A Goal-Oriented Approach for Modeling Decisions in ML Processes

1st Rohith Sothilingam
Faculty of Information
University of Toronto
Toronto, Canada
rohith.sothilingam@mail.utoronto.ca

2nd Eric Yu
Faculty of Information
University of Toronto
Toronto, Canada
eric.yu@utoronto.ca

***Abstract*—The ability to judiciously make systematic decisions throughout the entire Machine Learning (ML) lifecycle, encompassing model training, development, and deployment, is crucial for ensuring the robustness and adaptability of models in the face of evolving requirements. This paper proposes a goal-oriented (GO) approach which aims to improve the ML process so as to achieve overall ML goals. Our approach aims to understand the alternative types of decisions or choices that are made, and where in the ML process, based on what criteria. The main contribution of this research paper is to propose a GO approach, to support the strategic selection of technical alternatives, throughout the phases of model training, development, and deployment, to design ML solutions.**

***Index Terms*—Responsible AI, Machine Learning, Goal Modeling, Repetition Loops, Sensors, Actuators**

I. Introduction

Many decisions are required throughout the ML development process. Together, they contribute towards achieving the overall goals, including technical goals such as accuracy, as well as social responsibility goals such as fairness, and trustworthiness. The ML process is iterative, with some iterative cycles nested inside others. As decision criteria may need to change to reflect changing conditions, it is important to position decisions appropriately within the ML process in order to achieve overall goals.

Goal-oriented (GO) conceptual modeling has long been established in the Requirements Engineering literature as an approach to deal with such conflicts and tradeoffs in other areas of software engineering (e.g. [3] [13]). Several GO approaches exist which demonstrate the usefulness of GO reasoning to design, elicit, and analyze Business requirements (e.g. GRL [1] [2]). GO reasoning has also been shown to be useful for systematically designing and analyzing the interrelationships between business and ML objectives. For example, GR4ML [5] analyzes the strategic business aspects of data analytics solutions.

GO conceptual modeling can provide a structured reasoning approach for systematic decision-making at each iteration of the ML process. The ML process relies on iterative experimentation, with multiple layers of nested iterative loops. An important consideration is to position the right decisions in the right places in the ML process. Our approach supports the design of the ML process, to guide the right decisions to be made. These decisions can be achieved by several alternatives, which are represented as tasks, based on i* [13]. To choose among the alternatives, decision criteria are represented using softgoals. An example of a decision in ML may be "Hyperparameters be chosen". Alternatives available may be various types of hyperparameters to choose from, depending on the ML model type chosen such as K-Nearest Neighbours (KNN), Support Vector Machines (SVM), or Decision Trees. The decision points must be at the appropriate places.

Much like a process designer aiming to design the SDLC process in software development, we aim to determine where and when in the ML process that decisions must be made. For example, choosing the ML model type at the wrong place in the ML process can cause issues related to cost inefficiency or inaccuracy of prediction results. To make the most appropriate decision, it is crucial to consider the decision criteria associated with each alternative because each alternative will yield different, often competing benefits and pitfalls.

During the ML lifecycle, cycles occur in nested patterns, with various nested "loops" of iteration. In each iteration round, decision criteria may need to change (or remain the same). Examples include reconsidering how we evaluate our ML models, the techniques we use, and the metrics we care about, at each repetition cycle. By leveraging GO reasoning, the proposed approach intends to facilitate systematic reasoning about what decisions are made at every repetition are purposeful and aligned with the overarching goals of ML model development.

We propose the following novel GO modeling concepts to support systematic decision-making for designing ML solutions: Sensors, Actuators, and Repetition Loops. Sensors provide information needed to support making informed decisions, such as choosing the correct type of ML model. Sensors complement decision criteria (softgoals). Actuators represent adjustable knobs, to convey settings that can be adjusted, based on what the Sensors tell us. Repetition Loops represent iterative cycles, which are often nested within larger (outer) loops, such as ongoing cycles of development in the ML lifecycle. Together, Repetition Loops involve a constant process of tweaking and improving upon decisions, with the support of Sensors and Actuators.

Sensors and Actuators are not explicitly expressed in any

known goal-oriented or process modeling language. The proposed modeling constructs build upon existing goal-oriented modeling approaches such as i* [13] and the NFR framework [3].

II. Motivating Example

Feature engineering is the process of selecting and transforming raw data into relevant features that improve machine learning model performance. Feature engineering is essential to extract meaningful insights from data and build effective ML models. During this stage, how can we make adjustments while making decisions on chosen features to achieve the desired feature importance score? What further, non-intentional factors, can we consider as a decision input to aid us in our design? BIM Indicators can support the ability to measure against a given threshold, to support analysis of business objectives using current business data as business metrics [4]. However, it does not support the collection of data, as an input for tuning particular settings or parameters to conduct adjustments which would lead to goal satisfaction.

Due to the limitation of existing goal modeling techniques, we cannot answer these questions as existing goal modeling notations do not allow such information to be expressed, such as the decisions involved in the feature engineering process. To inform and guide these decisions as well as answer these questions, we propose modeling constructs that support collecting information from the real (or causal) world to help us improve our design of ML processes concerning appropriate decisions at each iterative loop. By using the modeling constructs we propose, we can now uncover missing elements, with respect to how we consider reasoning for goals.

III. Introducing Modeling Notation with a Toy Example

Consider the task of baking cookies. While baking the cookies, some important things to consider are the following: temperature and taste. We will walk through the cookie baking process as an example setting for applying the proposed modeling constructs. Figure 3 conveys a goal model consisting of the proposed modeling notation.

A. Sensors

At a high level, sensors act as the ”eyes” and ”ears”, as sensors represent the collection of data from the environment (or causal world). Sensors are conveyed using Indicators, as borrowed and adapted from BIM [4]. Sensors intend to gather data as inputs to the intentional actor from the causal world, of which these inputs will then be used as decisioncriteria for being used at the operationalization stage of the goal model. Quantities (i.e. thresholds) are converted into signals that can be interpreted by the intentional actor and used for reasoning based on related intentions and priorities, which are conveyed as softgoals. Once the threshold is satisfied, an adjustment is needed (represented as an Actuator).

In our baking example, a thermometer is typically used for baking to measure the actual temperature inside the oven accurately. Oven dials may not always reflect the true temperature, Heat and taste are the Sensors in this example to signify whether the cookie is baking at the correct temperature.

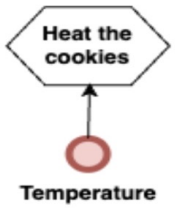

Fig. 1. Adjusting the temperature to achieve the right heat using an Actuator

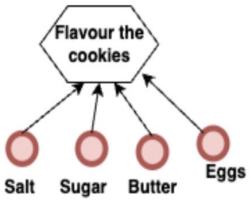

Fig. 2. Actuator: adjust flavour of the cookies

B. Actuators

Actuators complement Sensors, to facilitate the ability to adapt to environmental settings. Actuators refer to the adjustable settings that influence a specific task, such as influencing how well that task works. These settings are set before the task is set forth. The decision criteria for adjusting these settings are done based on the input from Sensory information from the causal world (i.e. a Sensor).

The Actuator can be attached to a Task to signify parameters that can be tuned to successfully achieve that task, based on the input gathered from Sensors. Often, a Task can be associated with multiple Actuators, signifying how a task can be tuning multiple control knobs.

Using the same baking example above, temperature is conceptualized as an Actuator (figure 1), signifying that the temperature can be adjusted until the cookie is baking at the correct temperature. For adjusting the taste, the following are conceptualized as Actuators: sugar, salt, butter, and eggs, to adjust the taste of the cookie (figure 2).

C. Repetition Loops

Repetition Loops have long been used as software programming constructs. Repetition Loops continue until a condition (or “stop criteria”) is specified. We use the “repeat loop” sign borrowed from BPMN to represent a Repetition Loop, which can be attached to a Goal, signifying that the task is repeated until a specified condition has been met.

Nested loops are represented through task refinement. The Repetition Loops associated with the refined Tasks are referred to as the Inner Loop, whereas the Repetition Loop associated with the parent Task is referred to as the Outer Loop.

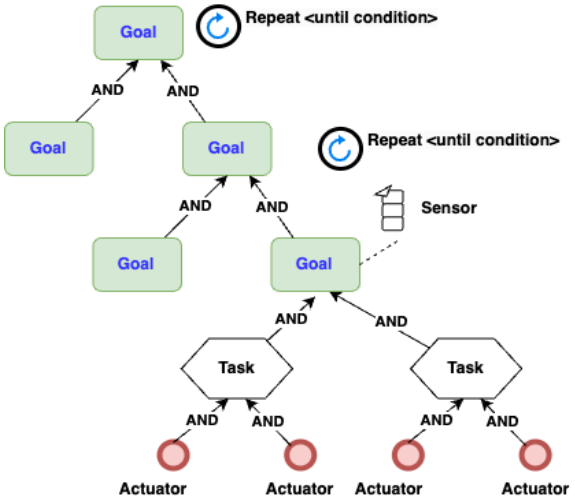

Fig. 3. Proposed modeling notation conveying sensors, actuators, and repetition loops

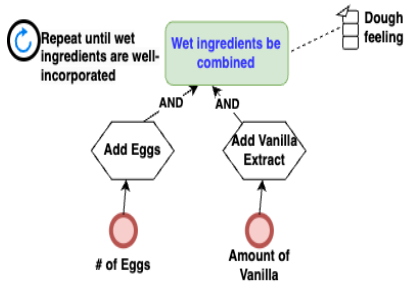

Fig. 4. Repetition Loop with baking

Continuing with the baking example, the outer loop represents the different batches of cookies. Within each batch, there are a series of inner loops, representing the different stages and steps for baking the cookies. Within each of these inner loops, there are further inner loops describing the tasks for the set of 5 main inner loops. This set of enveloped outer and inner loops are referred to as nested loops.

Each time you proceed through these nested loops, a systematic set of cookies is being created with set flavour, consistency, and texture. The result is a structured and organized way of managing the complexity of cookie baking.

In figure 4, the "Wet ingredients be combined" Goal represents an inner loop that is repeated until the stopping criteria of "wet ingredients are well-incorporated" is satisfied. The Sensor "dough feeling" provides an input as the Actuators " of eggs" and "amount of vanilla" are tuned until the stopping criteria of the repetition loop is satisfied.

In figure 5, we represent a metamodel that captures the semantics and formal relationships of the primary modeling constructs proposed: sensors, actuators, and repetition loops.

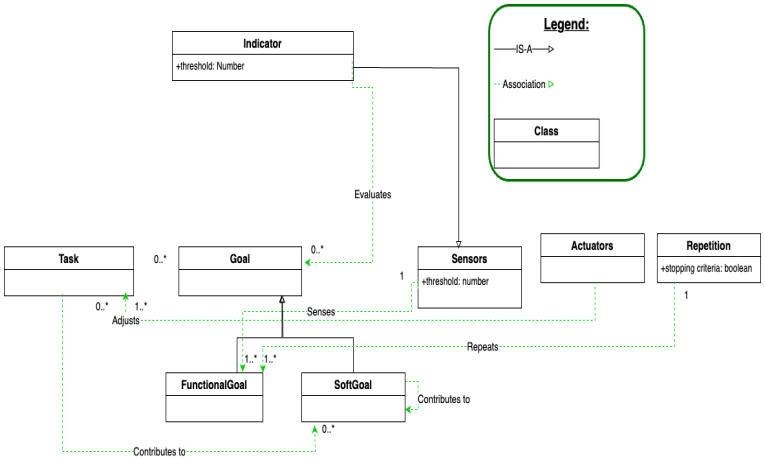

Fig. 5. Metamodel for Sensors, Actuators, and Repetition Loops

IV. Applying the Modeling Components to ML

Hyperparameters will need to be tuned at multiple points of time in the ML lifecycle, for various reasons. The reasoning is strategic as tuning specific hyperparameters will yield results that benefit different, often conflicting softgoals. We will explore the ability of our proposed modeling constructs to express and analyze the criteria for why a change (e.g. hyperparameter tuning) needs to occur, and what information substantiates this decision. We will explore how well our proposed modeling constructs can be used to express and reason the conflicts between technical ML and strategic decisions.

A. Sensors and Actuators in ML

Consider the following example in figure 6. The Sensors of "Feature Importance" and "Feature distribution statistics" are attached to the Goal of "Feature Evaluation" and are used to gather input on decision criteria for which task alternatives to choose (i.e. the feature evaluation technique alternatives. Feature distribution statistics provide summary statistics for each feature, while Feature Importance scores can be used as input for ranking features and initializing permutation importance calculations. The modeler can design actions to take based on available alternatives (tasks) to make informed decisions (goals) in the goal model below (figure 6). Softgoal decision criteria associated with each technique alternative, are represented as tasks ("permutation importance" and "recursive feature elimination"). The Sensors provide non-intentional inputs to tune the tasks of Permutation Importance and Recursive Feature Elimination, based on their associated Actuators. Tuning Step Size controls how many features are eliminated at each iteration. Tuning the number of permutations controls the accuracy and stability of the importance scores. More permutations lead to more robust results but increase computation time.

The alternatives which are chosen here are then used at the following operationalization stage, where the chosen alternatives are used. This is denoted by using Indicators, as borrowed from BIM [4].

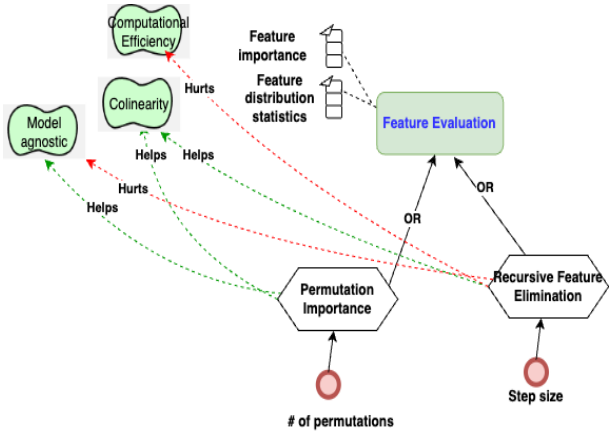

Fig. 6. Using Sensors as inputs to support the analysis of alternatives during decisions in the ML process

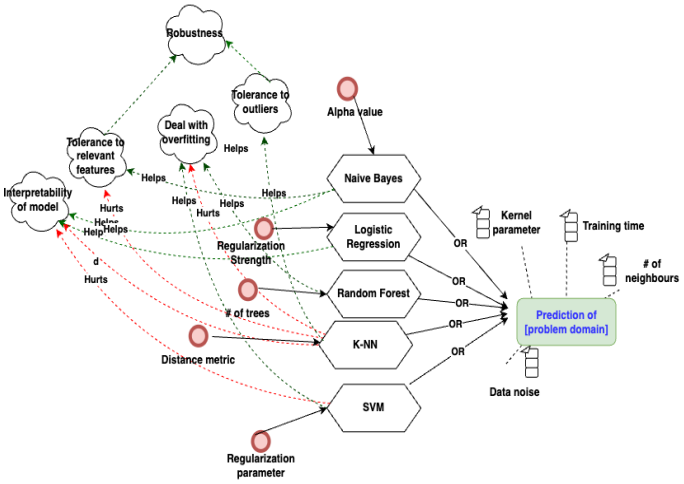

Fig. 7. Using Sensors as an input to deal with tradeoffs in ML

As another example, in the goal model fragment below (figure 7, the following Sensors are attached to the goal of "Prediction of problem domain": "kernel parameter", " of features to consider", "distance metric", and "learning rate". Each of these Sensors is used as input to tune associated hyperparameters, which are represented as Actuators. The kernel parameter controls the shape of the decision boundary in SVM, and the regularization parameter is used to tune the performance of SVM models. The "Distance metric" Actuator can be tuned in KNN models based on the number of neighbours (K). The "Number of trees" (or iterations) Actuator can be tuned, based on training time. If training time is a concern, one may want to reduce the number of trees to speed up the process. Regularization Strength for Logistic Regression and Alpha Value for Naive Bayes models can be tuned based on the data noise Sensor. If the data is noisy or has outliers, increasing regularization can help reduce their impact. Based on these inputs, the ML model alternatives can be reasoned between, to choose the appropriate alternative based on these Sensors.

B. Repetition Loops in ML lifecycle

Consider the following example below in figure 8, which involves an outer loop (goal of Deployment of ML model) and an inner loop (goal of Predictive model be trained). The repetition is complete once all model evaluation criteria have been met. Repetition loops illustrate how algorithms iterate through repetitive decision-making loops multiple times to learn patterns and achieve both ML performance and strategic goals. A set of Indicators is attached to the goal of "Prediction performance be monitored" with thresholds to monitor the success of the goal.

Building on the example earlier, in figure 9, we demonstrate an important use of the repetition loop modeling construct for conveying the tuning hyperparameters as part of a larger (outer) goal of training a predictive ML model. Alternative configurations are tested iteratively to find the optimal settings.

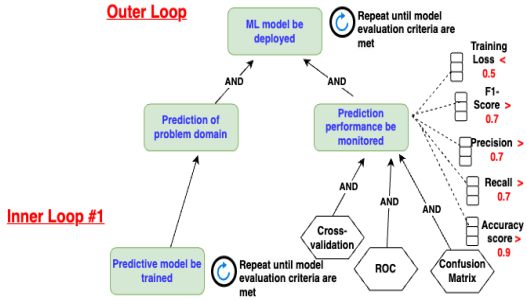

Fig. 8. Iterative Repetition Loops in ML

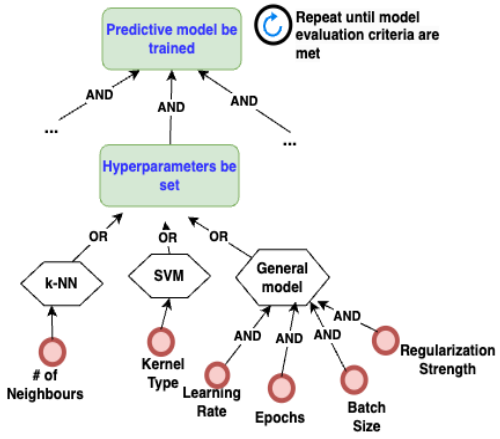

Fig. 9. Repetition loops and hyperparameter tuning for achieving successful ML model convergence

Many nested loops occur in the ML lifecycle. Using task refinement, we can see how complex goal realization of loop criteria is achieved involving multiple, nested, interrelated sub-steps. Nested Loops demonstrate how different stages (at different times) of the ML lifecycle continuously improve or are fine-tuned (outer and inner loop) to better meet the defined goals (i.e. the stop criteria of each Repetition Loop element).

The Repetition loop "repeat until model evaluation criteria are met" is attached to the Goal "Predictive model be trained". This indicates the Goal in question is repeated until the "stop criteria" of "evaluation criteria are met" is completed. Each time a hyperparameter is tuned (will happen 3 times to cover each actuator), the "train model" task will be conducted as well as part of the AND relationships with the Goal "Predictive model be trained" of which the Repetition Loop is associated with.

V. Related Work

Sensors and Actuators are not explicitly supported in conceptual modeling language notations. In broader conceptual modeling notations like UML or BPMN, the explicit representation of sensors and actuators is typically abstracted away in favour of focusing on high-level processes, interactions, and structures.

Repetition loops are supported in process modeling languages such as BPMN and UML. BPMN supports the ability to attach the condition on which a loop task executes for the first time or apply the condition on repeated executions as an annotation to the task. A task can also be repeated iteratively for multiple instances until the stop criteria are met. BPMN is limited in that beliefs and assumptions cannot be captured as well as refined tasks that are interrelated with tasks associated with the repetition loop. Though nested sub-processes are expressible consisting of tasks with their repetition loops, nested refinement of tasks, concerning sub-processes is not supported in BPMN.

Process modeling languages do not support the criteria in which decisions are made, and at different temporal loops. For example, how does one decide whether something should be decided within an inner loop rather than the outer loop? Are there certain decisions that need to be made in the inner loop as opposed to the outer loop?

Goal-oriented modeling languages have limited support for sensors, actuators, and nested repetition loops. Awareness requirements and adaptive systems in goal modeling touch on limited aspects of *sensing*. Morandini et al. [6] look at a goal-oriented approach for designing self-adaptive systems, drawing particular attention to the engineering of self-adaptive software.

Awareness Requirements [12] are requirements that refer to other requirements or domain assumptions and their success or failure at runtime. This type of reasoning encourages adaptivity to support the monitoring, diagnosis, planning, and execution of requirements. Our proposed Sensor modeling construct allows intentional Actors to decide what to do based on the value of the sensed variable, allowing the Actor to interact with the non-intentional world, in addition to the intentional relationships it has with other actors.

VI. Conclusion

In this work, we presented 3 novel modeling constructs for GO reasoning to support the design of ML processes: sensors, actuators, and repetition loops. This work is part of larger PhD research [7] [8] [9] [10], which aims to understand how these three parts can be used together to design ML models that are technically sound, responsible, and adaptable. In future work, we aim to extend the proposed GO approach to incorporate GO reasoning about the placement of decision points, to determine whether the decision points are in the right place as applied to ML processes. We will apply, validate, and refine the proposed GO approach to existing case studies identified in existing literature, such as [11]. We aim to then utilize the modeling constructs in empirical case studies.

References

[1] Amyot, D. (2003). Introduction to the user requirements notation: learning by example. Computer Networks, 42(3), 285-301.

[2] J. Castro, M. Kolp and J. Mylopoulos, "A requirements-driven development methodology", Advanced Information Systems Engineering: 13th International Conference CAiSE 2001 Interlaken, pp. 108-123, 2001.

[3] Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J. (2012). Non-functional requirements in software engineering (Vol. 5). Springer Science Business Media.

[4] Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J. (2014). Strategic business modeling: representation and reasoning. Software Systems Modeling, 13, 1015-1041.

[5] S. Nalchigar and E. Yu, "Designing business analytics solutions", Business Information Systems Engineering, vol. 62, no. 1, pp. 61-75, 2020.

[6] Morandini, M., Penserini, L., Perini, A. (2008). Towards goal-oriented development of self-adaptive systems. In Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems (pp. 9-16).

[7] Sothilingam, R., Eric, S. K. (2020). Modeling Agents, Roles, and Positions in Machine Learning Project Organizations. In iStar (pp. 61-66).

[8] Sothilingam, R., Pant, V., Eric, S. K. (2022). Using i* to Analyze Collaboration Challenges in MLOps Project Teams. In iStar (pp. 1-6).

[9] Sothilingam, R. (2023). A Requirements-Driven Conceptual Modeling Framework for Responsible AI. In 2023 IEEE 31st International Requirements Engineering Conference (RE) (pp. 391-395). IEEE.

[10] Sothilingam, R., Yu, E. (2023). Toward a Goal-Oriented Argumentation Approach for Fair ML Measures Using i*.

[11] Shankar, S., Garcia, R., Hellerstein, J. M., Parameswaran, A. G. (2023). "We have no idea how models will behave in production until production": How engineers operationalize machine learning.

[12] Silva Souza, V. E., Lapouchnian, A., Robinson, W. N., Mylopoulos, J. (2011). Awareness requirements for adaptive systems. In Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems (pp. 60-69).

[13] E. Yu, P. Giorgini, N. Maiden and J. Mylopoulos, Social modeling for requirements engineering, MIT press, 2011.