Palantir

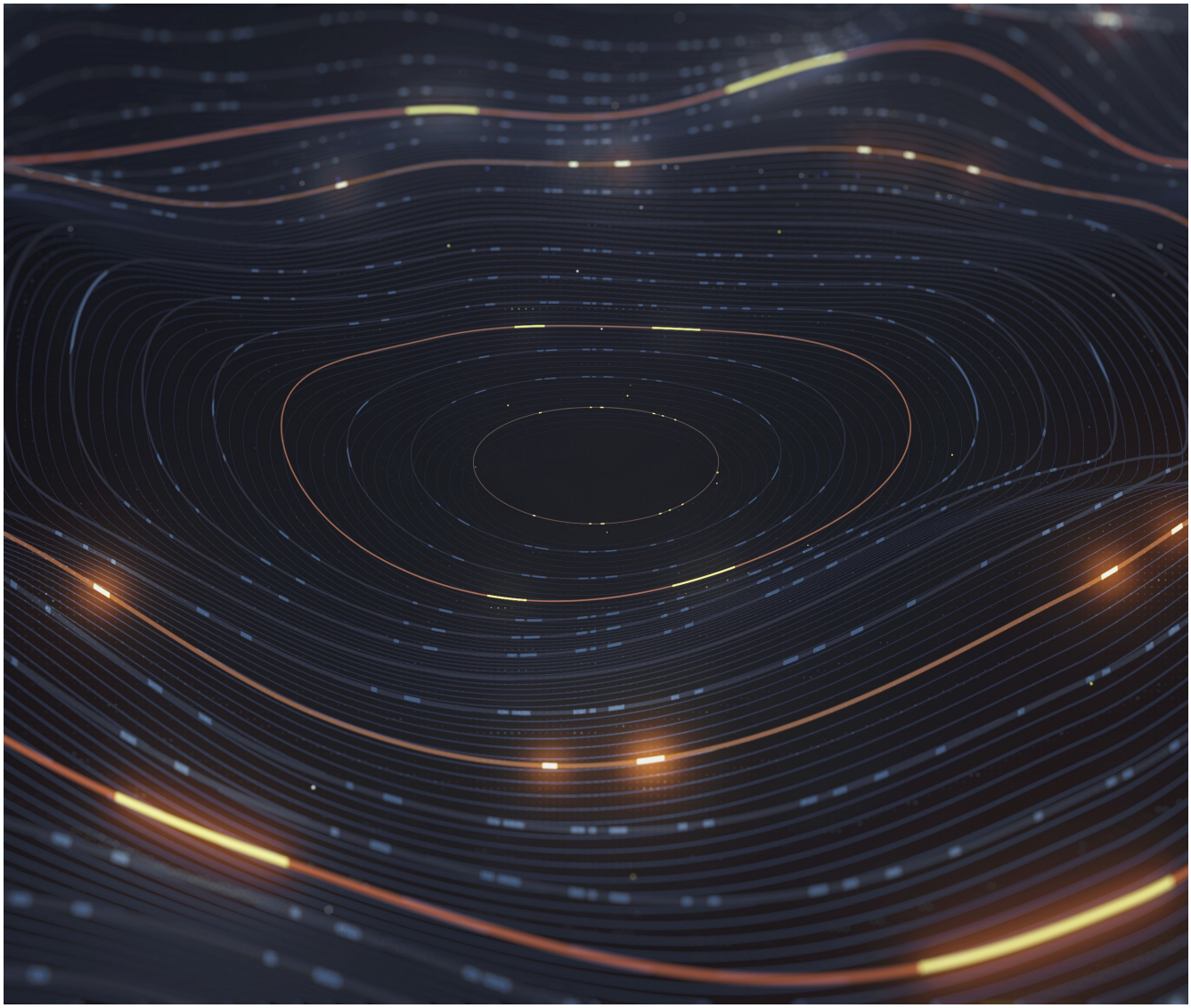
AI on RAILs

A Responsible AI Lifecycle Framework

palantir.com

Copyright © 2023
Palantir Technologies Inc.

All Rights Reserved



Introduction

A growing number of commercial and government organizations are adopting Artificial Intelligence (AI) and machine learning (ML) approaches to solve some of their most consequential problems. Despite the rapid proliferation of advanced AI techniques, architectures and benchmarks, the challenges organizations face are often broader than building or using the next most performant model. Important considerations exist, for instance, surrounding the fairness and bias in AI models and underlying training data, the governance of the appropriate use of AI systems, and the provision of training to equip employees with a necessary understanding of the impact of AI-based automation.

To address these challenges, many institutions have called for "Responsible AI" (RAI) in policy documents and statements of ethical principles. Though these principles are a step in the right direction, there is a need to further translate them into guidance that engineering teams can implement.

In fact, the engineering teams who build software platforms for AI systems have a crucial role in helping with adoption of RAI practices. Many of the techniques that engineers use for building software reliably and iteratively – version control, continuous integration, continuous delivery, agile development – should also be applied to the development and deployment of AI.

In this whitepaper we describe a novel model lifecycle framework built upon common software engineering techniques that enables engineering teams to implement RAI principles in practice.

This whitepaper proceeds by presenting:

i. Responsible AI Themes
ii. Model Lifecycles: The Emergence of MLOps
iii. RAI Roles
iv. Responsible AI Lifecycle (RAIL) Framework
v. Why does RAIL matter?
vi. Looking Ahead: Potential extensions of RAIL

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

Responsible AI Themes

Though the term “Responsible AI” (RAI) is increasingly entering the AI/ML discourse, particularly as many organizations publish statements about “Responsible” and “Ethical AI”, RAI is not a monolithic principle. Instead, when describing RAI, each organization may consider different principles that vary across institutional, cultural, and technical contexts. These RAI principles are helpful when aligning stakeholders towards a common goal in an organization’s AI strategy, providing an approachable vocabulary that can guide important conversations about how AI is used or whether it should be used at all.

We reviewed common principles across the technology industry and governmental organizations and have presented a synthesis of those most salient for engineering teams working on AI systems. Due to their abstract nature, purportedly distinct formulations of these principles bear likenesses to each other. We therefore adopted a "family resemblance" approach, grouping corresponding principles into key themes.

In total, we illustrate **eight key themes** of RAI relevant for engineering teams building AI systems. Below, each theme is presented with the overlapping terms identified, an explanation, and an example of how the theme may be operationalized. The themes presented are by no means exhaustive, but instead represent a sample of some of the commonly identified AI principles.

“AI System”: We define an AI system as the end-to-end system within which a model is embedded. The model itself can be based on a variety of mathematical techniques – from ML models to statistical models to rule-based approaches for modeling business logic. Considering RAI at the AI system level provides a wider context that includes the interfaces through which end-users consume model outputs, the data and processing pipelines that feed inputs to the model, and other key workflows apart from iterating on the model logic.

EQUITABLE (FAIR, UNBIASED, NON-DISCRIMINATORY)

AI systems should be inclusive and accessible and should not result in unfair discrimination against individuals or groups. AI systems should minimize unintended outcomes of such systems by providing capabilities for identifying and mitigating unwanted bias.

IN PRACTICE Providing tooling for evaluating model performance on subsets of data.

EXPLAINABLE (CONTEXTUAL, INTERPRETABLE, TRANSPARENT, UNDERSTANDABLE)

AI systems should not be black boxes. Instead, AI systems should be built to help stakeholders (users, regulators, supervisors, policy makers, advocates, etc.) understand a model throughout its lifecycle. AI systems should provide capabilities for associating contextual explanations with the model output.

IN PRACTICE Allowing for model interpretability on both individual examples and overall results.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

RELIABLE (SAFE, SECURE, RESILIENT, ROBUST)

AI systems should be built with capabilities for assessing the safety, security, and effectiveness of a model throughout its entire lifecycle. They should also be designed to reduce the potential impact of accidents or other unintended harmful behavior. Additionally, they should provide capabilities for preventing adversarial attempts aimed at degrading models or undermining the privacy and other fundamental rights of individuals whose data might have been used to train the models.

IN PRACTICE Offering a capability for recall and roll-back.

TRACEABLE (AUDITABLE, GOVERNABLE)

AI systems should provide the capabilities to document relevant development processes, data sources, and the provenance of all data used for building the model. Moreover, such systems should allow for both third-party oversight and internal audits.

IN PRACTICE Surfacing metadata about how a model was evaluated and subsequently deployed.

HUMAN-CENTERED (PARTICIPATORY, SOCIALLY BENEFICIAL)

AI systems should benefit individuals, society, and the environment overall. They should neither erode trust nor replace human decision making. Particularly for uses of automation that impact individuals' privacy, civil liberties, and other fundamental rights, the goal of AI systems should be to enhance quality of human judgment.

IN PRACTICE Representing model outputs in a manner that provides context to the end-user.

SCOPED (PROBLEM-DRIVEN, REPRODUCIBLE, RIGOROUS)

AI systems should be built for a well-defined and appropriately-scoped purpose. It should be expected that models powering the system are useful within that scope, but outside of that scope, no such guarantee holds. All steps of the model lifecycle must be performed with scientific rigor, so that model results can be reproduced for a given modeling problem.

IN PRACTICE Clearly recording the purpose and intended goal of the model.

COLLABORATIVE (MULTI-STAKEHOLDER, INTERDISCIPLINARY)

Building an AI system should be an interdisciplinary process where scientists, engineers, domain-experts, product managers, ethicists, compliance officers, and other relevant stakeholders work together. This theme undergirds the entire framework. There is widespread acknowledgement that the challenges of AI are technical, organizational, legal, and societal in nature, and expertise from a diverse breadth of disciplines are necessary to tackle them.

IN PRACTICE Providing interfaces for non-technical and technical stakeholders to engage.

ACCOUNTABLE (LIABLE, RESPONSIBLE)

Accountability has widely been cited as an important consideration for the development of algorithms and models. To put accountability into practice, there should be a clear definition of the roles and workflows for people responsible for the different parts of an AI system.

IN PRACTICE Clearly defining individuals and their roles in the model lifecycle.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

Model Lifecycles: The Emergence of ML Operations

Developing and deploying models in a production environment is a complex task. From strategies for evaluating a model, to monitoring the model's performance in production, to iterating on the model itself, there are a variety of tasks that several stakeholders must work collaboratively on to use AI in practice. Machine Learning Operations, or MLOps, for short, has emerged as a new discipline that focuses on improving the process of developing an AI system and in doing so addressing these challenges.

At its core, the goal of MLOps is to put models into production consistently and quickly. MLOps paradigms are mostly built on software engineering and DevOps techniques like version control, continuous integration (CI), continuous delivery (CD), automation, and monitoring. This approach recognizes that many of the challenges of ML and other types of modeling stem from the inability to scale the tasks involved in model development, deployment, maintenance, and monitoring

Many software platforms for building AI systems address these technical and organizational challenges by providing MLOps tools for different stages of the model lifecycle – from defining a modeling problem, to developing a model, through model evaluation, deployment, and monitoring. As we will explain below, building these types of MLOps tools into software platforms for AI systems is crucial for enabling engineering teams to promote RAI themes in practice.

RAI Roles

While most MLOps paradigms primarily focus on the technical stakeholders within the model lifecycle – software engineers, data scientists, and data engineers – the success of an AI system often relies on a broader set of stakeholders, including users with different domains of expertise, functions, and technical skill levels.

We distilled the common roles in MLOps workflows most relevant to implementing RAI in practice. Along with a description of the role function, we present the corresponding job titles for individuals who might fulfill this function across different organizations and the core skills and processes involved.

Importantly, the roles describe functional categories relevant for building and maintaining AI systems. And so, depending on the context of the modeling problem and organizational structure, one person might carry out multiple roles or many individuals might carry out one role. The roles identified, therefore, do not necessarily represent distinct individuals, which is common in other MLOps paradigms.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

MODELING PROBLEM OWNERS *(PROJECT LEAD, PROJECT MANAGER, DATA SCIENCE LEAD, SOLUTION ARCHITECT)*
Modeling problem owners are domain-experts who want to improve a process with an AI system. They could be intimately aware of how the AI system will be used or have deep subject matter expertise, including an understanding of key performance indicators (KPIs) that modeling is aimed at improving.

MODEL BUILDERS *(DATA SCIENTIST, STATISTICIAN, ML ENGINEER)*
Model builders are the technical stakeholders in the model lifecycle who understand the science of data. Their responsibilities include making the data modeling-ready, building the model logic, tuning the model, and performing validation using held-out data, among other model development tasks.

EVALUATORS *(COMPLIANCE AND DATA-PROTECTION OFFICER, PROJECT MANAGER, DATA SCIENTIST)*
Evaluators are responsible for conducting thorough evaluation of models to ensure they are fair and performant for the specific modeling problem. Evaluators are expected to be domain experts, have some technical background, and are aware of the context in which the model is going to be used. In some cases, the modeling problem owner could be the evaluator. When models must be deployed in high-stakes scenarios or heavily regulated industries, compliance and data-protection officers could also be involved in the evaluation process.

SYSTEM MANAGERS *(DEVOPS ENGINEER, MLOPS ENGINEER, SYSTEM ARCHITECT)*
System managers have the critical role of managing the AI system infrastructure. Their role includes managing compute environments, release and deployment processes, system availability, integrations with external systems, and other DevOps tasks.

END-USERS *(OPERATIONAL USER, ANALYST, DECISION MAKER)*
End-users are the individuals who use the output of an AI system. In an ideal scenario, the end-user has context about the domain of the modeling problem and can interpret the AI system output to make decisions.

APPLICATION BUILDERS *(PRODUCT MANAGER, SOFTWARE ENGINEER, DESIGNER, DATA SCIENTIST)*
Application builders are the individuals who enable the end-users to make meaningful use of the outputs of the model. It is unlikely that the end-user directly interacts with the raw output of the model. Instead, the model output is usually represented in a more interpretable form to the end-user. For example, the application builder might build a point-and-click dashboard that incorporates the model output, combines the model outputs with other data sources for enrichment, or post-processes the output to send it to an external system.

DATA ENGINEERS *(PRODUCT MANAGER, SOFTWARE ENGINEER, DATA SCIENTIST)*
Data Engineers build and maintain pipelines that feed inputs to and process outputs from AI systems. They manage integrations with source systems, implement data pre-processing logic, or may even perform featurization prior to actual model building. Moreover, data engineers could also be responsible for integrating feedback into the modeling pipelines. We focus less on this role explicitly for RAI and hold other stakeholders responsible for investigating and accounting for data bias and quality issues.

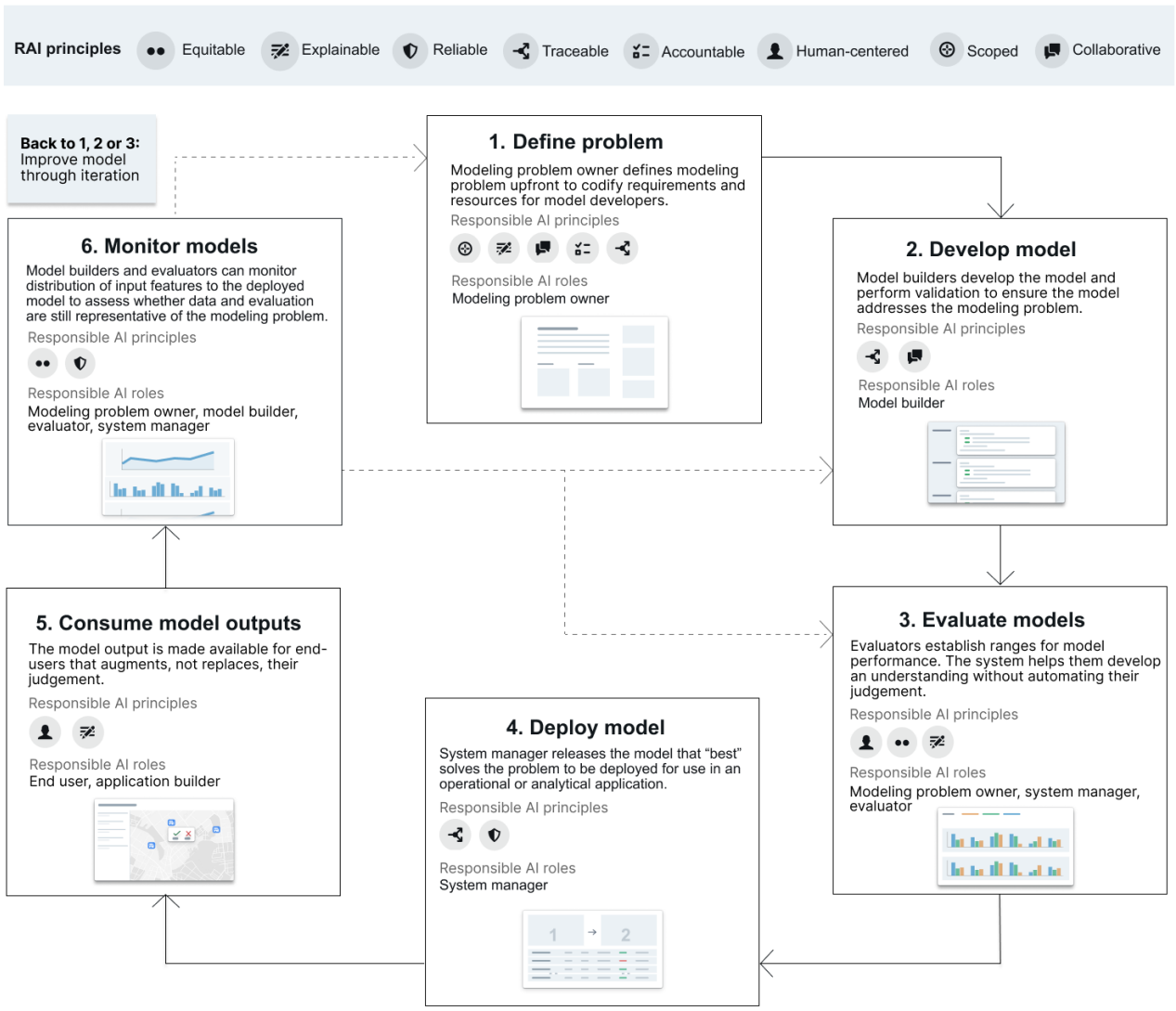
© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

Responsible AI Lifecycle Framework

The Responsible AI Lifecycle (RAIL), illustrated below, is a novel model lifecycle framework that emphasizes the key steps in the modeling process most important for RAI. Most MLOps lifecycles are not specifically focused on highlighting how to put RAI into practice. Instead, the goal of such lifecycles is to help teams build and deploy models rapidly and iteratively. While this goal is an important component of RAI, RAIL better addresses the multi-faceted nature of RAI, as described in the synthesis of RAI themes.

Software platforms for model development and deployment can implement RAIL to guide stakeholders towards RAI practices at crucial waypoints in the process of building and maintaining an AI system. The software foundation of this framework implies that once implemented, the use of the framework can easily scale within an organization and ultimately amplify a cultural shift towards RAI adoption.

Responsible AI Lifecycle (RAIL) Framework



© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

– "…the success of an AI system often relies on a [broad] set of stakeholders, including users with different domains of expertise, functions, and technical skill levels."

RAIL has six key steps, each of which is associated with the themes and roles earlier described. In the following sections, we describe each step, share the corresponding RAI themes and roles, as well as provide an example of how the step could be implemented within a software platform.

1. Define the modeling problem //

THEMES
SCOPED
EXPLAINABLE
TRACEABLE
COLLABORATIVE
ACCOUNTABLE

ROLES
MODELING PROBLEM OWNER

The first step of RAIL is for the *modeling problem owner* to define the modeling problem. Importantly, this problem definition should be qualitative and contextual and not just a description of the desired model output. For example, a *modeling problem owner* may want to use a model to reduce staff turnover, improve vaccine allocation, or sell more of a particular good. There are several benefits to starting a model lifecycle with a problem definition. First, changes to this definition can be versioned, audited, and reviewed. Second, since the problem is framed qualitatively and contextually, it does not prescribe any constraints on the modeling approach that the *model builder* should use, which provides space for exploratory and light-weight modeling to better consider potential solutions. On the other hand, the *modeling problem owner* could specify invariants that a model for this problem must respect such as a required output format.

At this step, the *modeling problem owner* should also gather all resources and requirements for the *model builders* and make them available in a central and accessible space. This should include details about training and testing data; requirements regarding explainability, fairness, and compliance; legal considerations about how the model can or should be used; and even collaborators and their roles. Software platforms that implement RAIL can offer a capability to create a "modeling project" that houses both the problem definition as well as these gathered resources.

Modeling problem definitions help ensure that the AI system remains **scoped.** Collaborators can better understand the application and context for which the model was originally built, which could prevent inadvertent re-use of models.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

The definition also aids the development of more **explainable** models by reducing the tendency to use an overly-complicated solution for a problem that might have a simpler one or even by encouraging a non-ML-based solution.

Centralizing all the resources needed for model development in one place along with provenance details leads to standardization and better **traceability.** Since we encourage that the problem definition should also include information about all the stakeholders who will be working on the modeling problem and their roles and responsibilities, this promotes better **collaboration** and **accountability.**

Of all steps in RAIL, defining the modeling problem and the corresponding modeling project is the most crucial step as it codifies several RAI themes very early in the model lifecycle.

2. Develop the model //

THEMES
TRACEABLE
COLLABORATIVE

ROLES
MODEL BUILDER

Once the modeling problem is well-defined, *model builders* can develop models to address it. There are many technical practices that model builders can perform to assess model fairness, safety, and performance. RAIL, however, is not prescriptive about specific assessments at this step. In fact, most of these technical approaches are highly dependent on the architecture of the model and specifics of the domain.

After the *model builders* have developed a model, they should "submit" their model as a candidate solution to the modeling problem from Step 1. For software platforms that implement RAIL in practice, this could mean creating a link between the model and the modeling project, moving the model to the modeling project, or giving the *modeling problem owner* access to the model or the model interface. This will ensure that model metadata (e.g., details of the *model builder*, model architecture, methodology, or training reports) can be made available to the *modeling problem owner*, and all models that solve the modeling problem can be accessed from a consistent location.

This approach allows many teams of *model builders* to work on the same problem simultaneously – either independently or jointly, thereby improving **collaboration.** With increased collaboration, it becomes even more important to understand who built each model and how they did so. The submission mechanism provides an opportunity to collect any metadata about the model development process required to maintain **traceability.**

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

—"…it is central to Responsible AI that model evaluation remains human-centered."

3. Evaluate the model //

THEMES

HUMAN-CENTERED
EQUITABLE
EXPLAINABLE

ROLES

MODELING PROBLEM OWNER
SYSTEM MANAGER
EVALUATOR

In RAIL, evaluation is mandatory for all models. Tools for evaluation should help the *evaluator* be more effective and not subsume the *evaluator*'s role and responsibility altogether. In some cases, there might be approaches to make the evaluation process more efficient by automating specific repetitive steps, but it is central to RAI that model evaluation remains **human-centered.**

In this step of RAIL, models should be consistently and comprehensively evaluated. Specifically, we illustrate two common strategies for model evaluation that a software platform that implements RAIL could offer.

METRIC-BASED EVALUATION: Upon submission, evaluators can apply the models to test data to obtain metrics necessary to assess model performance. Such metrics could stem from best practices of the problem domain, organizational conventions, or standard techniques from industry or academia. Metrics allow for direct comparison with models that were deployed in the past for the same modeling problem, including baseline models and benchmarks. From an **equity** lens, it is specifically important to evaluate whether models generalize and have comparable performance across different subsets of data. Average metrics over the entire test data can hide disparate model impacts on subsets within the data. Additionally, *evaluators* can use techniques for global **explainability** like feature importance scores to better understand how the model might perform in the intended production context.

CHECK-BASED EVALUATION: Evaluation procedures can also enforce more qualitative checks to strengthen **reliability**. Some examples of such checks include verifying if the *model builder* handled class imbalance, validating the exclusion of protected attributes and their proxies, or performing quality assurance tests for model deployability. The evaluation criteria can also be made stricter by requiring all checks to pass before the model can be deployed. *System managers* can also leverage infrastructure-related checks. These checks enforce the accountability of *evaluators* and *system managers*, as these stakeholders are responsible for ensuring that the model meets all criteria in the problem definition from Step 1.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

4. Deploy the best model //

THEMES
TRACEABLE
RELIABLE

ROLES
SYSTEM MANAGER

In the next step of RAIL, the *system manager* releases the model that "best" solves the problem defined in Step 1 based on the *evaluator*'s conclusions in Step 3, thereby deploying that model for use in a downstream application.

Much like "releases" in the software context, RAIL also uses the notion of releases for updating models in production. Software platforms that implement RAIL should integrate model releases with version control software and record all model metadata from prior steps at the time of release for **traceability.** This metadata also makes it easier to retrospectively answer questions about how the deployed model was evaluated, why model outputs may have changed over time, or who was responsible for making the deployment decision.

Much like with deployment of software releases, software platforms that implement RAIL can offer staged deployment processes for **reliability.** *System managers* can first send models to a canary or staging environment for stability testing before deploying them into a production environment. Moreover, providing rollback mechanisms at this step can improve the safety and fault tolerance of the AI system.

5. Use model outputs //

THEMES
HUMAN-CENTERED
EXPLAINABLE

ROLES
END USER
APPLICATION BUILDER

In this step of RAIL, once the model is deployed, the *application builder* makes the model output available to the *end-user* so that the *end-user* can use the model to make an informed decision. This nuance often does not get much attention in MLOps frameworks but is especially important for RAI.

At this point in the model lifecycle, *model builders* have limited control over how the models they developed are used or how *end-users* interact with model outputs. Ideally, the problem definition from Step 1 contains the relevant context about expected model use, but there are additional design, human-computer interaction, and agile development challenges that *application builders* need to explore at this stage. For example, *is there a way for the end-user to provide feedback on model predictions? Is the model enhancing human decision-making? Are there ways to fall back to a manual workflow when model prediction quality degrades?*

Critically considering how the model output is represented to the end-user drives **human-centered** model development. In high-stake settings, *application builders* should always ensure that models are complementing, not replacing, the *end-user*'s judgement.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

Moreover, at this step, *application builders* should be able to probe whether the model is **explainable** and determine if *end-users* will be able to understand why a model predicted a certain output.

6. Monitor the model and improve through iteration//

THEMES
RELIABLE
EQUITABLE

ROLES
MODELING PROBLEM OWNER
MODEL BUILDER
EVALUATOR
SYSTEM MANAGER

The final step of RAIL captures the idea that monitoring a model and its deployment context can inform previous stages of the model lifecycle. Specifically, such monitoring can help measure model drift or degradation.

Modeling problem owners, *model builders*, and *evaluators* can use the distribution of input data to the deployed model to assess whether the problem definition, training data, and evaluation criteria are still representative of model's real-time usage. In addition, monitoring the way the model is used in production can also help determine if the modeling problem definition needs to change since it was originally formulated in Step 1.

Monitoring allows stakeholders in the model lifecycle to address **equity** concerns that may arise when models are applied on data with a different distribution than that of the training data. Monitoring deployed models is also crucial for **reliability** and fault tolerance of AI systems. It empowers *system managers* with the information they need to apply release and rollback processes described in Step 4. Such information includes the live input data to the model, outputs from the model, system logs, and any end-user feedback on model predictions. Additionally, these insights can inform model re-training workflows in Step 2.

— "In high-stake settings, application builders should always ensure that models are complementing, not replacing, the end-user's judgement."

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

Why does RAIL matter?

RAIL provides a model lifecycle that emphasizes RAI considerations, providing engineers with a novel and practical approach for promoting RAI when building software for AI systems. Where existing RAI frameworks are often abstract and focused on models themselves, RAIL emphasizes specific themes and describes their application to AI systems. RAIL is differentiated from other MLOps frameworks in these and many other key ways, several of which are highlighted below.

PROBLEM-FIRST MODEL LIFECYCLE

RAIL emphasizes problem-first modeling over the commonly used data-first approach. In a data-first approach, data scientists start with data or a refined feature store and then perform exploratory data analysis to gauge whether useful ML models can be built. Many MLOps platforms optimize for such feature-store driven modeling processes. From an RAI lens, using only the data-first approach to modeling has several shortcomings. First, what "can" be built does not always align with what "should" be built. For example, if a company has access to CCTV footage data from its security cameras and wants to use this to understand employee productivity, there are contextual, cultural, ethical, and potentially legal considerations that the organization must consider before pursuing such a workflow. The problem-first approach makes these considerations "shift left," far earlier in the model lifecycle. Second, in the data-first approach, it is harder to be prescriptive about "what good looks like" without a scoped problem definition. Moreover, if AI systems do not have clear organizational ownership or purpose, there is a heightened risk of model misuse or erroneous repurposing.

The data-first approach and exploratory model development, however, are not entirely antithetical to RAI. There are ways of combining the two by incorporating a modeling problem definition step as a first-class entity within the data-first approach. In some cases, RAIL can even benefit from data-first modeling, for example, data-first modeling can be used to explore available data and more concretely define the modeling problem in the first step of RAIL.

EXPANDED SCOPE

RAIL expands beyond the scope of most MLOps frameworks to account for crucial waypoints where stakeholders should consider RAI themes. This expanded scope includes key decisions at early, pre-development stages of the model lifecycle and the many RAI challenges that surface after a model is put in production. For example, post-production considerations may include technical challenges like model drift, design challenges about how to represent model outputs, and organizational challenges like ensuring that *end-users* have the appropriate training and context to understand the AI system. As such, RAIL starts before model development and extends beyond model deployment.

GENERAL PURPOSE FRAMEWORK

RAI practices can be beneficial for a wide variety of model development techniques – from ML to statistical modeling to rule-based business logic. In this way, RAIL is designed to be agnostic to the specific

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

modeling technique used to develop the model in Step 2. Abstracting the model lifecycle framework away from any one type of modeling technique draws out the importance of RAI principles and themes across a variety of domains and approaches.

CENTRALITY OF OPERATIONAL CONTEXT

One of the shortcomings of existing MLOps frameworks is that they do not adequately place the relevant business and operational stakeholders within the model lifecycle. Specifically, many frameworks describe these stakeholders in a "business understanding" or "problem definition" step, but then do not describe the role of these stakeholders throughout the rest of the model lifecycle or how they should engage with other roles.

RAIL addresses this issue by situating the modeling problem owner – the function that many of these business and operational stakeholders carry out – and their definition of the modeling problem at the center of the AI system development process. For example, consider a bank building a model that predicts a client's income. A data scientist at the bank might take a different approach to building the model if the model output – the client's predicted income – is going to be used for a loan application review process compared to a product recommendation application. Evaluators might even consider using different evaluation criteria for these two problems despite the same model output requirement because of the heightened sensitivity around loan application reviews. The modeling problem owner's problem definition acts as a forcing function to consider such application context within the other lifecycle steps while ensuring that different stakeholders are able to engage with the process.

Looking Ahead: Potential extensions of RAIL

The RAIL Framework is deliberatively lean to highlight the most important considerations and give engineering teams flexibility on how to implement it in practice. However, RAIL could be further expanded to better suit certain problem domains, engineering practices, or organizational structures. Specifically, the following four areas may be considered:

ADDRESSING ORGANIZATIONAL GOVERNANCE

This whitepaper has focused on the way that RAIL can influence software engineering decisions to build systems for more responsible model development, deployment, and use. However, RAIL could also influence important organizational functions outside of engineering teams, including governance, policy, and oversight responsibilities.

For example, compliance or data protection teams may play a critical role in granting model builders access to training data in the first place and understanding for what purpose that data might be used. Though these other facets of implementation are out of the scope of software engineering, they may present a valuable direction for developing a whole-of-organization approach to RAI.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

INTRINSIC AND EXTRINSIC VIEWS OF RAI THEMES

For many of the themes identified in the first section of this whitepaper, there are two perspectives through which one can view their relevance to RAI. First, one can consider the viewpoint of stakeholders who are involved in the model lifecycle. This is the approach RAIL takes, and we describe this as RAI themes *intrinsic* to the model lifecycle. On the other hand, one can consider the application of the RAI themes *extrinsic* to the model lifecycle, where the themes guide interactions between the AI system and the individuals impacted by it. These impacted individuals may not be the *end-users*. For example, if an AI system is used to reduce average waiting times in a call center queue, the *end-user* may be a call center employee, but the impacted party is the caller.

To illustrate the different considerations between *intrinsic* and *extrinsic* applications of RAI principles, we can consider the accountability of an AI system used to provide medical advice to patients. The *intrinsic* view of the accountability theme would translate to clear intra-organizational accountability over the model development and deployment process. For example, a *model builder* could be accountable to a compliance team to ensure no personal health information was used during model development. This principle guides the processes of stakeholders within the healthcare organization iterating on the AI system. On the other hand, an *extrinsic* view of accountability would apply to patient interactions with the AI system. For example, the healthcare organization might be accountable to the patient if the AI system provides incorrect medical advice, and the patient seeks some means for redress.

RAIL focuses mostly on the intrinsic perspective of the RAI principles, and future work on RAI frameworks could explore extending RAIL to address the application of these themes from an *extrinsic* perspective.

SUSTAINABLE RAI

RAIL is particularly focused at early-stage or actively-developed AI systems and does not capture the full depth of considerations around mature, long-lasting AI systems. There are further considerations about long-term model sustainability and strategies for model maintenance that could be accounted for in a more expansive adaptation of RAIL.

ADDITIONAL MODEL LIFECYCLE STEPS

RAIL is a general-purpose lifecycle framework, and certain modeling approaches may require additional steps in the model lifecycle or changes to the lifecycle structure. For example, when working with large language models (LLMs), prompt engineering and management are important components of the model lifecycle beyond model development and model evaluation. In this case, both prompt evaluation and model evaluation would be pre-requisites to the deployment step. To better account for invariants of specific modeling problems, RAIL can be extended to map RAI themes and roles to these kinds of novel model lifecycles.

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.

— "…AI that is both effective and responsible must focus on the fully integrated system, not just its component tools. "

Palantir Technologies' approach to AI Ethics

Palantir has accumulated a considerable range of experiences over the years that has informed and enriched our approach to developing, building, and deploying AI enabling technologies. Our approach is constructed on a recognition that AI innovation is not only compatible with security, privacy, data protection, and other fundamental rights interests, but it is also most effective when it supports rights-protective outcomes. The RAIL Framework reflects this position, asserting that AI that is both effective and responsible must focus on the fully integrated system, not just its component tools.

To ensure our products meet the legal and ethical requirements in practice, Palantir has invested in an in-house privacy and civil liberties (PCL) engineering team. This interdisciplinary team of software engineers, data scientists, philosophers, designers, and policy experts is available to meet with customers to help them develop and implement responsible, accountable and ethical AI that is oriented toward humans.

Read more about Palantir's approach to AI Ethics at https://www.palantir.com/pcl/palantir-ai-ethics

© Palantir Technologies Inc. The information in this document is proprietary and confidential, and includes certain Palantir trade secrets. Unauthorized disclosure to any third party is strictly prohibited. The content herein is provided for informational purposes only and shall not create a warranty of any kind.