Applying i* in Conceptual Modelling in Machine Learning

Jose M. Barrera[1,2], Alejandro Reina[1,2], Alejandro Maté[1,2] and Juan C. Trujillo[1,2]

[1]*Lucentia dpt. of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n, Alicante 03690, Spain*

[2]*Lucentia Lab, Av. Pintor Pérez Gil, 16, 03540 Alicante*

Abstract

The i* framework is a popular and well-equipped technique for capturing the organizational environment and requirements of a system. However, i* heavily depends on the designer experience to cope with the idiosyncrasy of each specific field. While the machine learning field would benefit from a requirements representation, its complexity makes it unfeasible to directly use i*. The large number of constructs and nuances between elements puts a severe strain on the designer, leading to the creation of error-prone models. Therefore, in order to tackle this problem, we present an extension of i*. Our proposal covers the main gaps between machine learning and conceptual modeling with the aim of creating a suitable baseline methodology for machine learning requirements engineering. The advantage of our proposal is that our language specifies the main elements involved in machine learning models and constrains their interactions, filtering invalid designs and thus reducing the burden of knowledge while making the process less error-prone.

Keywords

machine learning, iStar, requirements engineering, conceptual modelling, methodology

1. Introduction

Machine learning (henceforth dubbed ML) is a specific field inside the Artificial Intelligence. The ML algorithms "learn" from data with the aim of performing different tasks such as forecasting, classifications, or clustering among others. Thus, a ML model can answer questions like "Will this patient develop a COVID morbidity?", "How much this house will cost in 5 years?" or "Will be an investment in a solar energy installation profitable in 3 years?".

Despite its widespread use, there is no methodology or language for capturing ML requirements. Therefore, most ML projects rely on the expertise of individual data scientists or data analysts to frame the problem at hand and translate implicit ML requirements into the actual implementation. While i* is suitable for capturing user requirements, it is defined on a very high abstraction layer so that it is flexible enough to be applied to almost every field. Therefore, it often lacks the constructs and rules specific to the field where it is being applied to. As such,

Proceedings of the 14th International iStar Workshop, October 18-21, 2021, St. Johns (NL), Canada

jm.barrera@ua.es (J. M. Barrera); alejandro.reina@ua.es (A. Reina); amate@dlsi.ua.es (A. Maté); jtrujillo@dlsi.ua.es (J. C. Trujillo)

0000-0002-3359-2685 (J. M. Barrera); 0000-0001-9949-2735 (A. Reina); 0000-0001-7770-3693 (A. Maté); 0000-0003-0139-6724 (J. C. Trujillo)

it is commonly tailored by adding new elements that provide a suitable RE language for the domain at hand. Ideally, a methodology for capturing ML requirements would be accompanied by a requirements language that captures the relevant concepts and relationships in the field, providing data scientists a familiar language to frame the problem at hand.

In recent years, some works have started exploring the applicability of i* to machine learning projects. For example, [1] used i* to capture non-functional requirements (NFR's). Nevertheless, the use of i* on the ML field has been scarce and the has not been properly explored yet from a data scientist or ML engineer point of view.

Therefore, in this paper, we propose an i* extension and a methodology considering the whole ML process and its associated requirements. Our proposal covers the main gaps between machine learning and conceptual modeling with the aim of creating a suitable baseline methodology for ML requirements engineering. The advantage of our proposal is that our language specifies the main elements involved in machine learning models and constrains its interactions, filtering invalid designs and thus reducing the burden of knowledge on the designer while making the process less error-prone. For instance, in an unconstrained language one could associate an accuracy metric (a specific classification metric) to the result of a KNN algorithm (a specific clustering task): both elements are related with ML, but their relationship is incorrect. Going further, one could use an accuracy metric in an unbalanced classification problem. While this relationship is semantically correct, the chosen metric is ill-defined for these problems.

The remainder of the paper is structured as follows: Section 2 presents the related work. Section 3 presents our extension of i* applied to ML. Finally, section 4 provides the discussion about the results of the model.

2. Related Work

The i* framework is one of the most widespread Goal-oriented Requirements Engineering (GORE). It is a basic tool for modeling and analysis at both the business and service design level, taking advantage of its agent orientation for modeling service relationships, and its goal orientation to facilitate adaptation from generic patterns to specific needs [2].

Despite many extensions across the years, the use of i* in ML has not received much attention until very recently. In [1] the authors propose the first approach of i* applied to ML, focusing on non-functional requirements (NFRs). However, the proposed approach ignores the rest of the aspects involved in an ML project. Consequently, the i* model is not extended with additional elements related to the field. In [3] the authors show how requirements modelling can be applied to the ML field. Nevertheless, they provide no extension, preserving the high abstraction layer of i* while lacking specific constructs from the ML field. Therefore, the construction of valid models relies entirely on the designer experience in both ML and i*, resulting in an error-prone process where inconsistencies such as using invalid metrics for the models at hand are likely to appear. Finally, in [4], the authors focus their efforts on testing and monitoring the effectiveness of a ML model. However, their study does not cope with requirements capture.

As i* is a widespread language, other extensions have been presented in recent years. In [5] and [6], the authors present an extension of i* adapted to data warehouses, enhancing its manageability, as well as the comprehension capability of the user when dealing with complex models. However, these extensions are focused on data warehousing and their applicability on machine learning is limited.

As shown, existing approaches mostly rely on the experience of the data scientist and ML engineer to use and adapt i* in order to capture them. Compared to these approaches, our proposal extends i* framework with the information needed from a machine learning engineer point of view, minimizing the errors that appear with a more generalist approach. Thus, our model is more restrictive but semantically valid. Second, since our proposal is more directed, even less experienced ML designers will not skip any important aspect that should be considered in a ML project. Third, thanks to the reduction of errors and the clearer process, the ML project is likely to be developed in a shorter time. Fourth, our extension tries to follow the methodology and guidelines described in [7] for i* extensions. However, note that due paper constraints a graphical notation could not be included.

3. Requirements Engineering in Machine Learning projects

In this section, we present our proposal for tackling RE in ML based on i*. The aim of our proposal is to serve as baseline approach to aid in the construction of valid models.

3.1. Capturing Machine Learning requirements

In order to implement a ML project, there are a series of questions that must be answered, allowing ML designers to frame the problem at hand. Moreover, the ML model may require some specific qualities (as for instance explainability or being capable of dealing class imbalances). Consequently, these requirements must be captured in advanced in order to constraint ML algorithms and tasks. Exploring these questions allows us to establish which new constructs must be added to the base i* while at the same time providing a guideline for practitioners to fill their requirements model.:

1. **Which problem must be solved?:** The objective needs to be clearly established at a qualitative level. By answering this question, we are limiting the valid ML models. This answer will provide the highest abstraction layer goal of the project. This goal will be split into more specific goals (as *MLGoals*).
2. **In which time frame should we have the answer?:** By answering this question we constrain the use of data for training the model. Adding this information modifies the model training data. This answer will affect the *Sources* and *Dataset*. For instance, if our model must predict if a patient will die or not with a 7 days margin, the last 7 days of data must be removed from the historic training data with the aim of provide a proper prediction.
3. **Which data do you think is important for the model?:** The answer to this question is not trivial and it should be provided by domain experts. This answer will provide the different *Sources* that will conform the *Dataset* element.
4. **Which granularity of data is available to tackle the problem?:** It is relatively easy to aggregate data at low granularity level to create answers based on a higher abstraction layer. On the other hand, handling data with an excessively small granularity can

impact the time cost of the model. Thus, with this information, *temporalResolution* and *refreshPeriod* parameters belonging to *Dataset* will be parametrized.

5. **Which metrics and hit rate would be valid to consider the project as a success?:** It is important to establish the appropriate model metric as well as the threshold that allows the model to be accepted. Depending on the nature of the problem, a specific variable (such as sensitivity or precision) may need to be used instead of accuracy.
6. **Is the explainability of the model necessary?:** Some machine learning techniques allow for a detailed description of which which features have been given more importance, while others (e.g. deep learning) do not provide them at all. Depending on the field of application or the project, the explainability can be mandatory.
7. **Is it likely that the data distribution will change?:** Traditional machine learning models obtain the distribution function based on the data they have been trained on. However, depending on the nature of the project, it is possible that the trend of the data may vary (known as Concept Drift) [8] which must be taken into account.
8. **Is there any bias in the data? Is data fair from an equity point of view?:** Biases in data are common in real projects. They can lead to unwanted effects, such as racism or xenophobia [9]. To avoid this result, there are three options: *Remove training data*, *Remove the variable that implies a bias*, or *Relabelling training data*.This answer will categorize Equity, a specific parameter of *MLQualityAspect*.

3.2. ML proposed model

The proposed metamodel is an extension of the i* 2.0 proposed in [10]. i* elements are represented in blue, whilst the new elements proposed tailored for ML requirements are represented in yellow (concepts) and green (enumerations). The extension has been defined respecting the core i* 2.0. As such, any element or attribute apparently missing has been omitted due to space constraints. The newly added elements include both new properties and relationships that must be specified for carrying out ML projects. For example, in ML projects it is mandatory to specify how the models will be validated and evaluated as well as a temporal resolution which describes the granularity of the expected prediction. Failing to specify this information, the ML models cannot be configured and trained adequately, leading to the failure of the ML project. In the following, we describe in more detail each of the newly added elements, which can be seen in Fig. 1.

First, a Source represents a data source. The information from one or more sources can be combined into a *DataSet*, which is used by *MLModels*. Each dataset has a *temporalResolution* and *refreshPeriod* imposed by the data itself.

Next, *MLGoals* are divided into 3 different children classes *ClassificationGoals*, *RegressionGoals* and *ClusteringGoals*. *ClassificationGoals* and *RegressionGoals* have a *temporalResolution*, which allows us to detect conflicts between the requirements and the datasets at hand.

Furthermore, each specific particular goal class has a relationship with their corresponding Indicator class. These indicators allow us to specify and measure the degree of satisfaction of the associated goals (desired value vs actual value). In order to avoid design errors indicators are drawn from an enumerated list.

Our model also includes an abstract *MLTask* class. This class includes specific attributes which are common to all ML tasks. For instance, all ML algorithms must be trained and tested. In order to do that, the *MLTask* allows to specify i) which method will be used for splitting the data, ii) which will be the size of the train, testing and validation sets, sets and iii) how many folds will be used if cross validation is selected.

In order to represent the specific nature of each *MLTask*, the class is specialized into three more specific elements. *ClassificationTask* includes a *binaryClass* attribute which denotes whether the classification is binary or multilabel (three or more classes). Furthermore, both *ClassificationTask* and *RegressionTask* include the possibility of creating aggregated ML models through bagging or boosting. Each of these three classes uses an enumeration with a list of valid algorithms. For demonstrative purposes, the list is not complete as it would be unreasonably large.

Next to the *MLTask* class, we have the *DataPreparation* class. This class has different parameters related with operations that can be done in a ML project. Some of these parameters are optional, according to the chosen *MLTask*. For instance, normalization is a must for a *SVM* algorithm, but it is not necessary in a *RandomForest* approach. On the other hand, a *classRebalancing* operation is needed if the dataset is not balanced, despite the chosen algorithm. This will be controlled through OCL constraints [11].

Finally, the different quality aspects of a ML project have been collected under the specialized *MLQualityAspects* class. These capabilities are linked to the different algorithms with a *ContributionType* connection. Thus, some algorithms can make or help to achieve a desirable *MLQualityAspect* (for instance, *DBScan* is an algorithm that ensures *OutlierRobustness*). On the other hand, some algorithms can hurt or even break the achievement of other *MLQualityAspect* (for instance, *KNN* has a "hurt" relationship with *Scalability* since it is affected by a large number of dimensions). Due to space constrains, the detail set of relationships between algorithms and *MLQualityAspects* is not included.

The definition of these constructs and their relationships allows us to define ML requirements models for complex ML projects. Although we have applied our proposal in one such projects based on gas turbines, due to paper constraints we cannot include the result in this paper.

4. Conclusions

Although Machine Learning (ML) and Artificial Intelligence (AI) have gained attention in recent years, a language for specifying ML requirements is still missing. As shown in the related work, only very recently approaches that make use of the widespread i* framework to capture ML requirements have appeared. Unfortunately, the complexity of the ML field together with the high-level abstraction of i* makes the requirements elicitation process a difficult, error-prone task that relies on the designer experience to properly specialize and use i* constructs.

Compared to these approaches, in this paper we propose a novel, baseline i* extension that captures the main concepts involved in ML learning projects and their relationships. In this way, our approach helps designers to build valid ML requirements models. Our approach is complemented by a series of questions that act as guideline, leading to a more systematic requirements capture.

As part of our future work, further empirical studies are under way to evaluate the benefits

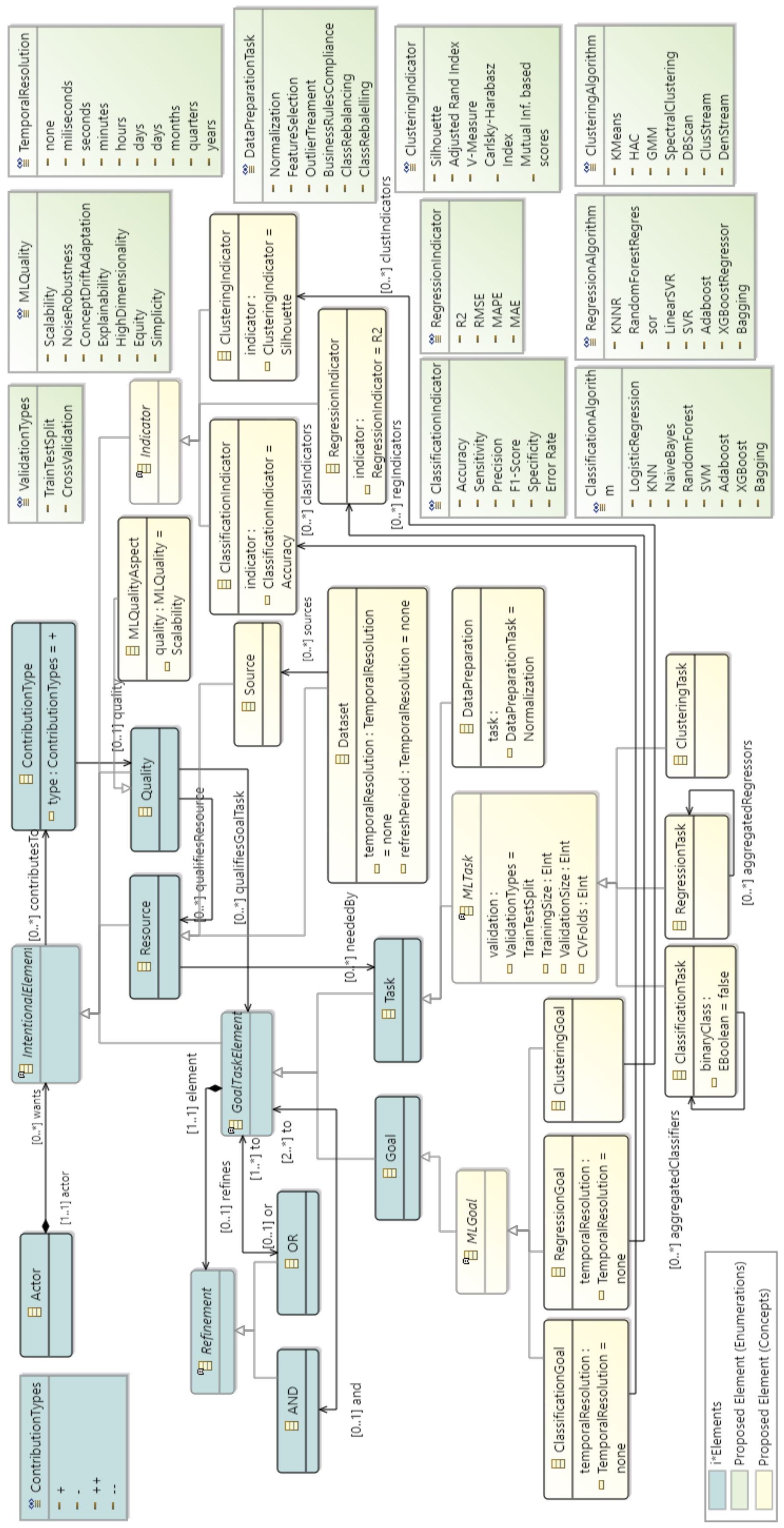

Figure 1: Proposed metamodel: i* for ML.

and limitations of the presented model, especially when applied to other fields. These studies include evaluating the comprehension and validity of the ML design against a task where the proposal is not used. To this end, we are working on the visual notation of the proposal, which will be evaluated together with out methodology using a controlled experiment.

For the planned experiment, two groups of people will develop an ML model to tackle two different problems. First, each group will tackle its problem without using the methodology. Then, the tasks will be exchanged between the two groups and carried out using the proposed metamodel. Both the time taken to solve the task as well as the suitability of the solution will be evaluated. Finally, each group will be asked to fill a questionnaire regarding the interpretability of the model created with the proposed approach and the comprehensibility of the notation.

References

[1] J. Horkoff, Non-functional requirements for machine learning: Challenges and new directions, in: Proceedings of the IEEE International Conference on Requirements Engineering, volume 2019-September, IEEE Computer Society, 2019, pp. 386–391. doi:10.1109/RE.2019.00050.

[2] A. Lo, E. Yu, From business models to service-oriented design: A reference catalog approach, in: Lecture Notes in Computer Science, volume 4801 LNCS, Springer Verlag, 2007, pp. 87–101. doi:10.1007/978-3-540-75563-0_8.

[3] S. Nalchigar, E. Yu, Y. Obeidi, S. Carbajales, J. Green, A. Chan, Solution Patterns for Machine Learning, in: Lecture Notes in Computer Science, volume 11483 LNCS, Springer Verlag, 2019, pp. 627–642. doi:10.1007/978-3-030-21290-2_39.

[4] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley, The ML test score: A rubric for ML production readiness and technical debt reduction, in: Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017, volume 2018-January, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1123–1132. doi:10.1109/BigData.2017.8258038.

[5] J. N. Mazón, J. Trujillo, A model driven modernization approach for automatically deriving multidimensional models in data warehouses, in: Lecture Notes in Computer Science, volume 4801 LNCS, Springer Verlag, 2007, pp. 56–71. doi:10.1007/978-3-540-75563-0_6.

[6] A. Maté, J. Trujillo, X. Franch, A modularization proposal for goal-oriented analysis of data warehouses using I-star, in: Lecture Notes in Computer Science, volume 6998 LNCS, Springer, Berlin, Heidelberg, 2011, pp. 421–428. doi:10.1007/978-3-642-24606-7_32.

[7] E. Gonçalves, J. Araujo, J. Castro, PRISE: A process to support iStar extensions, Journal of Systems and Software 168 (2020) 110649. doi:10.1016/J.JSS.2020.110649.

[8] J. Zenisek, F. Holzinger, M. Affenzeller, Machine learning based concept drift detection for predictive maintenance, Computers and Industrial Engineering 137 (2019) 106031. doi:10.1016/j.cie.2019.106031.

[9] J. Zou, L. Schiebinger, Design AI so that its fair, 2018. doi:10.1038/d41586-018-05707-8.

[10] F. Dalpiaz, X. Franch, J. Horkoff, istar 2.0 language guide, 2016. arXiv:1605.07767.

[11] J. Cabot, M. Gogolla, Object constraint language (OCL): A definitive guide, Lecture Notes in Computer Science 7320 LNCS (2012) 58–90. doi:10.1007/978-3-642-30982-3_3.