Holistic Explainability Requirements for End-to-End Machine Learning in IoT Cloud Systems

My-Linh Nguyen, Thao Phung, Duong-Hai Ly, Hong-Linh Truong
Department of Computer Science, Aalto University, Finland
{linh.m.nguyen, thao.phungduc, duong.ly, linh.truong}@aalto.fi

Abstract—**End-to-end machine learning (ML) in Internet of Things (IoT) Cloud systems consists of multiple processes, covering data, model, and service engineering, and involves multiple stakeholders. Therefore, to be able to explain ML to relevant stakeholders, it is important to identify explainability requirements in a holistic manner. In this paper, we present our methodology to address explainability requirements for end-to-end ML in developing ML services to be deployed within IoT Cloud systems. We identify and classify explainability requirements engineering through (i) involvement of relevant stakeholders, (ii) end-to-end data, model, and service engineering processes, and (iii) multiple explainability aspects. We present our work with a case study of predictive maintenance for Base Transceiver Stations (BTS) in the telco domain.**

I. Introduction

A huge amount of IoT data is collected within IoT Cloud systems for analyzing various domain problems. Besides typical big data analysis pipelines using batch and streaming processing, ML solutions have been increasingly developed and integrated into such IoT Cloud systems for inferencing IoT data [1]. While the development of such ML solutions is challenging, the need to explain the behavior of ML solutions is of paramount importance [2], [3]. Therefore, scoping explainability and identifying explainability requirements must be carried out at the starting point of the ML development to provide insightful information for making decisions on the design and implementation of ML solutions.

Motivating example: We consider the explainability requirements for ML solution development for predictive maintenance within Base Transceiver Stations (BTS). Previous works have discussed big data and IoT data analytics for predictive maintenance in BTS [4]. Leveraging the power of ML, a predictive maintenance company develops ML solutions for predicting behaviors of BTS equipment and infrastructures using alarms and IoT measurement data. For the development, IoT monitoring data within BTS is extracted as input, and possible existing ML models and techniques are collected. Selecting suitable techniques to develop ML models, the core part of an ML service to predict various types of equipment, is also crucial. Before and during the development of ML solutions, it is necessary to have a clear plan for collecting requirements for explainability and their impact on the design and development of the ML solutions. However, there are also multiple relevant stakeholders in predictive maintenance of diverse types of equipment and infrastructures; each stakeholder has different requirements and contribution for different processes, and aspects in the end-to-end ML solution. For example, besides the ML developer, we have the BTS owner (the telco), the predictive maintenance company, and several different third-party providers which manufacture or sell the equipment or provide infrastructure, such as electricity and power backup. To assure the success of the ML development, apart from developing ML models with high performance and accuracy, we must provide explanations based on the explainability requirements for ML services that are collected from these stakeholders. Furthermore, some of these stakeholders need the explanation w.r.t. the aspects for which they are accountable for. To achieve that, the development team faces common but challenging questions of "for whom the explanation is", "the type of explainability they need" and "proactively collecting sufficient explainability requirements about different processes and aspects of the system throughout the ML development cycle", especially when explainability might be hard to achieve with AutoML or pre-defined and black-box models. We can bring valuable inputs from examining stakeholders using scoping explainability to design suitable decision making, during different phases of the ML development, and multiple types of explainability aspects.

Contributions and paper structure: As we discuss in the related work (Section II), the work of explainability requirements so far has not been addressed adequately in an end-to-end manner. Motivated by the complex relationships among stakeholders [5], multiple processes and tasks of the ML design and development, as well as the vast number of possible explainability aspects [6], we investigate a holistic approach to have the best coverage of requirements for explainability. Thus, our work determines and classifies explainability requirements covering multi-stakeholders, multiple aspects in an end-to-end ML system, instead of only focusing on explainability of model outputs. In this paper, by presenting a holistic approach for explainability requirements, we contribute methods to scope and identify (i) relevant stakeholders in the business operation context, (ii) requirements for explainbility for ML processes/tasks, and explainability aspects in end-to-end ML service. We will present our work together with the motivating example to explain our methodology and its methods. The rest of this paper is structured as follow: Section II discusses the related work and their limitations. Section III presents our methods and examples in detail. We discuss tooling and integration in Section IV. We conclude our work and outline our future plan in Section V.

Papers/work	Summary of the papers/work	End- to-end ML	Multi-stakeholders	Multi-aspect explainability	Application domain/focus
L. Chazette and K. Schneider [6]	considers "explainability as non-functional requirement"; focus on the need for explanation from users	No	No	Yes	Software in general
U. Bhatt et al. [5]	designing explainability with "context explanations", "evaluation of explanations", "appropriate design for affected groups", and "stakeholder education"	No	Yes	Yes	ML in general
R. Selvaraju et al. [7]	explainability for the ML model by applying "Gradient-weighted Class Activation Mapping technique"	No	No	No	CNN model-families
A. Galli et al. [8]	focus on the interpretability of the model's prediction; explain a deep learning model's outcome.	No	No	No	Deep learning model in IoT
Y. Xie et al. [9]	explain the predictive models results by identifying failure reasons	No	No	No	IoT
M. Ribeiro et al. [10]	using explanation techniques to understand the classifier prediction based on locally model representatives	No	No	No	ML in general
P. J. Phillips et al. [11]	focus on the links between human factors and explainable decisions	No	No	Yes	ML in general

TABLE I: Explainability requirement approaches

II. Related Work

Explainability in ML is an important subject intensively studied in recent years. There is a wide range of topics that has been discussed, including explainability for multiple stakeholders [5], requirements for explainability [6], [11], and systems for achieving explainability in ML models [10], [7]. Table I summarizes the main related work in the explainability requirements. Overall, the related work emphasizes various aspects in explainability, indicating a wide range of requirements from multiple stakeholders. However, most of them do not focus on end-to-end ML, thus, fail to capture the requirements of a wide range of stakeholders whose input impacts the ML service directly. The work in [12] discussed having domain knowledge to obtain explainability and suggest differentiating between scientific and algorithmic explanations. In our holistic approach, instead, we observe the end-to-end ML through different views so that we can obtain a large number of related explainability requirements.

In terms of supporting explainability associated with ML processes and multiple explainability aspects, Table II shows related work. Most of the work focuses on the training phase for model explainability. From the technical perspective, there have been several works to tackle explainability for ML models. They focus on presenting "explainers" – the tool to provide explanation – mainly for the ML model. Our work is on explainability requirements, thus complementing these "explainers". However, by studying these works, we see that many explainability aspects must be supported. Our approach considers many requirements for aspects from these tools.

III. Holistic Explainability Requirements

Our holistic approach for explainability requirements consists of three main steps: (i) scoping stakeholders in the ML development and operations in an end-to-end manner, (ii) identifying key relevant ML processes and tasks and their associated explainability aspects, and (iii) collecting explainability requirements. We will explain our methodology for the holistic approach in the following.

A. Scoping explainability

Scoping Stakeholders: The final goal of an ML solution integrated into an IoT Cloud system is an *ML service*, which is delivered through several development processes in the so-called "ML pipeline". The development involves many stakeholders who are not only affected by the service but also influence the design and performance of the service. It is crucial that the stakeholders could understand the ML service in use to trust and utilize the output of the service. To achieve that, first, we have to know who the explanation is for, and what are their needs in terms of explainability. If we leave out a relevant stakeholder, we might miss important aspects linked to the ML service. For example, if we do not consider the IoT providers' expertise on the characteristic of the data, we might overlook the causal impact between the specific characteristic dataset and the ML inference accuracy. Secondly, some stakeholders act as explainers to other stakeholders, therefore, identifying these dependencies would help analyze stakeholders' needs. Therefore, defining the scope of stakeholders, to which the explainability covers, should be taken as the first fundamental step.

In our method, we first determine stakeholders into a business-as-usual scope and a corrective scope, as illustrated in Figure 1. The business-as-usual scope includes stakeholders that have a direct dependency w.r.t. to explainability: when stakeholder A uses tools to produce an explanation for stakeholder B directly, A and B have a direct dependency. In corrective scope, we have stakeholders with indirect dependencies, stakeholder C can indirectly influence the explanation for stakeholder D through a chain of stakeholders. Second, for a direct explainability dependency, an explanation can be triggered through an `explain` task which considers various trails from data, software components and other stakeholders. This may require metadata, model experiments, or logging parameters to provide explanation for stakeholder requests. The `explain` task can be provided by existing tools, e.g SHAP [20], SAP [23], GEMS [24]. For an indirect dependency, we will follow a chain of direct dependencies to identify the relevant stakeholders. Given the expected requirements

Aspect/Process	**Data & Model Collection**	**Data Preprocessing**	**Model Development**	**Feature Engineering**	**Training**	**Serving**
Data Summary	Carvalho et al. [13]	-	Bhatt et al. [5]	-	-	-
Data Drift	-	Webb et al. [14]	-	-	-	-
Quality of Data	-	-	Udo et al. [15]	Truong et al. [16]	T.Halvari et al. [17], Udo et al. [15]	-
ML Method Selection	-	-	-	-	Linardatos et al. [18]	-
Model Selection	-	-	Schlegel[15], Xin et al. [19]	-	-	-
Training Configuration	-	-	-	-	Lundberg et al. [20]	-
Feature Attribution	-	-	-	-	Pieter et al. [21]	-
ML Platform	-	-	-	-	Reddi et al. [22]	Reddi et al. [22]

TABLE II: Related researches on explainability aspects in existing ML processes/tasks

established from the explainability dependencies, we collect sufficient data for suitable explainers to be used/developed.

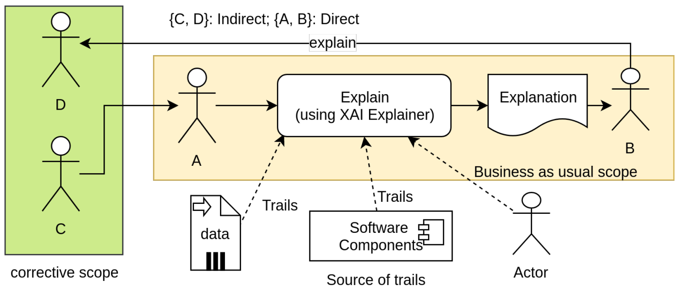


Fig. 1: Stakeholders in direct and indirect dependencies

Scope of explainability-related ML processes/tasks: We need to scope ML development processes/tasks to be considered for the explainability of the service. To meet the stakeholders' needs, the scope of explainability needs to cover all relevant development phases that they are responsible for and interested in. This requires mapping identified stakeholders to development phases, which generally cover requirement elicitation, ML service design, and ML development (data preparation, training, serving). This constitutes the explainability in an end-to-end ML service development. Providing explainability for an end-to-end ML service would preserve the informative connection among stakeholders, related entities (ML models, data, software), and ML processes for producing detailed explanations.

Scoping explainability aspects: Each stakeholder either works directly with specific entities or supervises processes/-tasks applied to these entities. Thus, there is a wide range of entities, and constraints (metrics and aspects) that they are associated with. We could determine these entities and their required explainability aspects by analyzing their role in each process/task, and mapping out requirements, and constraints they need to follow. For example, the developer is responsible for building the prediction model, and care about the accuracy of the model; the dataset used in the model comes from the data provider, who is responsible for the quality of the data; and the requirements for the deployment of the prediction model come from the IoT Cloud service provider, which also needs to ensure the response time constraint of the deployment, or the possibility of security breach influencing the result of the prediction.

Overall scoping approach: After analyzing the scope of explainability, we think explainability for an ML solution should be holistic and consider multiple stakeholders, and multiple aspects for an end-to-end ML service explainability.

Examples with BTS: Figure 2 shows how we apply our scoping methods for the BTS case. In this example, the stakeholders include the BTS owner, the predictive maintenance software company, lead developer, and ML developers. From the development team's perspective, ML developers have a direct dependency with the predictive maintenance company, and have an indirect dependency with the BTS owner. When we map these stakeholders with development phases, we see that BTS owner and the predictive maintenance company are involved with requirement elicitation. The development team works on the ML service design to identify the features and constraints of the service. The ML developers work on the ML service with 2 different approaches (AutoML with HpSklearn [25], manual with LSTM [26]), and take care of the infrastructure and serving in the ML service development.

B. Correlating Requirements in ML Processes and Explainability Aspects

From scoping explainability, we have identified explainability requirements associated with multiple ML processes/tasks and explainability aspects, due to the involvement of multiple stakeholders in an end-to-end ML development. Figure 3 presents key ML processes/tasks and important explainability aspects associated with these processes/tasks.

1) ML processes and tasks: Researchers and practitioners have a common view on key processes/tasks in ML pipelines, shown in Figure 3. These processes/tasks can be implemented manually by humans, automatically by AutoML tools, or by a combination of both.

Data and Models Collection: To start ML solutions, we need to collect data and possible existing models (e.g., from model marketplaces [27]). Here the explainability requirements would be centered on when, where, and why the data and models are collected and the original owners/providers.

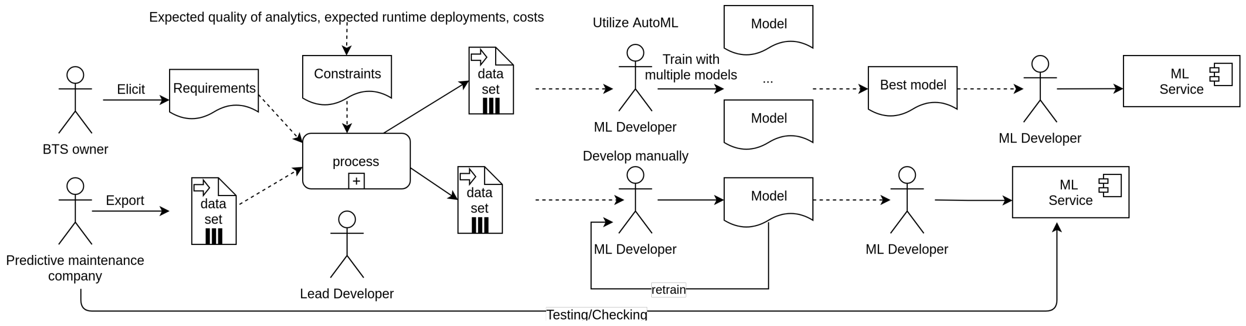


Fig. 2: BTS stakeholders and their interactions in an end-to-end ML service development

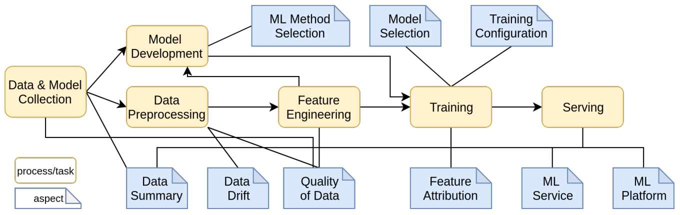


Fig. 3: ML processes/tasks and associated explainability aspects

This allows any explanation to be traced back to the crucial inputs at the beginning of the end-to-end ML system.

Data Preprocessing: Many data preprocessing tasks have to be carried out by humans, especially when we need the domain knowledge for preparing data for ML. For IoT Cloud systems, we focus on explainability requirements associated with the domain knowledge and the data preprocessing methods to be used. These requirements would be best obtained by interviewing the stakeholder who understands the data and the domain. One example with the BTS case is the explainability for the use of model ensembles [28]. The time-series data can be divided into separated slots within a day based on the BTS usage and location, whereas data of each slot would be trained differently to create different models. Another example is the explainability of the method and its impact for handling missing IoT data.

Model Development: We have many ways to develop an ML model. The development process may start by selecting target subjects and prediction types, e.g., the target is electricity whereas the type is the trend of disruption. Then the process may continue with selecting a model type, e.g., a pre-trained model or an existing, fine-tuning model . Thus, "which stakeholder is involved in the model development" and "how the stakeholder develops the model" are crucial questions to provide more explainable aspects. Furthermore, explainability requirements can impact whether we should use black-box or interpretable models [29].

Feature Engineering: As feature engineering creates proper input dataset and model compatibility data features, the explainability requires us to capture feature engineering techniques used for IoT data and the reasons for using these techniques. For example, in our BTS case, existing feature engineering techniques, such as Feature Split, Grouping Operations, and Scaling, can be used, hence, recording these techniques allows us to provide a better explanation for the ML service result.

Training: Many types of explainability requirements are associated with training. The first aspect is to capture who carries out the training, whether this is done manually by a developer or automatically by an AutoML tool [30]. Second, we need to capture the parameters and architectures configured during the training process [19]. This is done by the stakeholder or by recording parameters from the AutoML tool. More detail of the model architectures and parameters selection will allow us to explain the influences of parameters and models on the performance of the results [16].

Serving: We focus on dynamic ML serving for IoT data that implements dynamic inferences, serving predictions on real IoT time data, due to the domain requirements. Thus, explainability requirements are strongly related to the model used for dynamic serving, the ML service platform and its elasticity that hosts the serving model, type of servings the service provides, classification, or regression. For example, in the BTS case, to explain one false prediction which triggers wrong maintenance, one may need to traverse back to the trained model used for serving. Once we determine that model, we can extend the explainability by utilizing explanations of previous corresponding processes.

2) Explainability Aspects: Besides common explainability aspects such as *ML Method Selection* [16] and *Model Selection* [15], due to the nature of IoT data, we focus on several explainability aspects associated with data:

Data Summary: When a ML service fails to predict correctly, the data summary can be used to explain to stakeholders which attributes are not valid, causing ML problems. Furthermore, data summary gives overall statistics that strongly impact on the selection of data preprocessing and appropriate runtime techniques in ML processes and tasks, such as data cleansing or data patterns detection techniques [31]. Thus, we focus on explainability requirements associated with static data summary in `Data Collection` and with runtime data summary during `Serving`.

Data Drift: Many ML models cannot perform well when facing data drift problems [14]. Examples of issues are data inconsistencies, malicious traffic from input sources, or abnormalities occurred at levels IoT devices [32]. Under this explainability aspect, we focus on two issues: data drift in `Data Preprocessing` (which will influence feature engineering and training) and data drift in `Serving` (which will influence the prediction result). First, explainability requirements will focus on whether the extracted data is representative, or the difference between ground truth baseline and target distribution can be observed. It is expected that developers run multiple experiments with samples of data distribution to evaluate the model performance. Thus for explainability we need to capture such different evaluations. For `Serving`, the requirements will focus on methods capturing and managing runtime data drifts for correlating data drift to performance of ML services.

Quality of Data: Many problems of IoT ML solutions can be linked back to quality of data, which depends on different factors such as measurement errors, precision, environmental noise, and discrete observations [1]. Such problems can be explained in different places, such as `Data Collection`, `Data Preprocessing` and `Feature Engineering` tasks. For `Data Collection` and `Training`, requirements will focus on what data properties should be checked, which records are used, where the data is extracted, and the association between quality of data and trained model performance. At `Serving`, the requirements will focus on methods to detect quality of data and cause-effect between quality of data and ML service performance for dynamic inferences, which is the major focus in IoT Cloud systems.

Feature Attribution: For IoT data, domain experts know which data fields (considered for features) play an important role. Furthermore, there are many features extracted from IoT data. Therefore, for feature attribution explanation, requirements will be centered on the list of important features based on IoT data and domain knowledge, the explanation methods selected by the developer (e.g., SHAP values [20] or Integrated Gradients [7]) and the presentation forms of feature impact suitable for the stakeholder. Another type of requirements is related to how the developer and the domain expert design the input baselines used by explainers. For example, in our BTS case, such baselines are missing.

Training Configuration: The cause of poor behavior for a ML model may be related to the parameters of the ML model and its training data [19]. Stakeholders need explanation for ML model behavior and selection via training configurations, which indicate trained models, data and tuning parameters. Typically, different experiments are carried out by the developer to select the optimal parameters and to measure the robustness of the models through different datasets, by also adjusting the baseline configurations. Therefore, our explainability requirements will be focused on training configurations and obtained measurements (robustness, parameter influences [17]) as well as suitable tools for managing experiments.

C. *Explainability requirements elicitation*

Section III-A illustrates how to identify the scope of stakeholders for holistic explainability. However, it is not sufficient to meet their requirements if we cannot identify their specific needs in different ML processes/tasks and explainability aspects mentioned in Section III-B. To achieve that, we recommend the following method, (1) identify concrete individual stakeholders, (2) collect their requirements through survey or interview, (3) continue to update their requirements as the development progresses, e.g. every sprint if the Agile development method is employed.

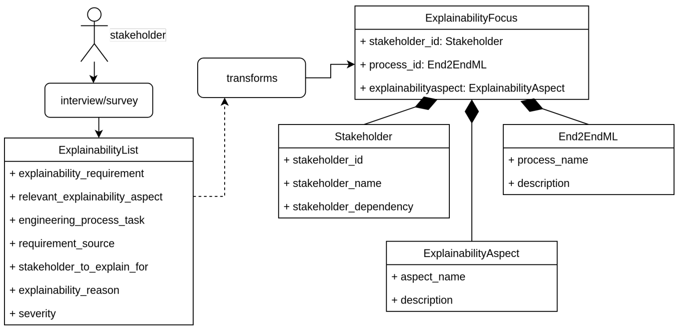
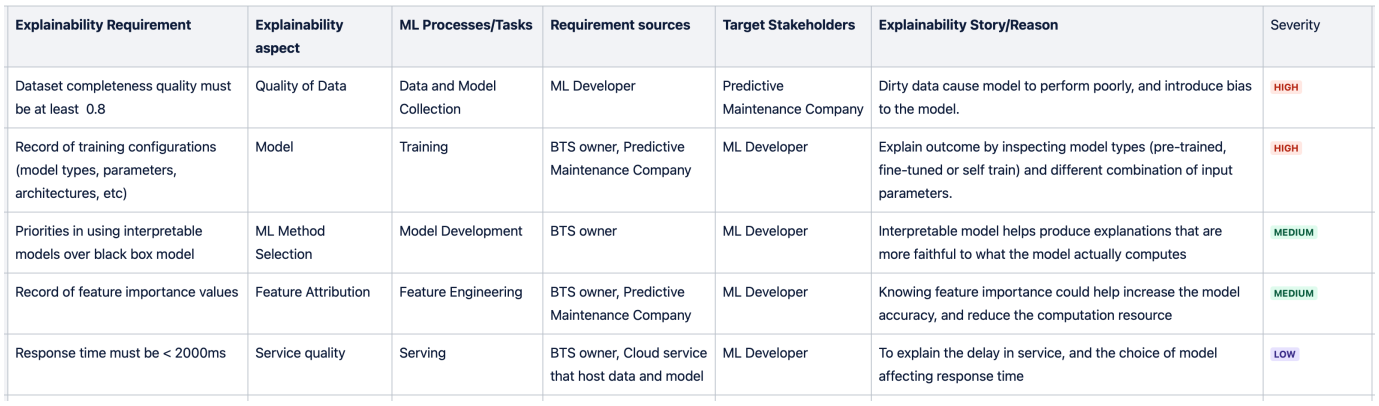


Fig. 4: Steps to collect and document explainability requirements

Figure 4 presents steps to collect and document explainability requirements from stakeholders. To help formulate the survey to collect requirements, we present *an explainability requirement template* with basic information specified in `ExplainabilityList`. This template covers relevant explainability aspects, phase, requirement source, stakeholders, reason, and priority. We use such a template to collect requirements from stakeholders. After the survey has been answered by individual stakeholders, it can be analyzed to extract explainability requirement details. Given the collected requirements, we transform the explainability requirements into explainability focuses. Each explainability focus consists of stakeholders, the related process in end-to-end ML, and the relevant explainability aspects. Explainability focuses could be the concise standard way to document and visualize the explainability requirements, and could be used as a means of communication in explainability solution design and planning.

Examples of BTS Explainability Requirements: Figure 5 illustrates an example of collected explainability requirements. For example, one high priority requirement is about the quality of the dataset from ML developer (requirement source) to the predictive maintenance company (target stakeholders - who are expected to satisfy the requirement). The reason is that the quality of the dataset is important to explain the problem of the model due to data. Another example is associated with response time in the serving process. BTS requires a timely response for the prediction to be useful, the BTS owner needs not only the explanations for the delay in the service but also the explanations for the model design with regard to the trade-off between response time and model accuracy.

After analyzing the survey, processed requirements can be extracted into `ExplainabilityFocus`. `ExplainabilityFocus` can be stored in the JSON,



Explainability Requirement	Explainability aspect	ML Processes/Tasks	Requirement sources	Target Stakeholders	Explainability Story/Reason	Severity
Dataset completeness quality must be at least 0.8	Quality of Data	Data and Model Collection	ML Developer	Predictive Maintenance Company	Dirty data cause model to perform poorly, and introduce bias to the model.	HIGH
Record of training configurations (model types, parameters, architectures, etc)	Model	Training	BTS owner, Predictive Maintenance Company	ML Developer	Explain outcome by inspecting model types (pre-trained, fine-tuned or self train) and different combination of input parameters.	HIGH
Priorities in using interpretable models over black box model	ML Method Selection	Model Development	BTS owner	ML Developer	Interpretable model helps produce explanations that are more faithful to what the model actually computes	MEDIUM
Record of feature importance values	Feature Attribution	Feature Engineering	BTS owner, Predictive Maintenance Company	ML Developer	Knowing feature importance could help increase the model accuracy, and reduce the computation resource	MEDIUM
Response time must be < 2000ms	Service quality	Serving	BTS owner, Cloud service that host data and model	ML Developer	To explain the delay in service, and the choice of model affecting response time	LOW

Fig. 5: Example of requirements

or CSV format for later usage. This example presents how we can effectively find the explainability focus by involving multi-stakeholders in collecting the requirements. The range of processes and aspects shows a holistic explainability is essential to capture the requirements from multi-stakeholders. At the same time, we could capture the dependencies between stakeholders, and their roles for the explainability through this method, which enables better planning to collect explainability requirements.

IV. Discussion on Tooling and Integration

For our holistic approach for explainability requirements, we must also select and provide suitable tools for collecting and maintaining such requirements and data for appropriate explainers to provide a useful explanation. We identify the following directions that we need to address:

Integrate explainability requirements into cloud-native DevOps: Given that today's ML solutions are cloud-native and ML engineering is based on DevOps [33], we need to integrate the explainability requirement workflows into cloud-based tools to manage their requirements to avoid duplicate or missing requirements. To the best of our knowledge, there have not been any tools concentrating solely on ML explainability requirements collection. However, there are several tools available to capture common requirements from stakeholders in software development, such as Accompa [34], Visure [35], and MindManager [36]. Existing cloud-based, continuous requirements management tools could be integrated into the requirements identification in Section III-C.

Provide techniques and services for managing diverse types of trails for explainability: for the holistic approach, we must collect trails from different phases of end-to-end ML including data and model collection, experiments, quality of analytics, and requirements to provide explanations to stakeholders. There are some tools available for supporting monitoring data quality tools such as Great Expectations [37], Databand [38], and Dataform [39]. Additionally, MLFlow [40] and Data Version Control (DVC) [41] also support data collection and management for model experiments, datasets, and parameters. However, these frameworks do not support a holistic explainability approach by keeping track of multi-aspect requirements discussed in Section III concerning end-to-end ML. This proposes future research about framework/-tool integrated with the centralized management platform mentioned above to capture various explainability aspects for data and ML pipelines, as well as keep track of documentation for requirements when validating data quality tests, metadata experiments during model development, or explaining data issues to relevant stakeholders in term of datasets, ML models, services, experiments.

Identify, recommend and configure explanation tools: Given collected requirements from different views of stakeholders, multiple explainability aspects (related to data, models, and services), and ML processes, we can identify and recommend suitable explanation tools. Such tools have been increasingly developed but it is difficult to know which tools are suitable for which types of requirements. Furthermore, the combination of different explanation tools, e.g., tools working with model internals and tools with benchmarks/blackbox tests methods, and newly developed explanation tools can address the multi-aspect explainability as well as can support the explainability across phases in end-to-end ML.

V. Conclusions and Future Work

Explainability is complex and being able to identify the scope to collect adequate requirements for explainability is crucial. A scope of explainability is dependent on specific ML solutions but it must be carried out from a holistic view. Our focus is on requirements for explainability in developing ML in IoT Cloud systems and we have presented a methodology with three distinguishable aspects for explainability requirements: multiple stakeholders, end-to-end ML, and multi-aspect explainability. Using our methods, the team developing ML solutions could gather a rich set of requirements, steering the ML design and development with the right solutions for explainability.

Our future work is to complete templates and steps for requirement elicitation. Furthermore, we are working on data collection techniques and services for analyzing requirements and relevant data for developing context-aware explainability.

References

[1] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[2] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.

[3] A. Barredo Arrieta, N. Daz-Rodrguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[4] H. Truong, "Integrated analytics for iiot predictive maintenance using iot big data cloud systems," in *IEEE International Conference on Industrial Internet, ICII 2018, Seattle, WA, USA, October 21-23, 2018*. IEEE, 2018, pp. 109–118.

[5] U. Bhatt, M. Andrus, A. Weller, and A. Xiang, "Machine learning explainability for external stakeholders," Jul 2020.

[6] L. Chazette and K. Schneider, "Explainability as a non-functional requirement: challenges and recommendations," *Requirements Engineering*, vol. 25, no. 4, p. 493514, 2020.

[7] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[8] A. Galli, V. Moscato, G. Sperl, and A. D. Santo, "An explainable artificial intelligence methodology for hard disk fault prediction," *Lecture Notes in Computer Science Database and Expert Systems Applications*, p. 403413, 2020.

[9] Y. Xie, D. Feng, F. Wang, X. Tang, J. Han, and X. Zhang, "Dfpe: Explaining predictive models for disk failure prediction," *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*, 2019.

[10] M. Ribeiro, S. Singh, and C. Guestrin, "why should i trust you?: Explaining the predictions of any classifier," *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2016.

[11] P. J. Phillips, C. A. Hahn, P. C. Fontana, D. A. Broniatowski, and M. A. Przybocki, "Four principles of explainable artificial intelligence," 2020.

[12] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," *IEEE Access*, vol. 8, pp. 42 200–42 216, 2020.

[13] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, p. 832, 2019.

[14] G. I. Webb, L. K. Lee, F. Petitjean, and B. Goethals, "Understanding concept drift," Apr 2017.

[15] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim, "Towards a rigorous evaluation of xai methods on time series," 2019.

[16] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of automl approaches and tools," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, pp. 1471–1479.

[17] T. Halvari, J. K. Nurminen, and T. Mikkonen, "Testing the robustness of automl systems," *Electronic Proceedings in Theoretical Computer Science*, vol. 319, p. 103116, Jul 2020.

[18] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, 2021.

[19] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 47684777.

[21] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren, "An open source automl benchmark," 2019.

[22] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "Mlperf inference benchmark," pp. 446–459, 2020.

[23] S. Sean, J. Wang, W. T. Chai, P. Ni, A. Raj, and M. Karthik, "Contextual ai." [Online]. Available: https://github.com/SAP/contextual-ai

[24] "Gems ai." [Online]. Available: https://www.gems-ai.com/

[25] M.-A. Zller and M. F. Huber, "Benchmark and survey of automated machine learning frameworks," Jan 2021.

[26] W. Zhang, W. Guo, X. Liu, Y. Liu, J. Zhou, B. Li, Q. Lu, and S. Yang, "Lstm-based analysis of industrial iot equipment," *IEEE Access*, vol. 6, pp. 23 551–23 560, 2018.

[27] "Deep open catalog," https://marketplace.deep-hybrid-datacloud.eu/.

[28] T. Tornede, A. Tornede, M. Wever, F. Mohr, and E. Hüllermeier, "Automl for predictive maintenance: One tool to rul them all," in *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*, J. Gama, S. Pashami, A. Bifet, M. Sayed-Mouchawe, H. Fröning, F. Pernkopf, G. Schiele, and M. Blott, Eds. Cham: Springer International Publishing, 2020, pp. 106–118.

[29] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, May 2019.

[30] Z. Weng, "From conventional machine learning to automl," *Journal of Physics: Conference Series*, vol. 1207, p. 012015, 04 2019.

[31] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, pp. 85–126, 2004.

[32] T. S. Sethi and M. Kantardzic, "Handling adversarial concept drift in streaming data," *Expert Systems with Applications*, vol. 97, pp. 18–40, 2018.

[33] R. Bianchini, M. Fontoura, E. Cortez, A. Bonde, A. Muzio, A.-M. Constantin, T. Moscibroda, G. Magalhaes, G. Bablani, and M. Russinovich, "Toward ml-centric cloud platforms," *Commun. ACM*, vol. 63, no. 2, p. 5059, Jan. 2020.

[34] "Accompa." [Online]. Available: https://web.accompa.com/

[35] "Visure." [Online]. Available: https://visuresolutions.com/

[36] "Mindmanager." [Online]. Available: https://www.mindmanager.com/en/

[37] C. James, G. Abe, M. Eugene, L. Rob, and M. Taylor, "Great-expectations," *GitHub repository*, 2021.

[38] "Databand." [Online]. Available: https://databand.ai/

[39] "Dataform." [Online]. Available: https://dataform.co/

[40] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, and C. Zumar, "Accelerating the machine learning lifecycle with mlflow," *IEEE Data Eng. Bull.*, vol. 41, pp. 39–45, 2018.

[41] A. Barrak, E. E. Eghan, and B. Adams, "On the co-evolution of ml pipelines and source code - empirical study of dvc projects," pp. 422–433, 2021.