Towards a DSL for AI Engineering Process Modeling

Sergio Morales[1][0000−0002−5921−9440], Robert Clarisó[1][0000−0001−9639−0186], and
Jordi Cabot[1,2][0000−0003−2418−2489]

[1] Universitat Oberta de Catalunya, Barcelona, Spain
{smoralesg,rclariso,jcabot}@uoc.edu
[2] ICREA, Barcelona, Spain jordi.cabot@icrea.cat

Abstract. Many modern software products embed AI components. As
a result, their development requires multidisciplinary teams with diverse
skill sets. Diversity may lead to communication issues or misapplication
of best practices. *Process models*, which prescribe how software should
be developed within an organization, can alleviate this problem. In this
paper, we introduce a *domain-specific language* for modeling AI engi-
neering processes. The DSL concepts stem from our analysis of scientific
and gray literature that describes how teams are developing AI-based
software. This DSL contributes a structured framework and a common
ground for designing, enacting and automating AI engineering processes.

Keywords: Domain-Specific Language · AI Engineering · Process Mod-
eling

1 Introduction

Modern business applications usually embed Machine Learning (ML) and other
Artificial Intelligence (AI) components as core of their logic [3, 6]. The engineer-
ing of AI components requires the introduction of new development activities
and profiles in development teams beyond software engineers, *e.g.*, data scien-
tists, psychologists and AI experts. As a result, there is a need for more support
and guidance when developing AI projects, as reported in recent studies [2, 8, 15].
Enterprises need to revise their practices and adopt a clear process for building
AI products.

A *process model* provides full visibility and traceability about the work de-
composition within an organization, along with the responsibilities of their par-
ticipants and the standards and knowledge it is based on. Process models are
guidelines for configuration, execution and continuous improvement.

In this sense, we propose a *domain-specific language* (DSL) to facilitate the
specification of AI engineering processes. The motivation for a DSL is to have
a shared language in a particular problem space that can foster communication
and collaboration. Our DSL encompasses standard process modeling concepts
plus AI-specific process primitives based on the analysis of research and gray

literature. We currently have introduced concepts from Machine Learning and leave other AI activities and facets for future work.

Our DSL facilitates the definition of AI engineering processes by enabling stakeholders to discuss and specify a single and formalized representation of such processes. This also brings additional benefits. For instance, it opens the door to automatic processing, *e.g.*, as part of a process execution scenario, it facilitates the detection of hidden or conflicting practices, and simplifies the onboarding of new team members.

The remainder of this paper is organized as follows. Section 2 reviews existing proposals for AI processes. In Section 3 we introduce our DSL, while in Section 4 we present an example of its usage, leading to Section 5 where we analyze other related work. Finally, Section 6 concludes and outlines the future work.

2   Background

There are several scientific papers and gray literature describing the development of real AI projects. Among those, we have selected the most cited research publications and influential contributions from the industry as inspiration for our DSL.

In particular, we have chosen 3 industrial methods: CRISP-DM [5] as the de facto standard process for data science projects; and Microsoft Team Data Science Process [14] and IBM AI Model Lifecycle Management [9] as two major players in the field. We have also included 3 scientific references that discuss the Machine Learning lifecycle in actual applications [2, 4, 10]; and 1 paper that blueprints a maturity framework for AI model management [1].

Each of those proposals has a slightly different grouping, distribution and granularity of activities, but all share the following high level structure:

1. *Business Understanding*, to set the business objectives and criteria linked to an AI project, and produce a plan along with an initial assessment of tools and techniques.
2. *Data Collection & Preparation*, to perform activities to gather and clean data, and prepare datasets and features for creating AI models.
3. *AI Model Training & Evaluation*, to select AI modeling techniques, optimize hyperparameters and train the AI models, which will be evaluated and ranked according to evaluation and business criteria.
4. *Production & Operation*, to finally make the AI models available for consumption and to build a monitoring system and pipelines for continuous improvement.

Our DSL generalizes and unifies these concepts to enable the specification of end-to-end AI engineering processes.

3   DSL Design

A DSL is commonly defined through a metamodel that represents its domain entities and their relationships. As shown in Figure 2, at its core, our DSL

contains the generic description of activities, the relationships between them, and the main elements they are related to. Based on the analysis of existing literature, we predefine four main activities (see Figure 1): (1) *BusinessActivity*, (2) *DataActivity*, (3) *AIModelingActivity*, and (4) *AIModelDeploymentActivity*.
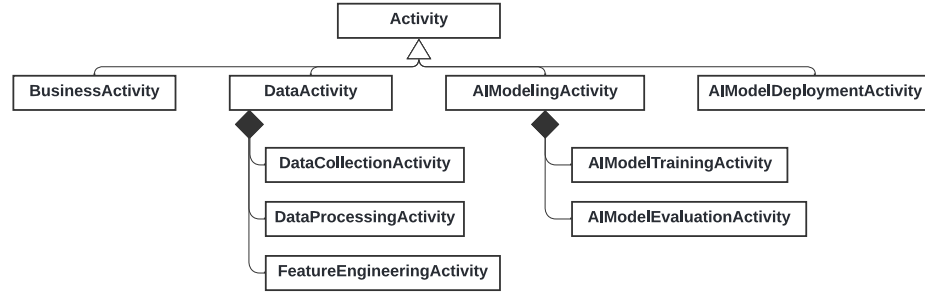
Fig. 1. High-level view of activities and subactivities.

In the next sections, we focus on the two most AI-specific ones: *DataActivity* and *AIModelingActivity*. We only briefly cover the *AIModelDeploymentActivity* and leave the *BusinessActivity* for future work. Due to lack of space, we describe an excerpt of the DSL. The complete metamodel is available online[1].

Note that our DSL does not prescribe any concrete AI engineering process model. Instead, it offers the modeling constructs so that each organization can easily define its own process.

3.1   Activity Core Elements

An *Activity* constitutes the core element of any process. Activities are composed of other activities (association *composedOf*). Completing an activity may require completing all subactivities (attribute *requiresAllSubactivities*). Process creators define if an activity is mandatory (attribute *isOptional*). There may also be a precedence relationship between activities (association *next*).

Several *Roles* perform the activities during development. Their *participation* could be specified according to the organization's levels of responsibility, *e.g.*, as responsible or accountable (class *Participant*).

Activities consume (*inputs*) and produce (*outputs*) *Artifacts*. An artifact could be a document that is generated as an output of an activity and is consumed as an input by the following one. Other examples of artifacts will be studied in the following sections.

Finally, *Resources* might be helpful to complete an activity. Resources are not consumed nor produced by the process –they are supporting components. An example would be a template for the document from the previous paragraph.
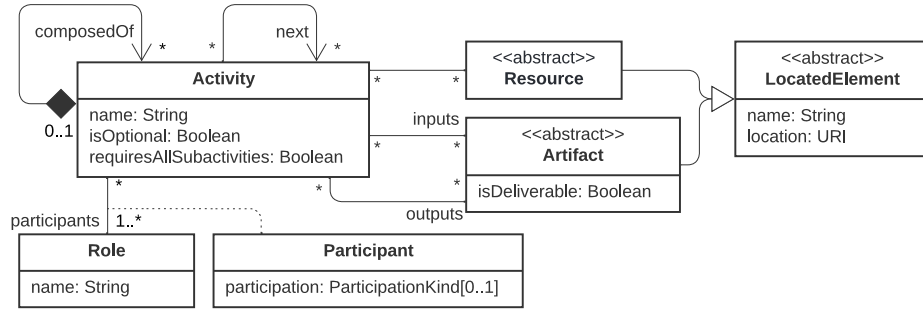
---

[1] http://hdl.handle.net/20.500.12004/1/C/PROFES/2022/422

Fig. 2. Generic elements of an activity.

3.2 Data Activity

The *DataCollectionActivity* is the acquisition of *DataInstances* from *DataSources*. The participants move data from internal or external data sources (attribute *isExternal*) into a destination (attribute *location*) for further processing.

In the *DataProcessingActivity*, data is cleaned and transformed via different *techniques* (*e.g.*, dummy substitution for cleaning empty values in relevant attributes, and data reduction or augmentation) to overcome deficiencies that might result in bad predictions if used for training an AI model. Additionally, data could be labelled to help AI models identify concepts in production data.
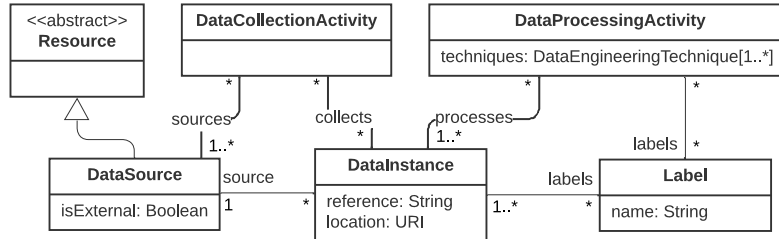

Fig. 3. An excerpt of activities and other elements of the *DataActivity*.

The *FeatureEngineeringActivity* (not included in Figure 3) comprehends the tasks and statistical techniques used to transform data attributes of a *DataInstance* into features that can be used by an AI model and enhance its prediction accuracy. During this activity, correlations between features are identified. As a result of this activity, a set of features are extracted from the data instances.

Data is then usually split into three disjoint sets: a training dataset, a validation dataset and a test dataset.

3.3   AI Modeling Activity

The *AIModelTrainingActivity* is the activity for creating, training and validating new AI models from the collected and prepared data. An *AIModel* is trained by an AI algorithm using the observations held in the *TrainingDataset*.

Once an AI model is initially trained, a data scientist tunes its *Hyperparameters* looking for the *OptimalValues* that yield its best performance. The *ValidationDataset* is applied in this procedure. Finally, the hyperparameter values that maximize an AI model performance are fixed for production.

The *AIModelPerformanceCriteria* will drive the AI model training and will be used to pursue an AI model or discard it; in other words, they dictate when it is not worthwhile to keep improving an AI model.
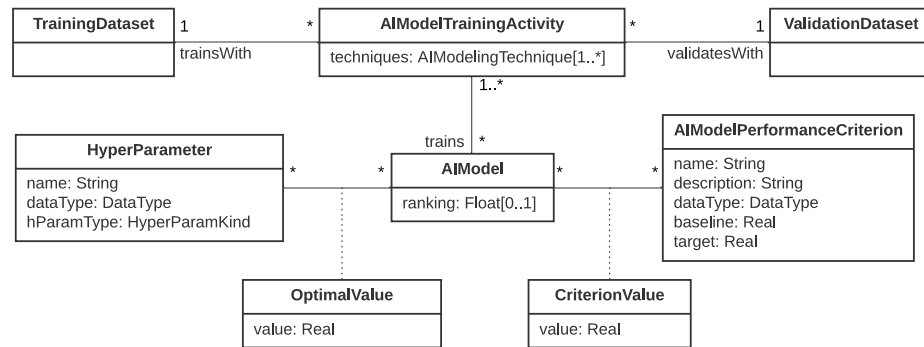

Fig. 4. An excerpt of the *AIModelingActivity* and its elements.

In the *AIModelEvaluationActivity* (not part of Figure 4), a data scientist checks if an AI model satisfies the AI model success criteria, along with its adequacy to its AI model requirements. A test dataset is used to assess this. Data scientists then set a *ranking* for each AI model.

3.4   AI Model Deployment Activity

In the *AIModelDeploymentActivity*, an AI model is deployed to a production *Platform* to serve end users or other systems. It may be useful to run *Scripts* to automate its installation and setup. An AI model can be deployed (attribute *pattern*) statically, dynamically (on the user's device or on a server), or via streaming. An AI model can make its *inferences* either: (a) in batch mode, periodically making predictions offline and serving the results to a repository; or (b) in real-time, making and serving predictions whenever requested to.
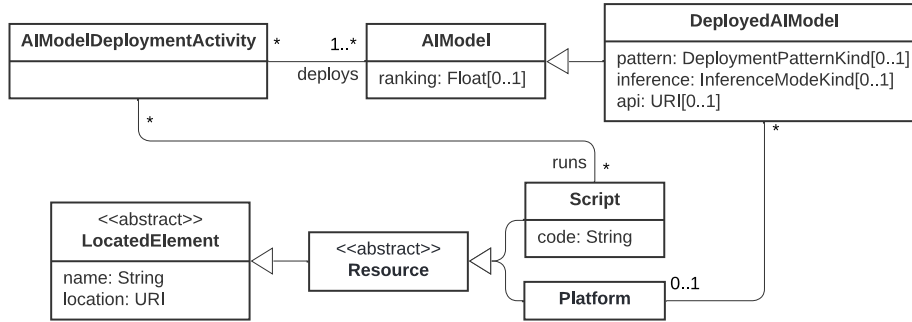
Fig. 5. An excerpt of elements of the *AIModelDeploymentActivity*.

4   Tool Support

We have implemented our DSL on top of Sirius Web[2], an open-source subproject of Eclipse Sirius. Given a metamodel and its mapping to a set of visual notation elements, Sirius Web generates a modeling environment that can then be used by modelers to design new graphical models conforming to that metamodel.

As an example, Figure 6 depicts an excerpt of the *DataActivity* of a simple process model with four subactivities: (1) *Ingest the data*, (2) *Clean the data*, (3) *Reduce the data*, and (4) *Set the data pipeline*. The first one is an instance of the *DataCollectionActivity* and employs a technique (*Load data to SQL Server on Azure VM*) for transferring data from the data source *Employees ERP* into the data instance *Extraction ref A0O451*. The activity *Ingest the data* has one participant, a *Data engineer*, who is *responsible* for its execution. The activities *Clean the data* and *Reduce the data* are instances of the *DataProcessingActivity*, and each of them performs different techniques to process the data instance.
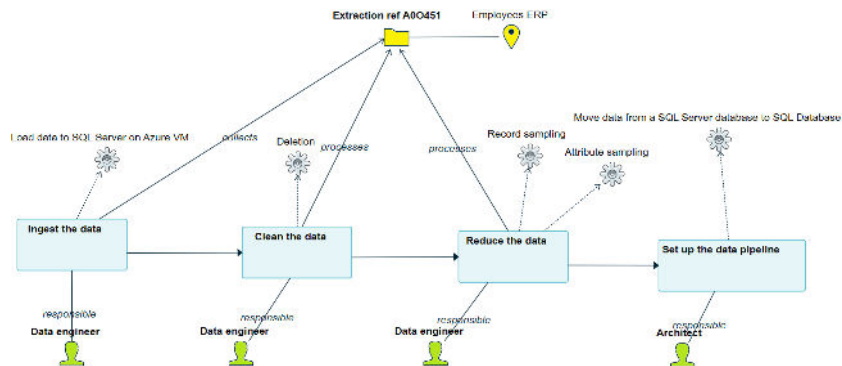

Fig. 6. A sample process model created with our DSL on Sirius Web.

---

[2] https://www.eclipse.org/sirius/sirius-web.html

The DSL provides flexibility for adding elements that are specific to a method. In the example, *Set up the data pipeline* does not correspond to any predefined AI activity described in our DSL. Therefore, it is based on the generic *Activity*.

5   Related Work

There are dozens of process modeling languages, *e.g.*, BPMN & SPEM and their extensions, UML profiles, and formal languages [7]. Specifically, SPEM is an OMG standard for describing software development processes, but it purposely does not include any distinct feature for particular domains or disciplines –like Artificial Intelligence. To the best of our knowledge, none of the process modeling languages includes AI specific extensions.

Regarding DSLs for AI, there are languages to model certain AI activities such as OptiML [13], ScalOps [16], Arbiter [18] or Pig Latin [11]. There are also DSLs for creating AI artifacts like ML-Schema [12], an ontology for interchanging information on ML experiments, or DeepDSL [17] for the creation of deep learning networks. Nevertheless, none of those DSLs focus on process aspects.

Therefore, as far as we know, our DSL is the first that provides elements for describing AI-specific activities and enables modeling AI engineering processes.

6   Conclusions and Future Work

In this paper, we have presented a first version of a DSL to model AI engineering processes. Our language covers the needs for such type of processes as described in academic and industry proposals. We believe this DSL is a step forward towards the adoption of software engineering practices in the AI area.

Our DSL will facilitate the formalization of AI processes within organizations. Moreover, this formalization will also enable the manipulation of the models via any of the existing model-driven tools –especially the EMF-based ones, which will be directly compatible with our DSL implementation.

As further work, we plan to create a tool set that would enable enacting and automating these modeled AI processes, thus providing real-time information of running processes and guidance for intervention.

Additional future work will involve extending the DSL. In particular, we will dive deep into the *BusinessActivity* for contextualizing and setting business purposes to AI projects. Similarly, we plan to enrich the *AIModelDeploymentActivity* to incorporate monitoring elements to ensure the performance of deployed AI models remains within acceptable limits. Besides, we will go beyond ML and include other AI methods. We will also add process snippets and templates that would help companies to create their own process without starting from scratch. Finally, we will empirically validate the usability of our DSL.

References

1. Akkiraju, R., Sinha, V., Xu, A., Mahmud, J., Gundecha, P., Liu, Z., Liu, X., Schumacher, J.: Characterizing machine learning processes: A maturity framework. In: Business Process Management. pp. 17–31. Springer (2020)
2. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., Zimmermann, T.: Software engineering for machine learning: A case study. In: ICSE-SEIP. pp. 291–300 (2019)
3. Anthes, G.: Artificial intelligence poised to ride a new wave. Communications of the ACM **60**(7), 19–21 (2017)
4. Ashmore, R., Calinescu, R., Paterson, C.: Assuring the machine learning lifecycle: Desiderata, methods, and challenges. ACM Comput. Surv. **54**(5) (2021)
5. CRISP-DM. https://cordis.europa.eu/project/id/25959, last accessed 6 Jun 2022
6. Deng, L.: Artificial intelligence in the rising wave of deep learning: The historical path and future outlook. IEEE Signal Proc. Mag. **35**(1), 180–177 (2018)
7. García-Borgoñón, L., Barcelona, M., García-García, J., Alba, M., Escalona, M.: Software process modeling languages: A systematic literature review. Information and Software Technology **56**(2), 103–116 (2014)
8. Hill, C., Bellamy, R., Erickson, T., Burnett, M.: Trials and tribulations of developers of intelligent systems: A field study. In: 2016 IEEE Symposium on VL/HCC. pp. 162–170 (2016)
9. IBM AI Model Lifecycle Management. https://www.ibm.com/blogs/academy-of-technology/ai-model-lifecycle-management-white-paper, last accessed 6 Jun 2022
10. Nascimento, E.d.S., Ahmed, I., Oliveira, E., Palheta, M.P., Steinmacher, I., Conte, T.: Understanding development process of machine learning systems: Challenges and solutions. In: ESEM 2019. pp. 1–6 (2019)
11. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A not-so-foreign language for data processing. In: SIGMOD. p. 1099–1110. ACM (2008)
12. Publio, G.C., Esteves, D., Ławrynowicz, A., Panov, P., Soldatova, L., Soru, T., Vanschoren, J., Zafar, H.: ML-Schema: exposing the semantics of machine learning with schemas and ontologies. arXiv preprint arXiv:1807.05351 (2018)
13. Sujeeth, A.K., Lee, H., Brown, K.J., Rompf, T., Chafi, H., Wu, M., Atreya, A.R., Odersky, M., Olukotun, K.: OptiML: An implicitly parallel domain-specific language for machine learning. In: ICML. pp. 609–616 (2011)
14. What is the Team Data Science Process? https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview, last accessed 6 Jun 2022
15. Wan, Z., Xia, X., Lo, D., Murphy, G.C.: How does machine learning change software development practices? IEEE Trans. on Software Eng. **47**(9), 1857–1871 (2021)
16. Weimer, M., Condie, T., Ramakrishnan, R., et al.: Machine learning in ScalOps, a higher order cloud computing language. In: NIPS. vol. 9, pp. 389–396 (2011)
17. Zhao, T., Huang, X.: Design and implementation of DeepDSL: A DSL for deep learning. Computer Languages, Systems & Structures **54**, 39–70 (2018)
18. Zucker, J., d'Leeuwen, M.: Arbiter: A domain-specific language for ethical machine learning. In: AAAI/ACM Conf. on AI, Ethics, and Society. pp. 421–425 (2020)