IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 9, 2019, accepted January 9, 2020, date of publication January 14, 2020, date of current version January 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2966630

Classification Algorithms Framework (CAF) to Enable Intelligent Systems Using JetBrains MPS Domain-Specific Languages Environment

SOFIA MEACHAM[1], (Member, IEEE), VACLAV PECH[2], AND DETLEF NAUCK[3]
[1]Department of Computing and Informatics, Bournemouth University, Poole BH12 5BB, U.K.
[2]JetBrains MPS, 140 00 Prague, Czech Republic
[3]British Telecom, Adastral Park, U.K.

Corresponding author: Sofia Meacham (smeacham@bournemouth.ac.uk)

This work was financially supported by research funding allocated directly from British Telecom (Data Science team in BT Research Headquarters in Adastral Park) to Bournemouth University.

ABSTRACT This paper describes the design and development of a Classification Algorithms Framework (CAF) using the JetBrains MPS domain-specific languages (DSLs) development environment. It is increasingly recognized that the systems of the future will contain some form of adaptivity therefore making them intelligent systems as opposed to the static systems of the past. These intelligent systems can be extremely complex and difficult to maintain. Descriptions at higher-level of abstraction (system-level) have long been identified by industry and academia to reduce complexity. This research presents a Framework of Classification Algorithms at system-level that enables quick experimentation with several different algorithms from Naive Bayes to Logistic Regression. It has been developed as a tool to address the requirements of British Telecom's (BT's) data-science team. The tool has been presented at BT and JetBrains MPS and feedback has been collected and evaluated. Beyond the reduction in complexity through the system-level description, the most prominent advantage of this research is its potential applicability to many application contexts. It has been designed to be applicable for intelligent applications in several domains from business analytics, eLearning to eHealth, etc. Its wide applicability will contribute to enabling the larger vision of Artificial Intelligence (AI) adoption in context.

INDEX TERMS Classification algorithms, domain-specific languages, framework, intelligent systems.

I. INTRODUCTION

It is increasingly recognized that intelligent and adaptive systems are the systems of the future [1]. The era of static software and systems development that were the same independent of the environment is coming to an end and all application areas would need to add some ''intelligence'' that will allow them to learn and adapt as they operate and collect data. Furthermore, data collection and processing, commonly called data analytics, are going to be part of all systems and software in the modern big data era.

In order to add ''intelligence'' to systems the most commonly used algorithms are the classification algorithms. The current practice followed by the data scientists is to use libraries for classification algorithms such as *scikit-learn*, etc. that provide python implementation code for a wide range of algorithms. Most of the times, the data scientists are using directly the *scikit-learn* interfaces in order to run the algorithms and get the results. This is a detailed and not very user-friendly process.

There is a need to abstract the use of classification algorithms at a system-level and provide an interface that would be more user-friendly and achievable for the non-data scientist user. This need has been recognized in environments such the *weka* library that provide a user-friendly web interface.

However, in both the above approaches, an integrated environment where the classification algorithms would be considered at the system-level and in application context is missing.

On the other hand, recent research in software and systems engineering focuses on domain modeling and providing tools for abstracting technical implementation details and leaving the domain expert-user to handle only domain-related information. Domain-specific languages (DSLs) are a

The associate editor coordinating the review of this manuscript and approving it for publication was Zhe Xiao.

powerful tool for customized solutions that provide abstract and domain-specific interfaces.

In this research, we used the JetBrains MPS DSL development environment in order to develop a Classification Algorithms Framework (CAF). This abstracted the detailed information normally required for using the classification algorithms by offering the user a one-page interface. Last but not least, the developed CAF can be reused in other application domains through the language modularization offered by MPS. This can be accomplished by developing a new domain language corresponding to the application, such as a language for interfacing with business managers, medical professionals, education professionals, etc. and then using CAF in order to perform data analysis.

A background study on related work is presented in Section II. Specifically, we start with the definition of intelligent systems, existing research on frameworks for classification algorithms and research for system-level descriptions of intelligent systems. A background study of work on domain-related research from software engineering is also presented as it is part of our proposed solution. In Section III, our developed Classification Algorithms language-Framework is presented. The structure of the language is given through the corresponding diagrams of the concepts. The way the language can be used and extended as well as the strong points of the approach such as the code generation and extensibility are also demonstrated. In Section IV, the evaluation through feedback and workshops by BT and MPS engineers is presented. Finally, in Section V our conclusions and future plans are detailed.

II. RELATED WORK

To the best of our knowledge, there is no direct comparable work both in industry and academia.

This section consists of related work that attempts to address the general problem of designing and implementing intelligent or smart systems as they are called interchangeably in the literature.

By intelligent systems, we refer to all systems that make use of data collection and processing and adapt according to the outcomes of these processes. Intelligent systems range from systems that employ simple machine learning techniques (adaptive systems) to inform the users (human-in-the-loop) of the results from collecting and processing the data, leaving the final decisions to the users, to fully autonomic systems where the systems make decisions automatically and without human-in-the-loop [2]. In all cases and irrelevant of the level of automation, special attention and new techniques should be developed for intelligent systems in order to address their complexity [3] and their applicability to many contexts from business analytics [4] to IoT and smart cities [5].

On the other hand, the algorithmic Artificial Intelligence (AI) research has progressed so much in itself, and complicated algorithms have been developed. However, their applicability to several domains is being hindered by the difficulties in explaining these algorithms and encapsulating them at different contexts and at a system-level.

In this research, we have designed and developed a new language for using classification algorithms (CAF) in domain context. In the following subsections, existing work for classification algorithms frameworks, current industrial practice, system-level descriptions of intelligent systems and current status for domain-related and AI in context research are detailed.

A. EXISTING WORK

1) RESEARCH: FRAMEWORKS FOR CLASSIFICATION ALGORITHMS

The term ''Framework'' in general refers to providing a predefined set of common functionalities in order to assist a solution development. This can be applied to several domains with most known ones the web frameworks for web development (e.g. web2py, Django, etc.) and for different application contexts such as education in [6]. Frameworks can be software implementations that encapsulate expertise of some domain and can therefore be based and built upon architectural design patterns or a set of solutions [7] .

In this research, we developed a Classification Algorithms Framework (CAF) using domain-specific languages. According to our knowledge, there is no exact work that directly compares with this.

Most of the research work in ''Frameworks for Classification Algorithms'' falls in the category of trying to solve a particular application problem. For example, in [8], the focus is on benchmarking of classification algorithms applied to the domain of predicting software defects. In [9], a framework for characterizing electric energy consumers is being built using quality metrics. Both these works are focusing on how to evaluate classification algorithms' performance in specific applications. No work up to now focuses on providing the classification algorithms in their context and most of the research revolves around optimizing the algorithms or choosing a better algorithm in order to solve a problem. This type of research is algorithmic research and it is a continuing and very promising field. However, the advances in this field will be of limited use if the algorithms are not adopted in several application domains-contexts.

It is important to note that there are attempts to provide tools that aid the use of AI for problem solving, such as *DataRobot*, *Dataiku*, etc. However, none of these works approach the problem from the domain focus and particularly the separation between business and IT aspects that domain-driven design is bringing in our proposed solution. The *DataRobot* company is focusing on providing tools that aid the automated and rapid evaluation of potential predictive modelling solutions for prediction problems using statistical learning techniques. This approach utilizes AI to enhance software tool development (statistical learning to enhance the tool) for developing AI itself. This is an advanced technique that optimizes tool development (irrelevant of whether it is

for AI application development or not) but doesn't address the domain focus (through separation between business and IT aspects) that is required by modern complex applications. On the other hand, the *Dataiku* company is focusing more on providing tools for the different stakeholders through a collaborative team-based interface that would be very beneficial for the multidisciplinary and multiple-stakeholder projects of our times. The main difference with our proposed environment is that, through domain modelling using MPS Jetbrains DSL development, a personalized interface for each domain expert can be delivered; this will contain all the details relevant for their domain and abstractions for the rest. However, the idea of collaborative interfaces (from *Dataiku*) could be a useful future extension for our work, as well as using AI to enhance the developed tool (from *DataRobot*), In conclusion, none of these approaches utilize the latest research on domain modeling, with all of its corresponding benefits, such as separation of business and IT or domain focus (that abstracts implementation details from the user).

2) CURRENT PRACTICE MAINLY IN INDUSTRY: LIBRARIES FOR CLASSIFICATION ALGORITHMS AND NOT SYSTEM-LEVEL DESCRIPTIONS

In industrial applications and in many publications, libraries that provide code for classification and other data science algorithms are being extensively used. The following platforms were identified as most commonly used: *Weka* library is an open source Java-based platform. The advantage is that it is an approachable platform for new comers as it has a graphical user-interface for enhanced usability [10]. One of the disadvantages of this tool is that it does not include all the latest machine learning techniques. *Scikit-learn* is the most popular python-based framework built upon two existing libraries *NumPy* and *SciPy*. The advantage of using *scikit-learn* is the number of algorithms available to the developer and the fact that it provides source code in a widely used language such as *Python*. However, although many algorithms are provided, they are not tailored to solve deep learning [11]. *Deep Learning 4 J* is a framework that is primarily used for deep learning tasks [12]. *MLPack* is efficient as regards to speed [13]. *Caffe* is ideal for image recognition [14]. *Apache mahout* is used for distributed computing therefore offering scalability [15].

Summarizing and from our experience, the *weka* library and *scikit-learn* stand out the most with *weka* having the best user-interface and *scikit-learn* providing source-code python implementations and a wider range of algorithms.

However, the level of abstraction used in all these libraries is implementation/code-level. This is a big problem in complex systems as code-level often means at least thousands of lines making very hard to depict AI in context.

3) RESEARCH: SYSTEM-LEVEL DESCRIPTIONS OF INTELLIGENT-SMART SYSTEMS

The literature in the area of intelligent systems (smart systems in this context) and their system-level descriptions focuses on specific application domains and most commonly IoT-related applications and discusses about formalisms (SysML, SystemC, etc.) and simulations of complex systems. Smart systems are equivalent to IoT systems in these contexts as is presented recently in [16], [17].

Another area of research named Model-Based Machine Learning focuses on creating models for solving application problems that can be converted to machine learning algorithms [18]. Part of this direction is the Bishop's model [19]. This area is concentrating in the algorithmic requirements posed by applications and can be categorized as algorithmic research.

B. DOMAIN FOCUS IN ORDER TO PROVIDE AI IN CONTEXT

All the above research doesn't address the main issue we are increasingly facing with AI adoption: the applicability of AI algorithms in context. AI algorithms are very complicated and require extensive research in a specialized field that takes years to build. However, even the greatest performing algorithms won't be useful if they cannot be applied in real-world applications.

Domain modeling is a rising area of research in systems ad model-based engineering and most of the latest conference papers are presenting ''some kind of domain model'' as their innovative artefact.

The domain-related research has two main forms: domain-driven design and domain-specific modelling.

The domain-driven design (DDD) according to its pioneer books in [20], [21] (called the ''blue'' book and the ''red'' book in the domain-driven societies by Eric Evans and Vaughn Vernon) focuses on changing the way programmers code and making it more efficient. It's main strategy for this is the separation of the domain issues from the implementation which is effectively the evolution of the well-researched field of business-IT alignment [22]. When following the patterns defined by DDD pioneers, your code becomes a lot easier to handle and maintain and there is a massive trend in the industry for this approach. In order to benefit the most from DDD, a range of patterns and DDD principles should be applied for the duration of the project. In DDD, the system-level model is not used, and your model is your code, which means that the effort in learning modeling formalisms and applying them to application contexts is being replaced by learning DDD principles and applying them directly to your coding. In our experience, it is debatable if system models should be eliminated after all as the maintenance of the code and handling its complexity are more easily handled by system models and classical model-based approaches and not through enormous amounts of low-level code.

The domain-specific languages (DSLs) are the other domain-related research that is focusing on developing new languages to solve particular problems with emphasis on isolating the domain-specific interaction from the actual code implementation. Generally, the DSLs follow the domain-driven design principles as the underlying idea is the same.

However, they focus on a development of a new language that will give the domain expert an interface that targets only the domain and hides all other implementation details. They require effort and advanced technical expertise to build. However, once they are built, they are optimal for the specific problem [23]. Several advantages of DSLs have been detailed in the literature [24]–[26], the most important of which being the fact that they are usually smaller than general programming languages and are therefore more efficient and quicker to get used to.

There are two main streams of work in the area of DSLs: the xText based work and the JetBrains MPS development environment. In this work, we have chosen the JetBrains MPS development environment as it contains more high-level descriptions and better tooling according to our personal experience. Also, an important aspect for our application domain which is language composability is very strongly supported by JetBrains MPS as detailed in [27].

This capability stems from the projectional nature of the MPS editor. The languages in MPS, unlike in most today's language technologies, are not defined in terms of grammars and parsers. Instead, MPS languages are defined in an object-oriented manner using the notion of *Concepts*. *Concepts* define essential language elements, such as *IfStatement*, *VariableDeclaration*, *InputFileSpecification*, *ClassificationRule*, etc. *Concepts* specify their properties, children and references and may form inheritance hierarchies, just like, for example, classes in most general-purpose object-oriented languages. So, an *IfStatement* may be a sub-concept of a concept *Statement* and so may inherit some capabilities from it.

The actual models consist of *Nodes*. *Nodes* are instances of *Concepts* and form tree-like hierarchies with *Root Nodes* at the base of the trees and other nodes being organized in parent-child hierarchical relationships. These trees are generally called Abstract Syntax Trees (AST).

When the user desires to open a particular *Root Node* of the model, the MPS editor projects the *Nodes* that form the tree of that *Root Node* on the screen. The way the AST is projected on the screen and the way it reacts to user actions is defined by the editor aspect of the language. The visualization, also called notation, may vary depending on the user requirements. Structured text, non-textual symbols (e.g. math symbols), tables, diagrams - these are all possible, as well as their combinations. Language modularity is a natural consequence of projectional editing. The code of the model is never in an ''unresolved'' textual form, instead it consists of *Nodes*. Each of these *Nodes* unambiguously identifies the *Concept* it represents, irrespectively of the language that defines the *Concept*. Adding more languages to a project merely expands the virtual palette of *Concepts* that become available to the user.

III. PROPOSED SOLUTION – CLASSIFICATION ALGORITHMS FRAMEWORK

A. JETBRAINS MPS FOR DSL DEVELOPMENT

In order to develop the CAF, the JetBrains MPS environment was chosen [28].

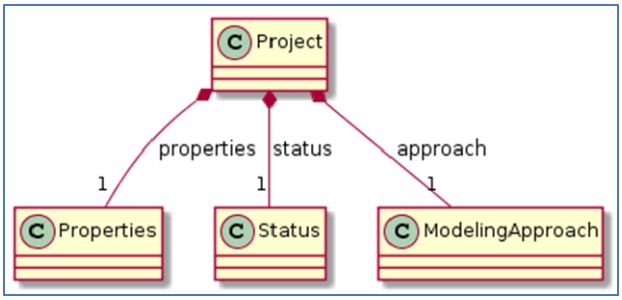

FIGURE 1. High-level project structure diagram.

This was decided for many reasons. The JetBrains MPS domain-specific languages environment offers a projectional editor which is a very powerful feature [29]. The fact that the user gets corrected as he/she types according to the underlying rules defined by the language inserts ''formality'' to the language designed and minimizes errors. The non-projectional editors don't offer this strong typing characteristics. Also, MPS offers excellent provision for automatic code-generation and model-to-model transformations through templates which is very beneficial.

B. CLASSIFICATION ALGORITHMS DSL

1) LANGUAGE STRUCTURE

The *Project* structure is depicted in Fig. 1 and consists of three main parts:

- The properties of the project (*Properties concept*) which define the input/output.
- The status (*Status* concept) which is connected with the results of running a classification algorithm.
- All the different classification algorithms and their parameters for calling them (*Modeling Approach* concept).

Each of the above concepts can be further analyzed as follows:

The *Properties* concept depicts interface logic between the project and the outside world. This communication happens through input/output files and tags. Specifically, in Fig. 2, the *Properties* concept and its main parts are as follows:

- *Tags* (as a property) which are keywords that are connected semantically to the domain of the application that these algorithms will be used.
- *Directory* (as child) which is the directory where the project is stored.
- *Dataset* (as child) which refers to the file that contains the input dataset file.
- *(Output) File* (as child) which contains the classification algorithm output (classification of the dataset and assigning to the endpoint categories).
- *(evalFile) File* (as child) which contains performance metrics for the chosen (ran) classification algorithm.

The *ModelingApproach* concept is depicted in Fig. 3:

The *ModelingApproach* concept consists of parameters for cross-validation and an interface to several classification algorithms.

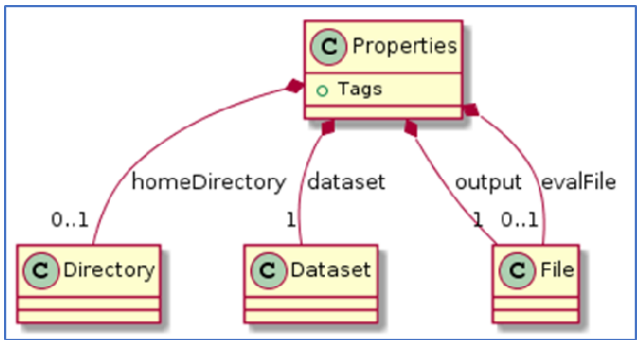

FIGURE 2. The *properties* structure description.

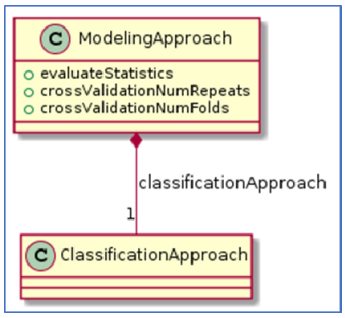

FIGURE 3. *ModelingApproach* concept.

In this DSL implementation, we have included cross-validation which is a machine learning technique for optimizing classification algorithms. The number of repeats and folds are defined as properties of the *Modeling Approach*.

A range of classification algorithms can be called using the *weka* libraries java calling implementation. The classification algorithms supported are depicted in Fig. 4.

FIGURE 4. Classification algorithms supported.

2) LANGUAGE EDITOR – I/O

The editor page is the interface provided to the user for configuring the framework by choosing the classification algorithm and its related parameters.

There is a one-to-one correspondence between the concepts defined in the *Structure* section of the language and editor screens provided to the users. Most of the concepts are represented in the editor and the user is required to provide input in order to run the classification algorithms and obtain the results.

There are three main steps in order to use the system and are detailed as follows:

Step 1: Input data and rebuild the model

The initial screen that appears when creating a new project in the sandbox is depicted in Fig. 5 together with example input data.

In Fig. 5, we can see how we can configure a new project by assigning the following:

1. Name - *Samples*
2. Home directory – *c:/Users/sofia/MPSProjects/ClassificationProject*
3. Input - *files/train.arff*
4. Output – *files/PredictionsNB.csv*
5. Evaluation file – *files/TrainNBeval.txt*
6. Approach = the choice of algorithm
7. Cross-validation parameters

Then you need to rebuild the model for the input to take effect.

Step 2: "Run" the selected classification algorithm

We "run" the generated code by right-clicking and choosing the *Run 'Node Samples(1)'* MPS option. The code is automatically generated by using MPS constructs and by encapsulating the *weka* library code. More on the automatic code generation in a following section.

Step 3: Load the Results in the same screen

In order to see the results of running the project, we click on the *LoadResults* as appears in Fig. 6.

Note that the *Load results* button is only working when there is some output file to be loaded. If you haven't run your simulation or have already loaded the results with the intention, the *Load results* button doesn't print any results and a *No results available* window message appears.

In order to implement the results to appear in the same page, we implemented a public class called *ResultProcessor* and the code is located inside the *Editor* language construct. The *ResultProcessor* class methods traverse information from the generated files (the *Output* and the *Evaluation file*) through going back and forth in the Abstract Syntax Tree of the language.

The final screen that contains the results of running the algorithms and performance evaluation is depicted in Fig. 6.

3) CODE GENERATION

The *ModelingApproach* concept consists of parameters for cross-validation and an interface to several classification algorithms.

The code generation (see Fig. 7) is a very strong point of this work as it provides Java code automatically which is ready for deployment.

For each algorithm the language author prepares a code generator template. This template contains java code that correctly configures and triggers the desired algorithm using the target library - *weka* in this case. The template is parametrized by the values inserted by the analyst into the model. In essence, the template takes care of the ceremony required to run an algorithm, while the model contains the important properties to initiate the algorithm.

The level of automation is quite high for several reasons. Only code that is necessary to be added to the high-level model is provided in the MPS templates and the process that needs to be followed is well-defined and simple. The language developer only needs to identify the points where

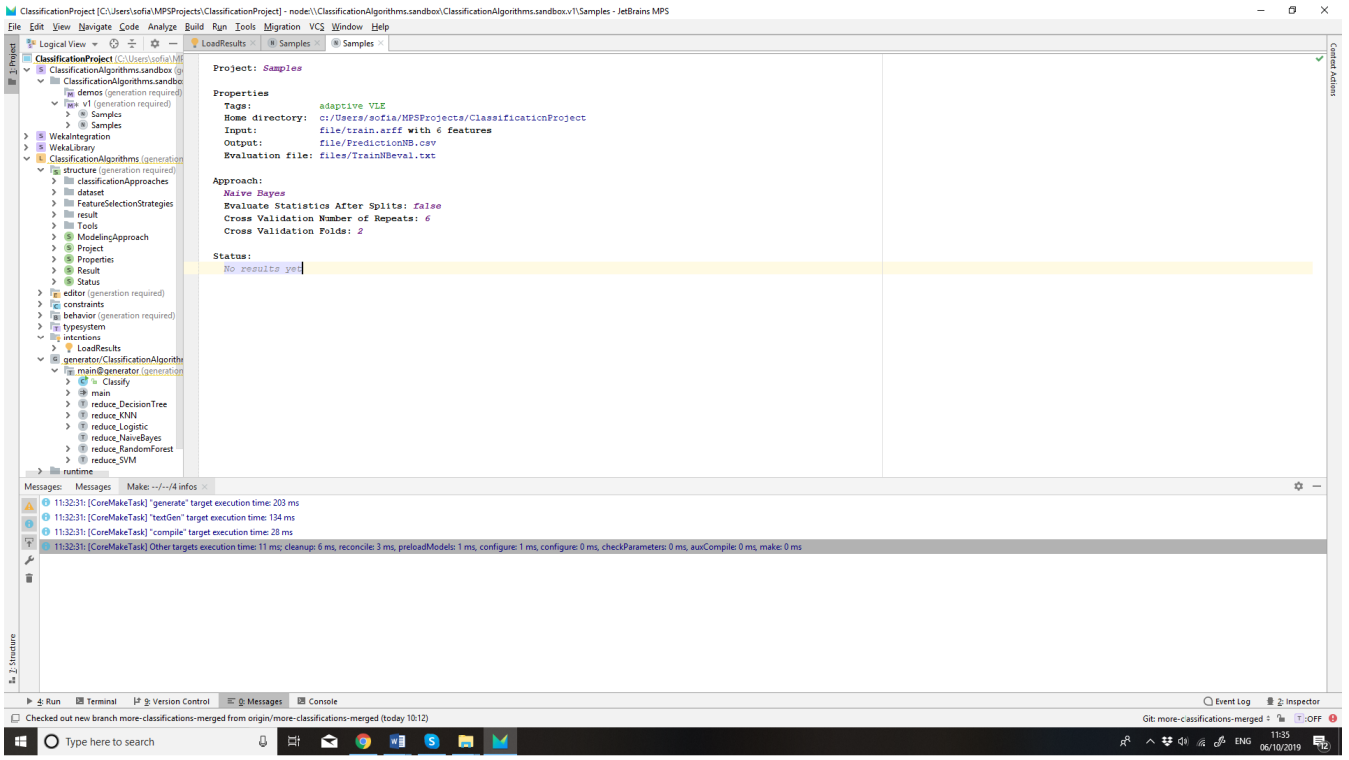

FIGURE 5. Step1. Example input data.

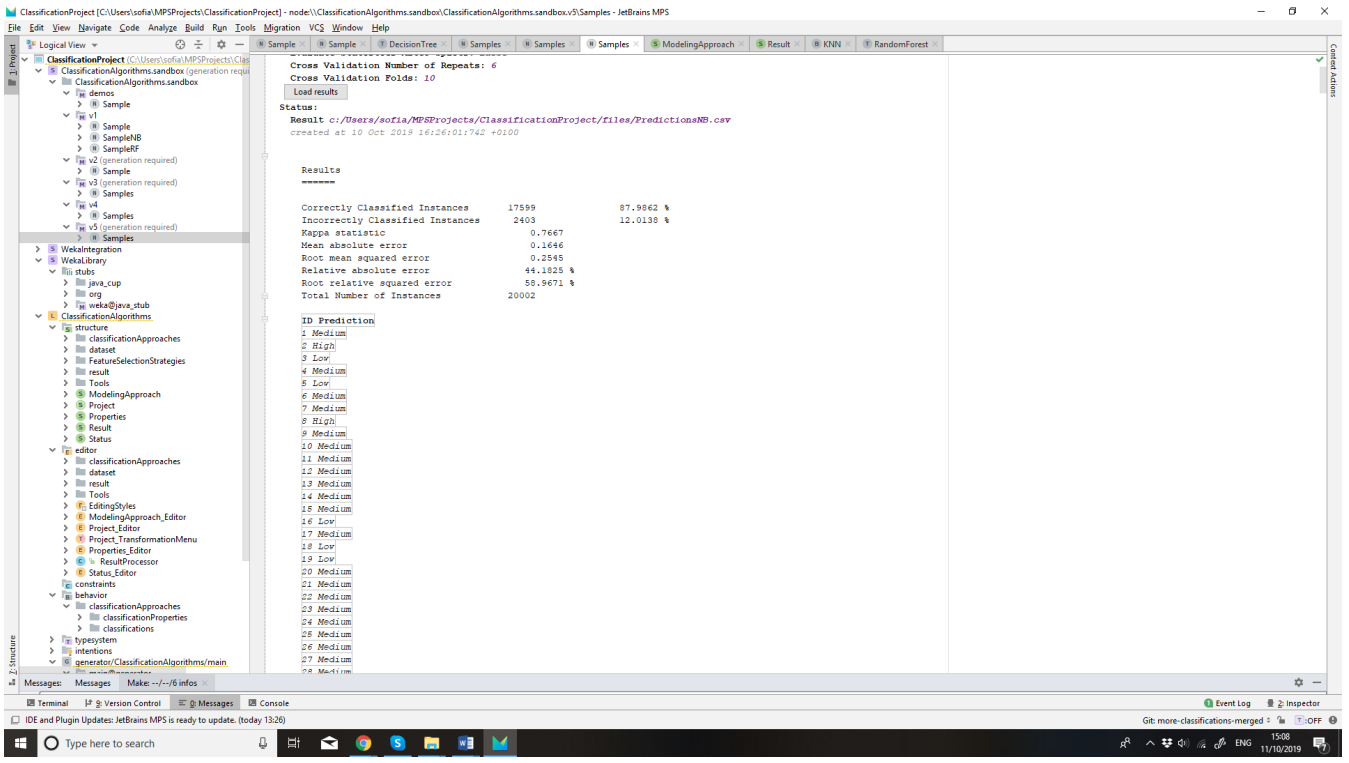

FIGURE 6. Loading of the results of running the classification algorithms in the same page.

the code should relate to the language high-level concept description and use the template to define that. For calling code from libraries such as *weka* libraries in this project, separate solutions were developed to include them.

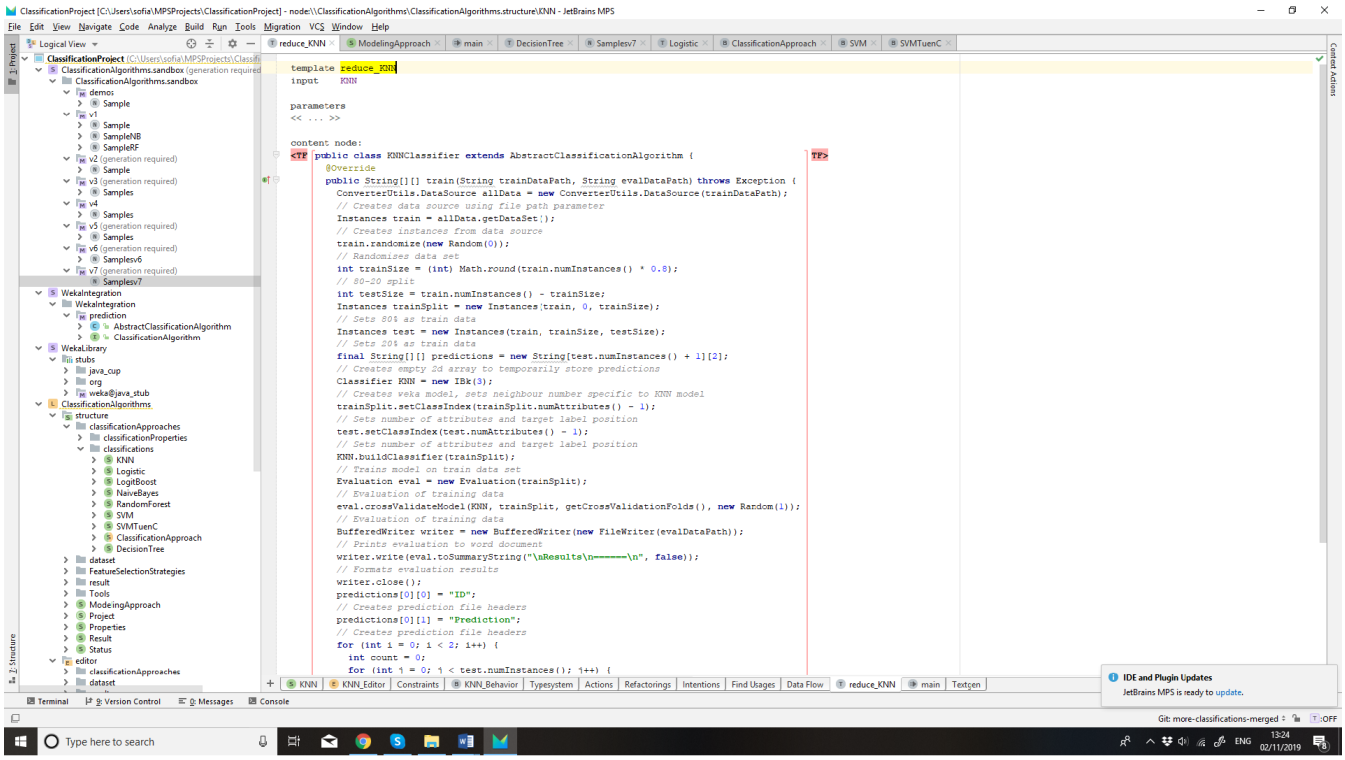

FIGURE 7. Jetbrains MPS code generation using templates.

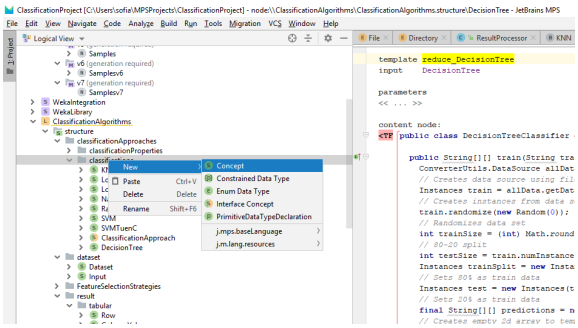

FIGURE 8. Adding new classification algorithms as concepts and inserting corresponding template code.

4) FRAMEWORK EXTENSIBILITY – ADDITION OF NEW CLASSIFICATION ALGORITHMS

The *ModelingApproach* concept consists of parameters for cross-validation and an interface to several classification algorithms.

This framework is easily extensible with more classification algorithms in three steps as depicted in Fig. 8:

Step1: Create a new concept that directly corresponds to the new classification algorithm.

Step2: Create a new reduction rule in the main of the generator aspect.

Step3: Fill in the additional code in the *reduction_template* with the links to the model concepts (see Fig. 7 above).

As part of our future plans, an editor with a user-friendly interface for the language designer will be built so that these steps can be automated. The user will be asked only the necessary information specific to the new algorithm implementation and the DSL will automatically add the new algorithm to its pre-existing structure.

IV. EVALUATION

We evaluated CAF according to the Quality characteristics defined in paper [23] and using the feedback we got from BT data scientists and MPS language developers (30 in total).

Several tasks from basic to advanced were set as part of a CAF workshop. The basic tasks involved using the language to choose and run a classification algorithm on a given data file. The advanced tasks included extending the language with more classification algorithms.

The results from the two different perspectives were as follows:

Data scientist perspective: The results from the data science perspective were as follows:

Functional suitability: CAF scored very high according to its suitability for all the functionality required. The domain was well specified, and it included processing of the data using different classification algorithms.

Usability: The feedback regarding the usability of the language was also quite good. The one-page interface that presents the results as well as the output in one page was very appreciated by the data scientists. It provided the option for quick experimentation of alternatives and in context. However, the appearance of the one-page interface could be further improved by integrating web technologies.

FIGURE 9. Evaluation through feedback by BT and MPS JetBrains.

Reliability: The language has precise semantics and the user is not ''allowed'' to make mistakes from the standard characteristics of the projectional editors. This was the most attractive feature to the users although it required some initial extra time of getting used to the editor.

Productivity: The use of this language will result in increased productivity for the data-scientists as they will be able to implement a quick turn-around time and experiment with several classification algorithms. Also, the system-level approach is expected to assist the adoption of the algorithms in several contexts.

Compatibility: The fact that java code is automatically generated from this language makes it very powerful for its deployment to application contexts.

Expressiveness: The language is very expressive as it has been developed through numerous meetings with BT's engineers and therefore extensive domain analysis has taken place.

Language designer perspective: The results from the language designer perspective were as follows:

Reusability: This language provides an excellent reusability element as it can be used as part of any system that requires data analysis. Its use within the education context proves the plug-and-play strong element of the language.

Maintainability: The process of adding new algorithms is not easy and requires technical expertise to understand and maintain it. Appropriate interfacing needs to be developed to enable easy extensibility of the language.

In Fig. 9 an Excel diagram depicts graphically the above results and shows that the average evaluation of the language is 8 out of 10 with strongest points the productivity, compatibility and expressiveness as is expected for domain-specific languages. Reusability has been scored excellent as this language is anticipated to be at the core of many adaptive systems applications. More evaluation and usage of the language inside BT's extensive commercial application areas is planned.

V. CONCLUSION AND FUTURE WORK

There is a gap in AI research, specifically in providing AI in context and at the system-level. Most of the research in the AI area has been focusing on algorithmic improvements and their performance. However, AI will have to be adopted in several different application contexts from business processes to autonomous cars. Domain modeling is a rising area in software and systems engineering, and in this paper, we presented its application for providing AI in context. We developed a new domain-specific modeling language, using the JetBrains MPS development environment, to model a Classification Algorithms Framework (CAF); this has as its main target to enable data scientists and other domain experts to perform quick, system-level and in-context experimentation of several algorithms. Our future plans include applying CAF in several business-related application contexts within BT and other application areas. In the longer term, and on more ambitious grounds, we will continue development of the language by ''blurring'' the limit between the algorithms and the domain, enabling optimal algorithm adoption in context.

REFERENCES

[1] S. Meacham, ''Towards self-adaptive IoT applications: Requirements and adaptivity patterns for a fall-detection ambient assisting living application,'' in *Components and Services for IoT Platforms*. Cham, Switzerland: Springer, 2017, pp. 89–102.

[2] M. T. Ibrahim, R. J. Anthony, T. Eymann, A. Taleb-Bendiab, and L. Gruenwald, ''Exploring adaptation & self-adaptation in autonomic computing systems,'' in *Proc. 17th Int. Conf. Database Expert Syst. Appl. (DEXA)*, Sep. 2006, pp. 129–138.

[3] P. Lalanda, J. A. Mccann, and A. Diaconescu, ''Future of autonomic computing and conclusions,'' in *Autonomic Computing* (Undergraduate Topics in Computer Science). London, U.K.: Springer, 2013, pp. 263–278.

[4] R. Akerkar, ''Employing AI in business,'' in *Artificial Intelligence for Business* (Springer Briefs in Business). Cham, Switzerland: Springer, 2019, pp. 63–74.

[5] A. A. Osuwa, E. B. Ekhoragbon, and L. T. Fat, ''Application of artificial intelligence in Internet of Things,'' in *Proc. 9th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Sep. 2017, pp. 169–173.

[6] S. Ivanova and G. Georgiev, ''Using modern Web frameworks when developing an education application: A practical approach,'' in *Proc. 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2019, pp. 1485–1491.

[7] R. E. Johnson, ''Frameworks = (components+patterns),'' *Commun. ACM*, vol. 40, no. 10, pp. 39–42, Oct. 1997.

[8] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, ''Benchmarking classification models for software defect prediction: A proposed framework and novel findings,'' *IIEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, Jul. 2008.

[9] V. Figueiredo, F. Rodrigues, Z. Vale, and J. Gouveia, ''An electric energy consumer characterization framework based on data mining techniques,'' *IEEE Trans. Power Syst.*, vol. 20, no. 2, pp. 596–602, May 2005.

[10] I. H. Witten, ''Data mining: Practical machine learning tools and techniques with Java implementations,'' *Acm SIGMOD Rec.*, vol. 31, no. 1, pp. 76–77, 2016.

[11] *Scikit-Learn: Machine Learning in Python—Scikit-Learn 0.21.3 Documentation*. Accessed: Nov. 1, 2019. [Online]. Available: https://scikit-learn.org/stable/

[12] *Deeplearning4j*. Accessed: Nov. 1, 2019. [Online]. Available: https://deeplearning4j.org/

[13] *Home*. Accessed: Nov. 1, 2019. [Online]. Available: http://www.mlpack.org/index.html

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, ''Caffe: Convolutional architecture for fast feature embedding,'' in *Proc. ACM MM*, 2014, pp. 675–678.

[15] J. Withanawasam, ''Apache Mahout essentials?: Implement top-notch machine learning algorithms for classification,'' clustering, and recommendations with Apache Mahout.

[16] M. Lora, S. Vinco, and F. Fummi, ''Translation, abstraction and integration for effective smart system design,'' *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1525–1538, Oct. 2019.

[17] M. Crepaldi, M. Grosso, A. Sassone, S. Gallinaro, S. Rinaudo, M. Poncino, E. Macii, and D. Demarchi, ''A top-down constraint-driven methodology for smart system design,'' *IEEE Circuits Syst. Mag.*, vol. 14, no. 1, pp. 37–57, 2014.

[18] *Model-Based Machine Learning (Early Access): An Online Book*. Accessed: Nov. 1, 2019. [Online]. Available: http://mbmlbook.com/
[19] C. M. Bishop, "Model-based machine learning," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 371, no. 1984, Feb. 2013, Art. no. 20120222.
[20] E. Evans, *Domain-Driven Design?: Tackling Complexity in the Heart of Software*. Reading, MA, USA: Addison-Wesley, 2004.
[21] V. Vernon, *Implementing Domain-Driven Design*. Reading, MA, USA: Addison-Wesley, 2012.
[22] J. Luftman and T. Brier, "Achieving and sustaining business-IT alignment," *California Manage. Rev.*, vol. 42, no. 1, pp. 109–122, Oct. 1999.
[23] B. Tekinerdogan and E. Arkin, "ParDSL: A domain-specific language framework for supporting deployment of parallel algorithms," *Softw. Syst. Model.*, vol. 18, no. 5, pp. 2907–2935, Oct. 2019.
[24] M. Challenger, G. Kardas, and B. Tekinerdogan, "A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems," *Softw. Qual. J.*, vol. 24, no. 3, pp. 755–795, Sep. 2016.
[25] A. G. Kleppe, *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*. Reading, MA, USA: Addison-Wesley, 2009.
[26] S. Benz and M. Völter, *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. Scotts Valley, CA, USA: Create Space Independent, 2013.
[27] M. Voelter, *Language and IDE Modularization and Composition with MPS*. Berlin, Germany: Springer, 2013, pp. 383–430.
[28] *MPS: The Domain-Specific Language Creator by Jet Brains*. Accessed: Nov. 1, 2019. [Online]. Available: https://www.jetbrains.com/mps/
[29] M. Völter and K. Solomatov, "Language modularization and composition with projectional language workbenches illustrated with MPS," in *Proc. 3rd Int. Conf. Softw. Lang. Eng. (SLE)*, in Lecture Notes in Computer Science, M. G. J. van den Brand, B. Malloy, and S. Staab, Eds. Springer, 2010.

SOFIA MEACHAM (Member, IEEE) received the Diploma degree in computer and informatics engineering and the Ph.D. degree from the University of Patras, Greece, in 1994 and 2000, respectively. She has been working in EU-funded projects as a Researcher/Embedded Software Engineer both in industry and in university, since 1995, and has accomplished a large amount of teaching experience in several institutions (U.K. and Greece), since 2000. She has strong links with British Telecom research headquarters in Adastral park and currently working in cutting-edge research in Explainable AI. She is also a Senior Lecturer with Bournemouth University and part of the Smart Technology Research Group (STRG). Her Ph.D. research interests fell in the area of system-level design for embedded systems, and include specification techniques for complex embedded telecommunication systems, hardware-software co-design, formal refinement techniques, and reuse practices. Her latest research interest involves methodologies (processes, tools, and methods) to improve design and development of systems, such as model-based design and domain-specific modelling for several applications from business processes to education.

VACLAV PECH received the master's degree in computer science from the Faculty of Mathematics and Physics, Charles University, Prague, in 1999. Since then, he has participated as a Developer and Consultant in various projects across Europe working mainly with server-side Java technologies and domain-specific languages. He is a Seasoned Software Developer and a programming enthusiast with 22 years of Java development and consultancy experience. He is currently involved in the MPS project with JetBrains.

DETLEF NAUCK received the Ph.D. degree and the Postdoctoral degree (Habilitation) in machine learning and data analytics. He is currently the Chief Research Scientist for Data Science with the Research and Innovation Division, BT, Adastral Park, Ipswich, U.K. He is leading a group of international scientists working on research into data science, machine learning, and AI. He focuses on establishing best practices in data science for conducting analytics professionally and responsibly leading to new ways of analyzing data for achieving better insights. Part of his role is leading the initiative on the development and the use of responsible and ethical AI in the company. He is a computer scientist by training. He is also a Visiting Professor with Bournemouth University and a Private Docent with Otto-von-Guericke University Magdeburg, Germany. He has published three books, over 120 articles, holds ten patents, and has 30 active patent applications.