techniques.

The rest of this paper is organised as follows: Section 2 introduces *FLANDM*, i.e., our framework for the generation of DSLs for data mining. Sections 3 and 4 describe *Lavoisier* and *Pinset*, which are our languages for the transformation of data into an analysis-ready format. Finally, Section 5 concludes this work.

2 FLANDM: A Model-Driven Framework for the Generation of DSLs for Data Mining Democratisation

To address the first issue stated in the introduction, some authors created frameworks for the development of data mining applications, where an expert initially configures some elements of the framework so that the resulting application is adapted to a specific domain. In these cases, it is important to reduce the intervention of experts as much as possible, in order to decrease development cost.

With this idea in mind, we created *FLANDM (Framework to develop LANguages for Data Mining)* [7]. FLANDM is an MDE-based framework that can be used to create DSLs for data mining democratisation. These DSLs hide technical details of the applied analysis techniques behind a high-level, query-based syntax, in order to be usable by people without expertise on data mining. Generated DSLs are adapted to the particularities of each domain, which makes them feel familiar to use, and contributes to improving the accuracy of the analyses.

Figure 1 provides a general overview of how FLANDM works. As it happens in any data mining process, we start with a set of *business questions* to be answered. For instance, a software engineer might want to know why some classes of a software system are more likely to contain bugs than others.

These questions are complemented with a characterisation of the analysis context by means of a *domain model*. The purpose of this domain model is

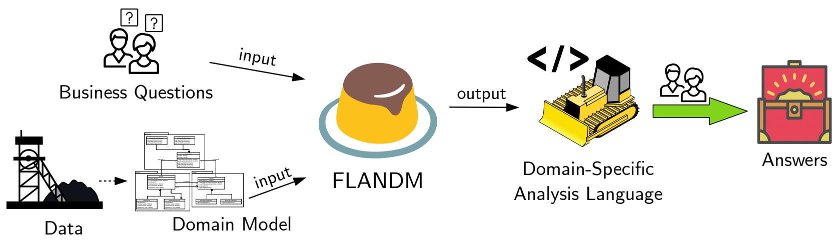


Fig. 1. FLANDM's language generation process (Icons: Flaticon, Noto Emoji font).

twofold: (1) to indicate the terminology with which domain experts are familiar; and (2) to specify the available *data* for the analysis. These data might be present in a well-defined source, such as a relational database; or it might need to be extracted from several sources. For instance, continuing with our previous example, we can use as data some quality metrics computed for each class of the software system. In addition, these data could be complemented with information extracted from a bug tracking tool. The steps of extracting and integrating data from different sources are not currently addressed by FLANDM, and need to be performed manually.

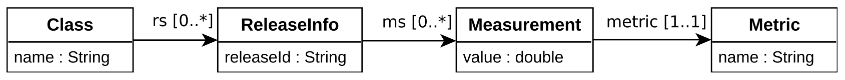
Listing 1.1. Query examples of an analysis language generated with FLANDM.

```
1  find_reasons for num_bugs > 10 of classes_bug_info;
2
3  find_reasons for num_bugs > 10 of classes_bug_info
4      with package not_equals "legacyAccountMng";
```

Using this information as input for FLANDM, we could generate a query-based language such as the one depicted in Listing 1.1. As it can be seen, the employed terms (*num_bugs*, *class*, *package*) should be familiar to software engineers. The structure of these sentences would be similar for all domains. Each query is composed of a command, that specifies the kind of answer to be computed; a dataset, which determines the data to be used for that analysis; and, optionally, filters that might exclude some data from the analysis. In Listing 1.1, line 1 we try to find reasons that lead to a number of bugs higher than a specific threshold using a dataset called *classes_bug_info*. In lines 3-4 we perform the same query, but in this case we omit those classes from package "legacyAccountMng" from the analysis.

These high-level sentences are translated, by means of model transformation and code generation techniques, into low-level code that configures and invokes specific data mining algorithms. This generated code is then executed to provide an answer to the specified query.

Both the DSL generation infrastructure and the sentence transformation process have been designed so that they can be easily configured by data mining experts to fit with the particularities of each domain. For instance, an expert can change easily the underlying algorithm that is used to compute a specific command, or fine tune some of its parameters.



name	rs_3.1		rs_3.2	
	ms_cbo	ms_dit	ms_cbo	ms_dit
Book	3	0	4	0
Novel	1	1	1	1

```
dataset releasesInfo {
  mainClass Class [name]
  include rs by releaseId {
    include ms by metric.name
  }
}
```

Fig. 2. Top: domain model for class-level metrics; bottom-left: target dataset; bottom-right: Lavoisier query that performs the extraction.

To evaluate the benefits of FLANDM, we carried out two different actions. First, we compared the effort of developing DSLs for data mining from scratch and with the help of FLANDM, for four different domains. Results showed that our framework helps reduce around 50% of development efforts. Secondly, we checked whether the generated languages can be actually used by people without expertise in data mining by carrying out some empirical experiments. University teachers from heterogeneous areas used an educational analysis language to study courses data from an e-learning platform. At the time of writing this paper, we are still processing the gathered data, but preliminary results indicate that teachers were able to correctly use this language after a minimum training.

As commented, each executed query indicates a *dataset* as input data. In our framework, a dataset is a tabular representation of a data bundle selected from the domain model. The need of being tabular is a requirement imposed by most data mining algorithms. Our framework provides two languages, called *Lavoisier* and *Pinset*, which allow non-experts to create datasets from a domain model by themselves, i.e., without the assistance of data mining experts. These languages are briefly described in next sections.

3 Lavoisier: High-Level Data Selection and Processing

Lavoisier [6] is a language for creating datasets from object-oriented domain models. Dataset creation, i.e., the process of transforming data into a two-dimensional format to serve as input of an analysis algorithm, is considered one of the key stages of any data mining process [5]. In our framework (Figure 1), once a domain model has been created and populated with accurate and clean data, datasets can be produced by specifying, through a Lavoisier query, a subset of this domain model to be considered for a specific analysis. Then, this subset must be transformed into an analysis-ready dataset to be digested by data mining tools.

The problem of data formatting is illustrated in Figure 2. The top of this figure shows a domain model about quality metrics of a software system. For each class contained in this system, several metrics per release are computed. Examples of these metrics could be *CBO (Coupling Between Objects)* or *DIT*

(Depth of Inheritance Tree). These and other metrics have been previously used, for instance, to predict the defects that will be found in a software release [3].

A domain model represents information in a graph-like format, whereas most analyses require data to be transformed into a tabular format like the one depicted in Figure 2 (bottom left). To perform this task, several data transformation operations, such as *joins* or *pivots*, are typically used.

Domain experts are key for the proper creation of datasets, since they might give some useful input to correctly guide an analysis. So, it would be desirable if these experts were able to define their own datasets. Nevertheless, domain experts often lack the technical skills to accomplish this task.

Lavoisier tries to alleviate this shortcoming. This language provides a high-level syntax that we expect can be used by domain experts, since it tries to hide any technical details of the dataset creation process. Therefore, a domain expert might focus on data selection, rather than on which combination of low-level operations has to be used to obtain data in the required format.

Figure 2 (bottom right) shows an example of dataset creation using Lavoisier. The dataset `releasesInfo` will be used to compare class metrics of different releases, so each row of this dataset would contain the information of a class. We indicate this in the query by selecting `Class` as the `mainClass` of the dataset. From each class object, we include its `name`. As information for the analysis, we include data from all the releases `rs` of each class. Each set of columns extracted from a release will be identified by its `releaseId`. Finally, for each release, we include all measurements, each one corresponding to a metric name (e.g. *cbo* or *dit*). Lavoisier automatically uses the *value* of a measurement to fill the corresponding columns. This specification, when executed, produces a dataset like the one shown in Figure 2 (bottom left). It should be noted that, in this case, the number of columns of the resulting dataset varies dynamically depending on the number of releases and gathered metrics.

The execution of a dataset specification is carried out by Lavoisier transparently, freeing domain experts of these low-level details. To perform this execution, Lavoisier employs a set of data transformation patterns [6] that we defined by adapting some typical procedures applied in object-relational data mappers and in data management tools.

4 Pinset: MDE that Helps Data Mining Help MDE

Following a current trend [1,2], we tried to employ Lavoisier to enable the use of data mining techniques on data extracted from MDE artefacts. During this evaluation, we realised that Lavoisier's high-level syntax might be inadequate for domain experts with programming skills, such as software engineers. We found that some fine-grain aspects of a dataset creation, like the computation of aggregate values, cannot be easily specified using *Lavoisier* constructs. Thus, we extended the initial objectives of *Lavoisier* to create a new DSL, called *Pinset* [8], which offers a lower-level syntax for performing some computations.

Listing 1.2. Dataset extraction with Pinset.

```
1 dataset classAggregates over c : Class {
2   properties [name as className]
3   column numDefectiveReleases : c.rs.select(r | r.ms.exists(m |
4       m.metric.name = "num_bugs" and m.value > 0)).size()
5   ...
6 }
```

Listing 1.2 shows a dataset extraction over the domain model of Figure 2. In this dataset, the entities to be analysed are again *Classes* (line 1). Several metrics are computed for each class. For space reasons, only the *numDefectiveReleases* metric is shown (lines 3-4), which indicates the number of releases per class where at least one defect was detected. This metric is calculated by chaining different operators that are interpreted to generate the resulting value.

5 Conclusions and Future Work

This paper has briefly described our MDE-based contributions in the field of data mining democratisation: the FLANDM framework [7], and the Lavoisier [6] and Pinset [8] languages. As future work, we plan to perform new empirical experiments of these contributions, also including new analysis domains. We also want to explore new research lines, such as how to define explainable (white-box) analysis processes for non-experts, or how to allow for a more fine-grained configuration of these processes with a controlled increase of the complexity.

Acknowledgements. Funded by the University of Cantabria's Doctorate Program, and by the Spanish Government under grant TIN2017-86520-C3-3-R.

References

1. Babur, Ö., Cleophas, L., van den Brand, M.: Hierarchical clustering of metamodels for comparative analysis and visualization. In: Modelling Foundations and Applications - 12th European Conference, ECMFA. pp. 3–18 (2016)
2. Basciani, F., Rocco, J.D., Ruscio, D.D., Iovino, L., Pierantonio, A.: Automated clustering of metamodel repositories. In: CAiSE. pp. 342–358 (2016)
3. D'Ambros, M., Lanza, M., Robbes, R.: An extensive comparison of bug prediction approaches. In: IEEE Int. Conf. Mining Software Repositories. pp. 31 – 41 (2010)
4. de la Vega, A., et al.: How Far are we from Data Mining Democratisation? A Systematic Review. arXiv e-prints 1903.08431 (2019), https://arxiv.org/abs/1903.08431
5. Munson, M.A.: A study on the importance of and time spent on different modeling steps. SIGKDD Explor. Newsl. **13**(2), 65–71 (May 2012)
6. de la Vega, A., García-Saiz, D., Zorrilla, M., Sánchez, P.: On the Automated Transformation of Domain Models into Tabular Datasets. ER FORUM **1979** (2017)
7. de la Vega, A., García-Saiz, D., Zorrilla, M., Sánchez, P.: FLANDM: a development framework of domain-specific languages for data mining democratisation. Computer Languages, Systems and Structures **54**, 316–336 (2018)
8. de la Vega, A., Sanchez, P., Kolovos, D.: Pinset: A DSL for Extracting Datasets from Models for Data Mining-Based Quality Analysis. Quality of Information and Communications Technology (QUATIC) pp. 83–91 (2018)
9. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: Practical Machine Learning Tools and Techniques. 4th edn. (2016)