# Component Mismatches Are a Critical Bottleneck to Fielding AI-Enabled Systems in the Public Sector

## AAAI 2019 Fall Symposium Series
## AI in Government and Public Sector


Grace A. Lewis

Stephany Bellomo

April Galyardt

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213





[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

Copyright 2019 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

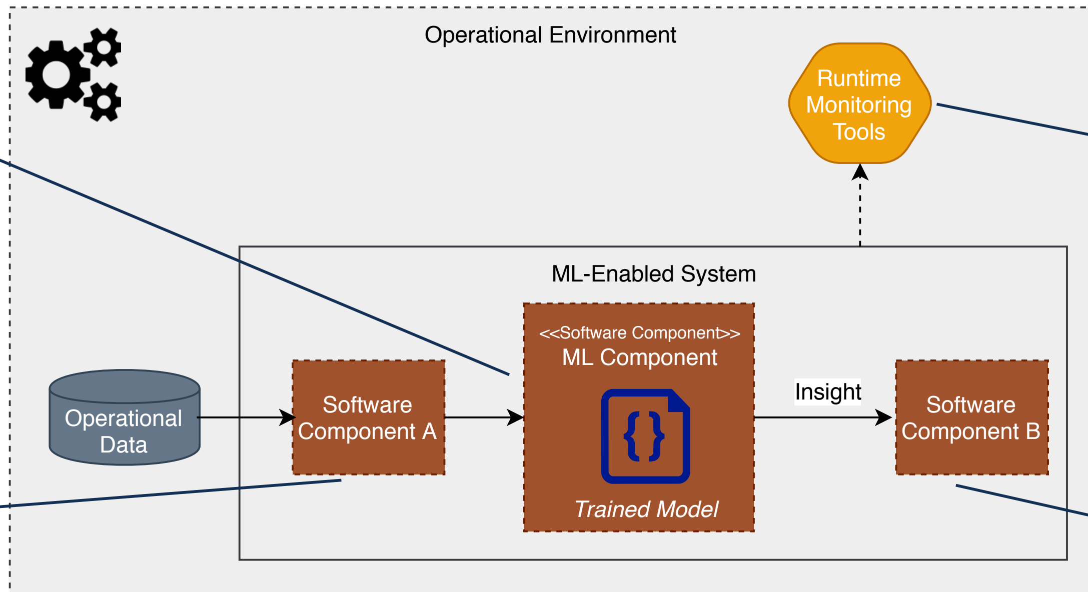
DM19-1090

# ML-Enabled System

We define an ML-enabled system as a software system that relies on one or more ML software components to provide required capabilities

A trained model(s) is wrapped as a special type of Software Component called an ML Component

ML component receives (processed) operational data from one software component …

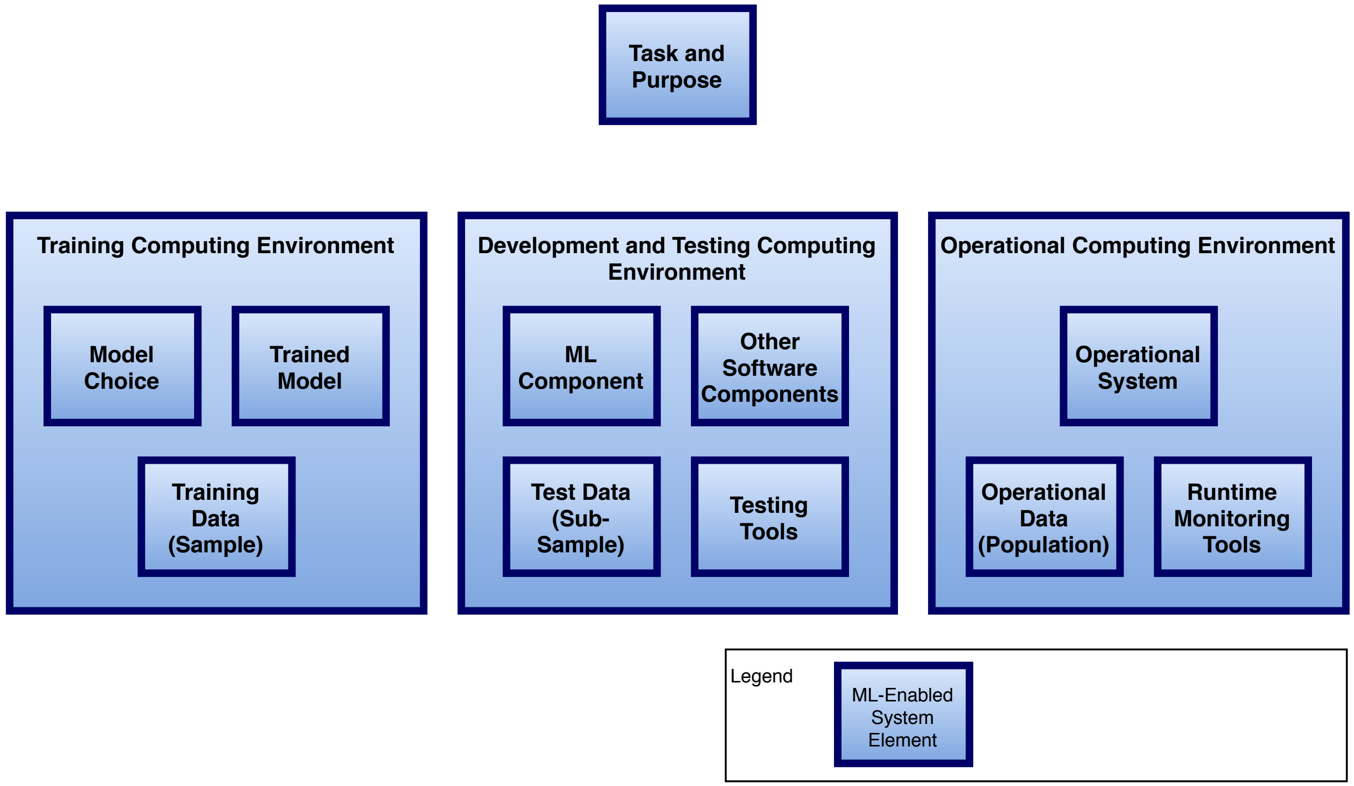



Runtime Monitoring Tools in the operational environment obtain and react to measures produced by the ML-Enabled System

… and generates an insight that is consumed by another software component.


© 2019 Carnegie Mellon University
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Elements of ML-Enabled Systems

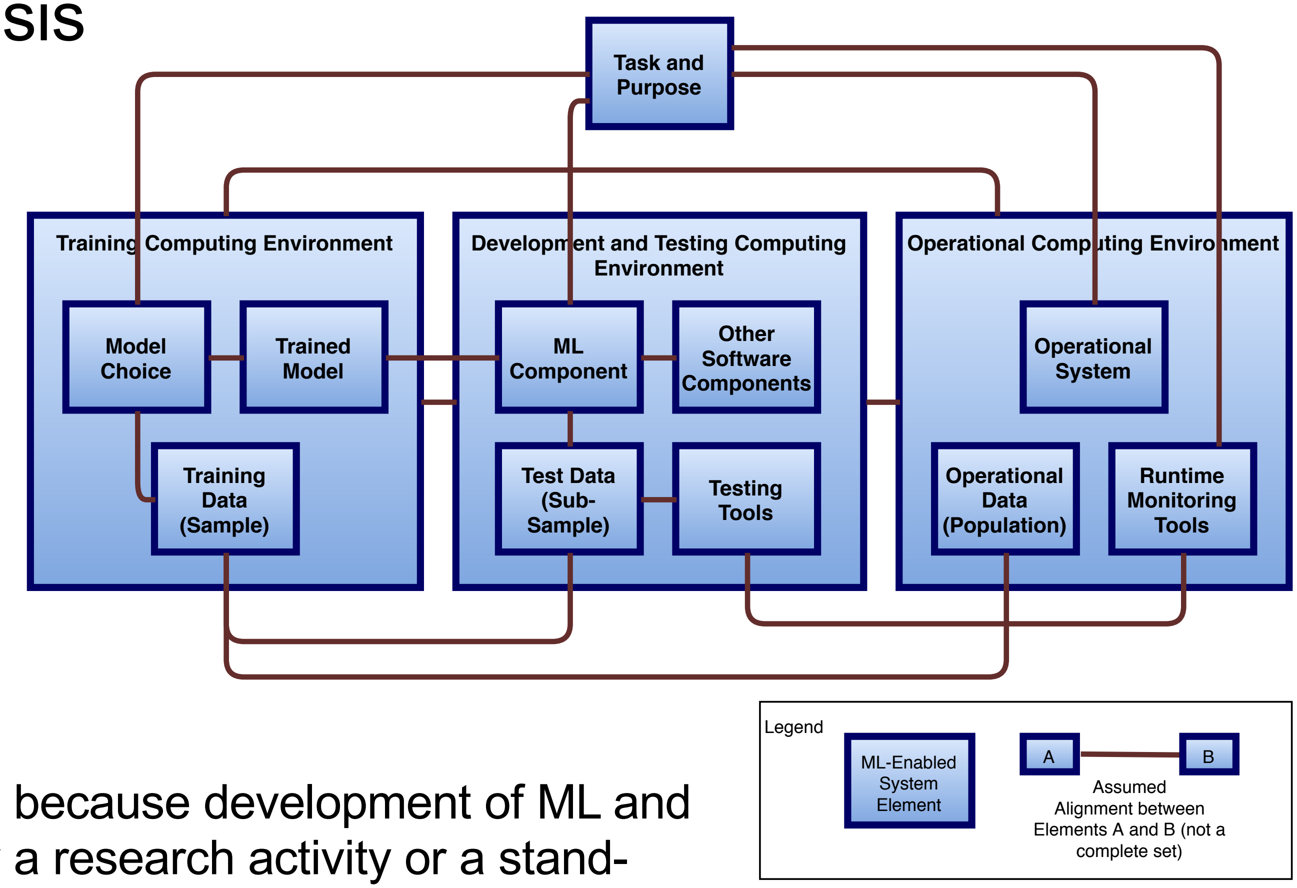



We define elements of ML-enabled systems as the non-human entities involved in the training, integration and operation of ML-enabled systems.


© 2019 Carnegie Mellon University
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Motivation/Hypothesis

Many of the challenges for deploying ML-enabled systems into operational environments is due to mismatch between elements of ML-enabled systems




Very little existing guidance because development of ML and AI capabilities is still mainly a research activity or a stand-alone project, with the exception of large companies

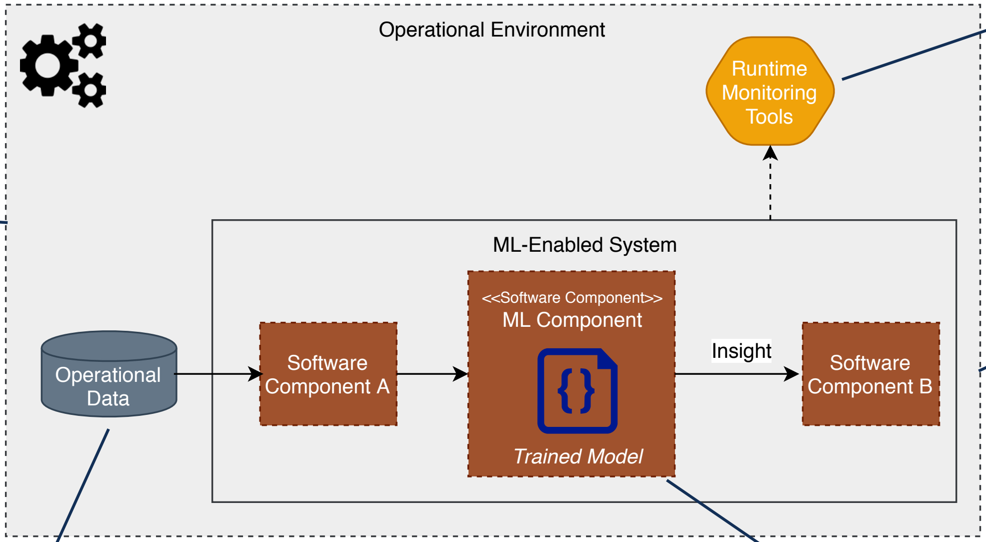


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Examples of Mismatch

Poor system performance because computing resources for model testing different from operational computing resources (**computing resource mismatch**)




Tools not set up to detect diminishing model accuracy, which is the "goodness" metric defined for the ML component (**metric mismatch**)

System failure due to poor testing — developers not able to replicate testing done during model training (**test environment mismatch**)

Poor model accuracy because model training data different from operational data (**data distribution mismatch**)

Large amounts of glue code because trained model input/output very different from operational data types (**API mismatch**)

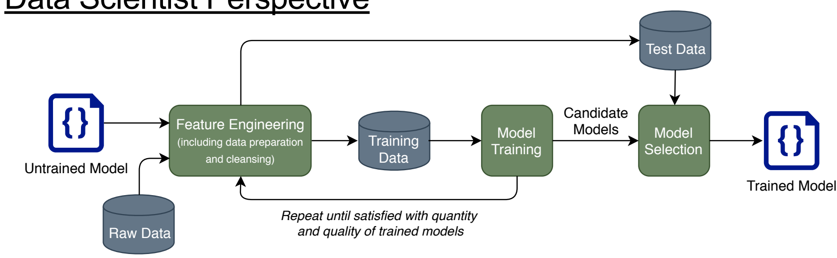


© 2019 Carnegie Mellon University

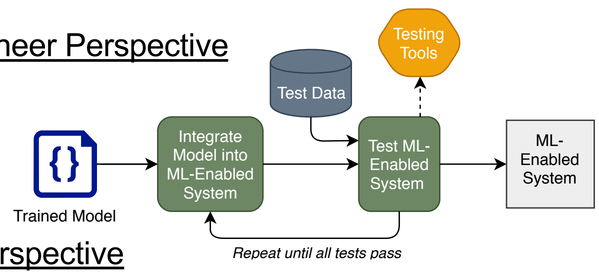
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Problem: Multiple Perspectives

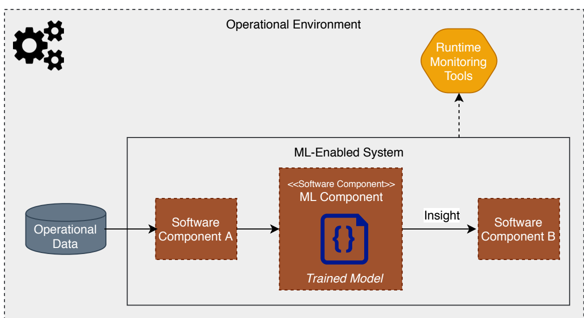
Data Scientist Perspective




Software Engineer Perspective




Operations Perspective




ML-enabled systems typically involve three different and separate workflows

- Model training
- Model integration
- Model operation

… performed by three different sets of stakeholders ...

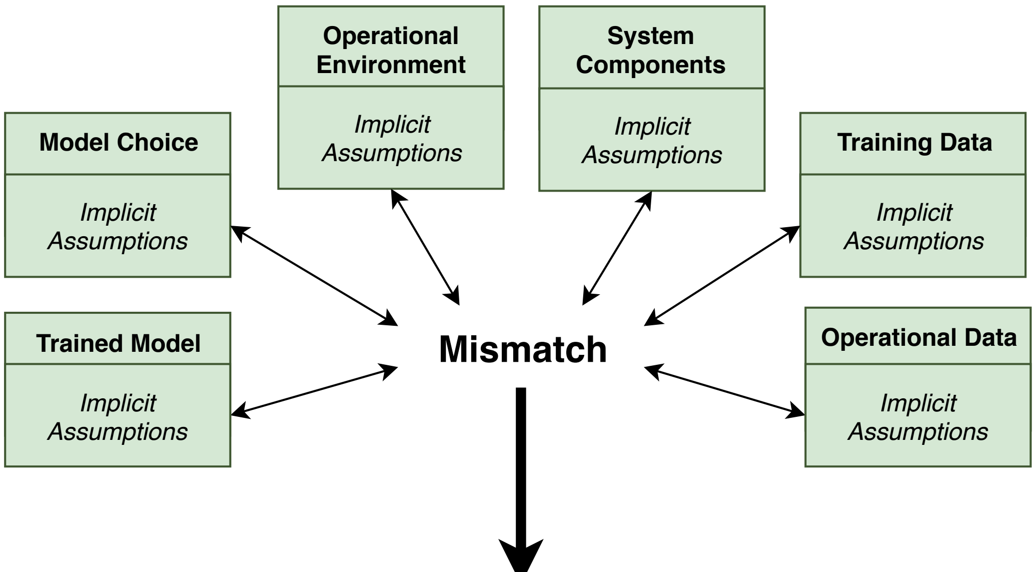
- Data scientists
- Software engineers
- Operations staff

… with three different perspectives


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Problem: Mismatch between Assumptions made by each Perspective




Late discovery of mismatch results in

- System delivery delays due to rework
- Incorrect results
- Poor system performance
- **Mission failure**

For an operational system to produce appropriate results, all of these elements and assumptions must remained aligned.

As each element evolves independently and at a different rhythm, this increases the risk of unintentional mismatch arising over time.


© 2019 Carnegie Mellon University

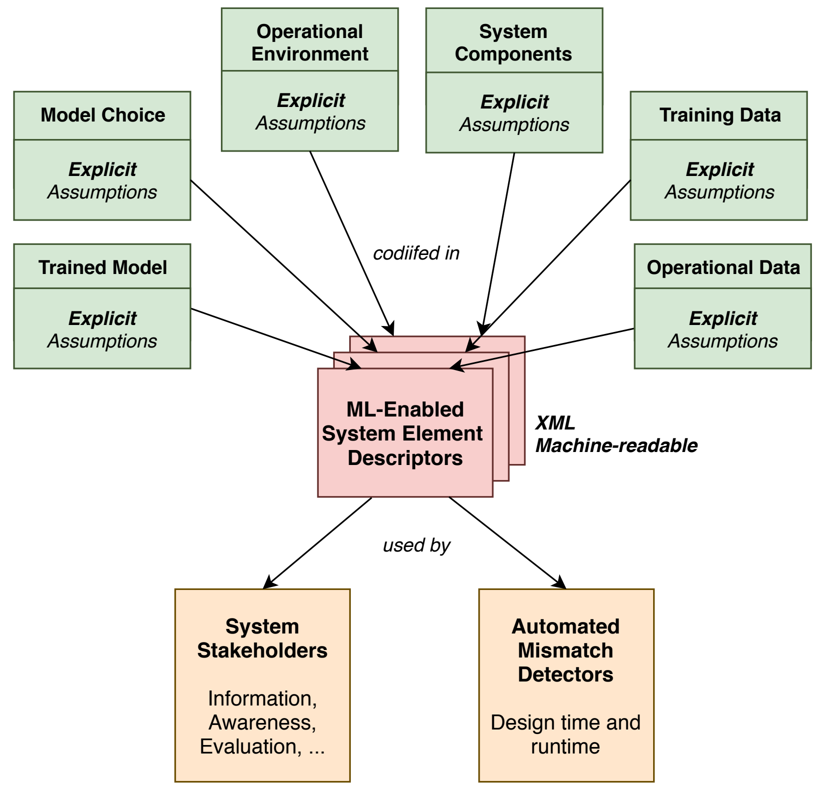
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Solution: Mismatch Detection and Prevention in ML-Enabled Systems

**Longer-Term Vision**: Codified assumptions and tools exist that allow many types of mismatch to be prevented and/or detected, at design time and runtime

**Contribution of this Project:** Develop descriptors for elements of ML-enabled systems by

- eliciting examples of mismatch from practitioners
- formalizing definitions of each mismatch in terms of data needed to support detection
- determining sources and validation mechanisms for this data
- identifying potential for using this data for automation of mismatch detection





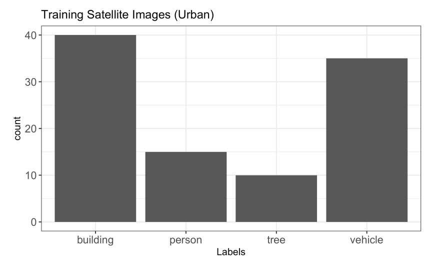

© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Simple Example Using Descriptors for Mismatch Detection

*Data Scientist trains model and fills in descriptor values for distribution of the training data (Attribute TrD-A1) and accuracy of the trained model (Attribute TM-A7)*

Training Data: A1: Data Distribution =

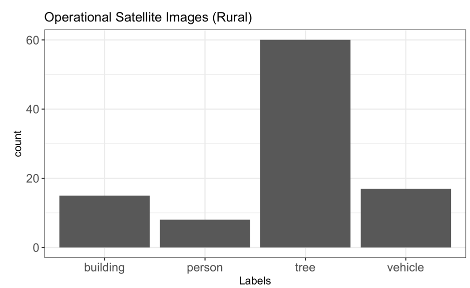



Trained Model: A7: Accuracy = 92%

1. *Software Engineer receives the trained model and verifies that the accuracy of the trained model (Attribute TM-A7) is greater than the required accuracy (Attribute MLC-A5)*
   ***(MLC-A5 <= TM-A7)***
2. *Software engineer integrates and tests the trained model. Measured accuracy for available test data is **65%**.*
3. *Software engineer runs test data through the "Data Distribution Analysis Tool" and discovers that the distribution of the test data (Attribute TeD-A2) is different from the distribution of the training data (Attribute TrD-A1), which could be the cause of the accuracy differences.*

Machine Learning Component: A5: Required Accuracy = 90%

Test Data: A2: Data Distribution =




Formalization Used by Data Distribution Analysis Tool (Chi-square test)

$$\sum_{i \in \text{Labels}}^{d} \frac{(X_i - Np_i)^2}{Np_i} > T$$

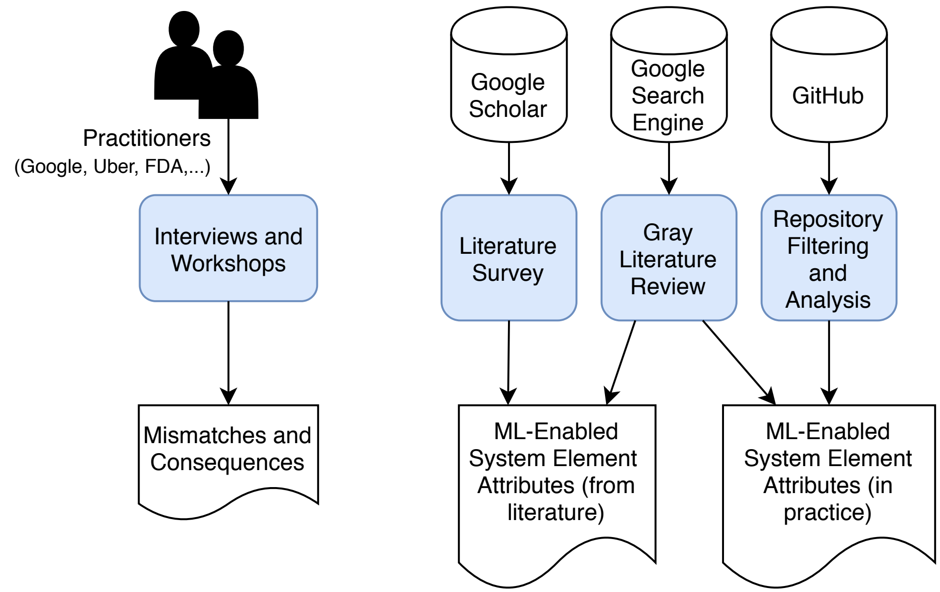


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Technical Approach: Information Gathering

1. Identify examples of mismatches and their consequences via interviews, workshops, and other mechanisms
2. Identify attributes for describing elements of ML-enabled systems
   - Mining descriptions from GitHub repositories that contain ML models
   - Literature survey
   - Gray literature review



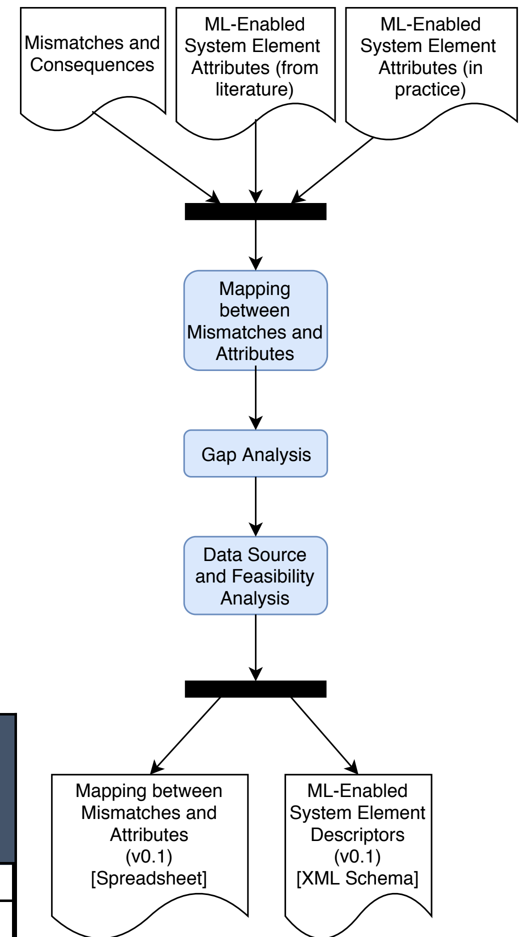


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Technical Approach: Analysis

3. Mapping between mismatches and attributes
   - For each mismatch, what is the set of attributes needed for detection, expressed as a predicate over identified attributes
4. Gap analysis
   - Which mismatches do not map to any attribute (and vice versa)?
   - What additional attributes are necessary for detection?
5. Data source and feasibility analysis
   - For each attribute, what is the data source, it is feasible to collect, how can it be validated, and is there potential for automation?




| Mismatch | Descriptors | | | | | | | | | | | | | | | | | Formalization |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Trained Model | | Training Data | | Untrained Model | | | System Components | | | Operational Environment | | | Operational Data | | | Descriptor M | |
| | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 | A16 | Am | |
| Mismatch 1 | X | X | | | X | | | | | | | | | | | | | A1 + A2 > A5 |
| Mismatch 2 | | | | | | | | X | | | | X | | | | | | A8 = A12 |
| … | | | | | | | | | | | | | | | | | | |
| Mismatch N | | | | X | | | | | | | | | | X | | | | Chi-Square(A4, A14) |

An = Attribute n

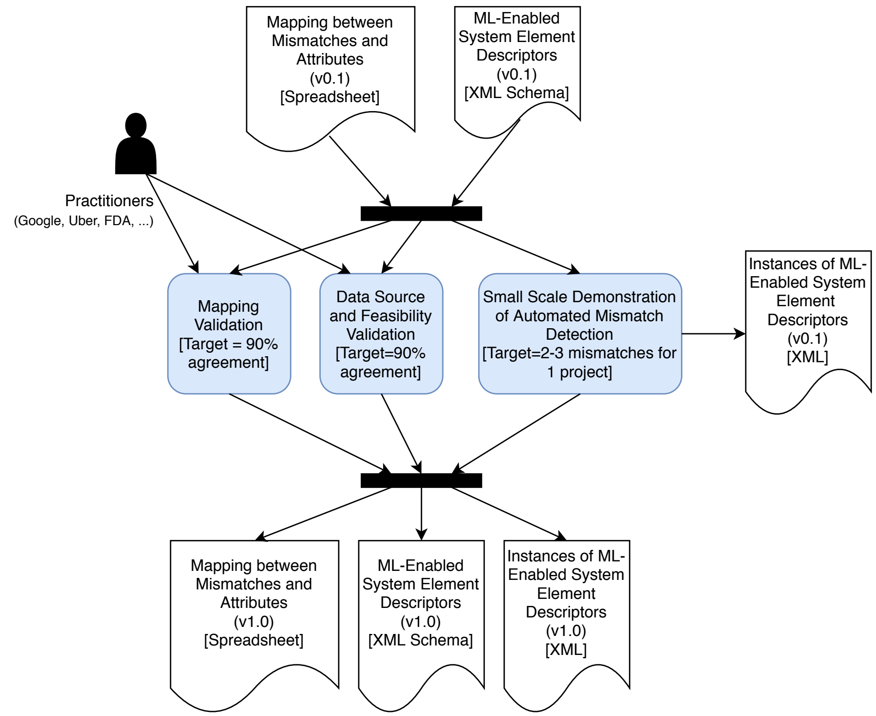


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Technical Approach: Evaluation

6. Mapping validation with practitioners
   - Target is 90% agreement on the definition of the mapping between mismatches and sets of attributes
7. Data source and feasibility validation with practitioners
   - Target is 90% agreement on data sources and collection feasibility for attributes
8. Small Scale Demonstration of Mismatch Detection
   - Target is to identify 2-3 mismatches in a project that can be detected via automation and develop scripts that can detect the mismatch






© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Applications of Project Results

- Definitions of mismatch can serve as checklists as ML-enabled systems are developed
- Recommended descriptors provide stakeholders (e.g., program offices) with examples of information to request and/or requirements to impose
- Means identified for validating ML-enabled system element attributes provide ideas for confirming information provided by third-parties
- Identification of attributes for which automated detection is feasible defines new software components that should be part of ML-enabled systems



© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Points for Discussion

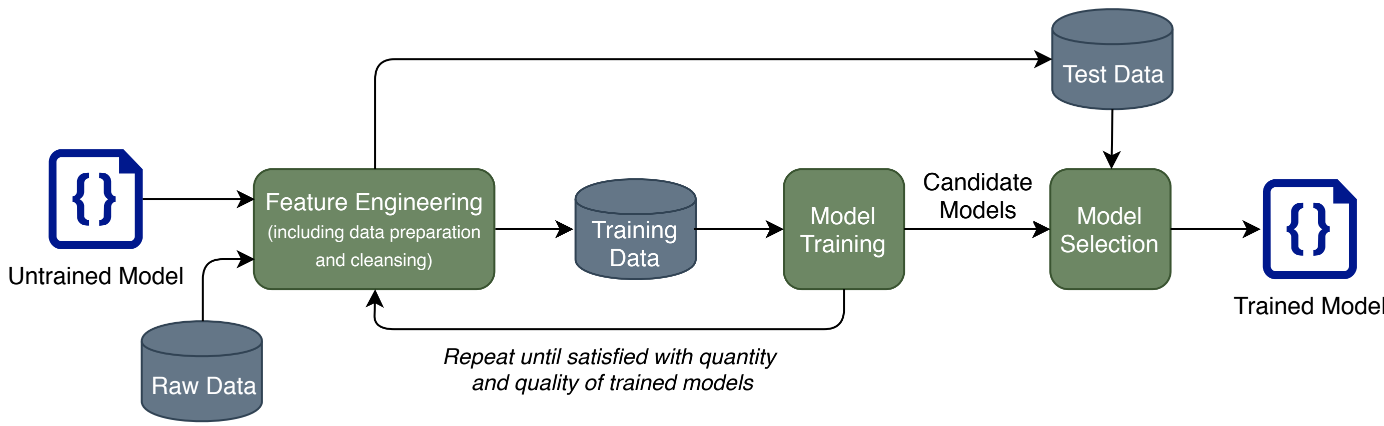
1. Is mismatch one of the challenges that you have experienced when deploying ML-enabled systems into production settings?
2. Do you have examples of mismatch that you can share?


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Additional Slides

# Data Scientist Perspective




**Data scientists** focus on the statistics

- Work in an exploratory process
- Main objective is high confidence in produced insight (e.g., classification results)

The goal of the data scientist is to create a *Trained Model* from an *Untrained Model,* plus some collection of *Training Data*.

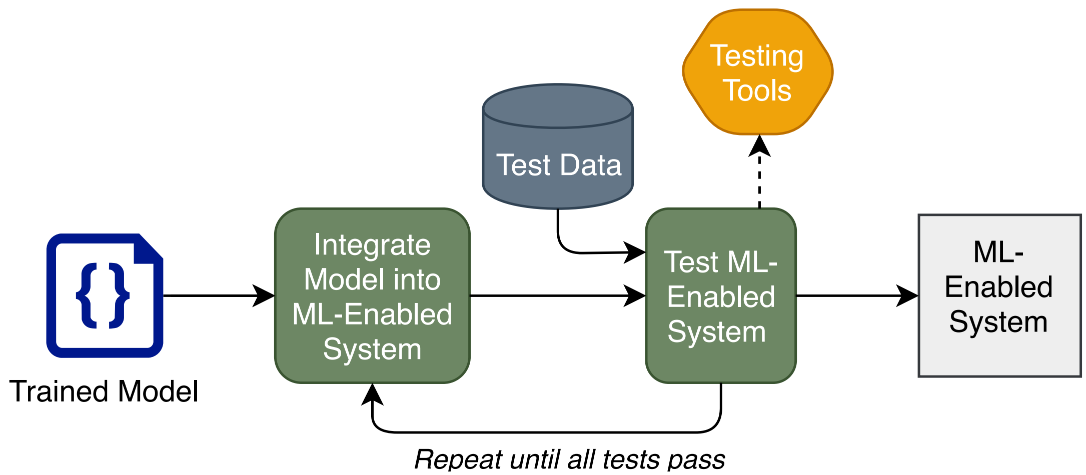
In the best case, they have some notion of the *Operational Data* and the requirements and assumptions of the *ML-Enabled System* in which the trained model will be integrated.



© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Software Engineer Perspective




**Software engineers** focus on getting correct software components integrated to serve mission
- Focus on typical software concerns: functional correctness + performance, security, maintainability, ...

Software engineers assemble an *ML-Enabled System* from a number of *Software Components*, some of which contain a *Trained Model*.

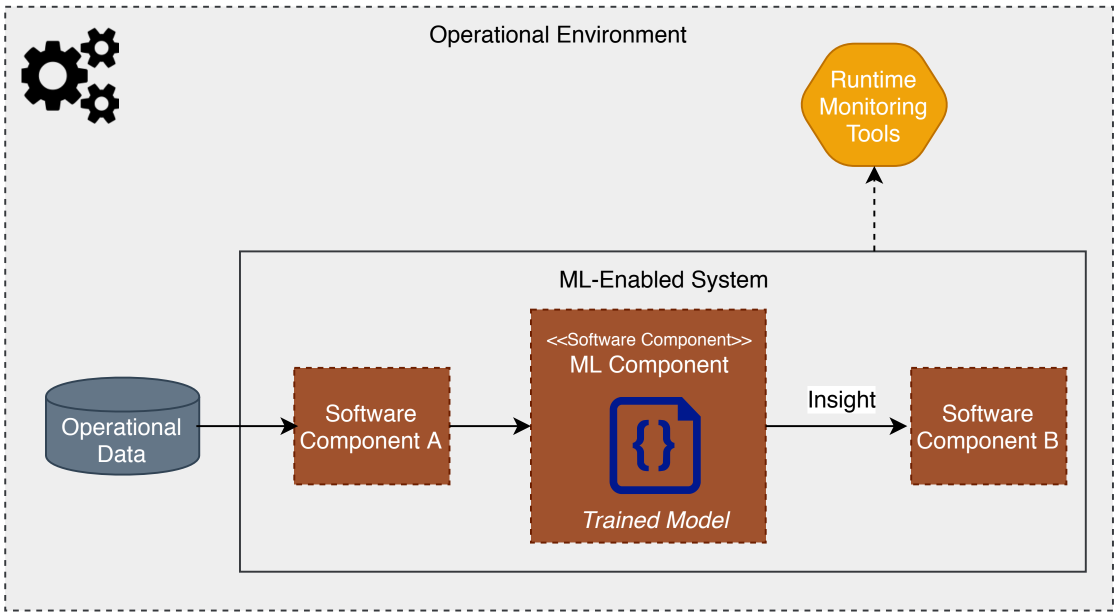
They generally have some notion of what kind of *Operational Data* they are targeting, but they may not ask the same questions that a data scientist would have looking at the same data.



© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Operations Perspective




**Operations staff** focus on monitoring operational results and failures

- More attuned to crashes and bugs than statistical drift in input as operational conditions change
- Do not have information to determine if an insight produced by the system is correct

Operations staff monitor the performance of the *ML-Enabled System*, but often without deep insight into its structure — *Software Components* (including ML components)

Operations staff also deploy and maintain the *Operational Environment* and the sources of *Operational Data*


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Sample Attributes Extracted from GitHub

| Search String | Repos* |
| --- | --- |
| “machine learning” + ”trained model” | 63 |
| “machine learning” + system | 1,685 |
| “machine learning” + model | 7,994 |

* Requires additional filtering prior to analysis

## Trained Model: datumbox/datumbox-framework-zoo

- Pre-trained models for Datumbox Machine Learning Framework. [http://www.datumbox.com/](http://www.datumbox.com/)

**Important Notes:**

- The models support only English.
- The binary files should be loaded using their corresponding Framework version.
- All the models should be loaded using the InMemory storage engine.
- Within the folder of each model you will find a stats.txt file which contains the accuracy metrics of the classifier. The metrics were estimated using 10-fold cross validation.
- All the remaining API methods which are not included here (Readability Assessment, Keyword Extraction, Text Extraction & Document Similarity) are directly powered up by standalone classes of the framework.

Language

Runtime Dependencies

Accuracy Metrics

## Untrained Model: microsoft/LightGBM

- A fast, distributed, high performance gradient boosting (GBT, GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, used for ranking, classification and other machine learning tasks. It is under the umbrella of the DMTK (http://github.com/microsoft/dmtk) project of Microsoft.

LightGBM grows trees leaf-wise (best-first) [7]. It will choose the leaf with max delta loss to grow. Holding `#leaf` fixed, leaf-wise algorithms tend to achieve lower loss than level-wise algorithms.

Leaf-wise may cause over-fitting when `#data` is small, so LightGBM includes the `max_depth` parameter to limit tree depth. However, trees still grow leaf-wise even when `max_depth` is specified.

Tree maximum depth
(Algorithm-specific attribute)


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Sample Attributes Extracted from Literature Survey

**Software Engineering for Machine Learning: A Case Study**

Saleema Amershi, *Microsoft Research*, Redmond, WA USA, samershi@microsoft.com
Andrew Begel, *Microsoft Research*, Redmond, WA USA, andrew.begel@microsoft.com
Christian Bird, *Microsoft Research*, Redmond, WA USA, cbird@microsoft.com
Robert DeLine, *Microsoft Research*, Redmond, WA USA, rdeline@microsoft.com
Harald Gall, *University of Zurich*, Zurich, Switzerland, gall@ifi.uzh.ch
Ece Kamar, *Microsoft Research*, Redmond, WA USA, eckamar@microsoft.com
Nachiappan Nagappan, *Microsoft Research*, Redmond, WA USA, nachin@microsoft.com
Besmira Nushi, *Microsoft Research*, Redmond, WA USA, besmira.nushi@microsoft.com
Thomas Zimmermann, *Microsoft Research*, Redmond, WA USA, tzimmer@microsoft.com

[Amershi 2019]

*"Each model is tagged with a provenance tag that explains with which data it has been trained on and which version of the model. Each dataset is tagged with information about where it originated from and which version of the code was used to extract it (and any related features)."*

Trained Model
- Training Data
- Model Version

Data Set
- Provenance
- Data extraction code version
- Features

**Hidden Technical Debt in Machine Learning Systems**

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, Dan Dennison
{ebner,vchaudhary,mwyoung,jfcrespo,dennison}@google.com
Google, Inc.

[Sculley 2015]

*We have access to some of these authors*

*"A second possible strategy is to focus on detecting changes in prediction behavior as they occur. One such method was proposed in [12], in which a high dimensional visualization tool was used to allow researchers to quickly see effects across many dimensions and slicings. Metrics that operate on a slice by-slice basis may also be extremely useful."*

Accuracy Metrics

*"Oftentimes, a prediction from a machine learning model $m_a$ is made widely accessible, either at runtime or by writing to files or logs that may later be consumed by other systems. Without access controls, some of these consumers may be undeclared, silently using the output of a given model as an input to another system."*

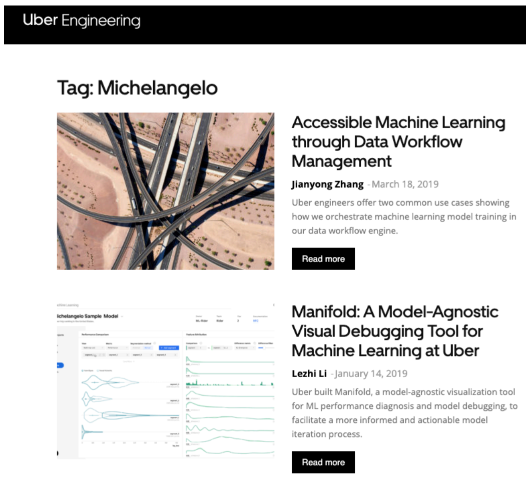
ML Software Component Consumers


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Sample Attributes Extracted from Gray Literature

https://eng.uber.com/tag/michelangelo/




https://eng.uber.com/machine-learning-data-workflow-management/

1. The **model training task** tells Michelangelo to start training using a predefined project template and the feature dataset generated by the second workflow. Once training completes, Piper attaches a model unique identifier to this training cycle that can be referenced by performance validation, model deploy, and monitoring tasks.
2. The **performance validation task** compares select metrics values such as receiver operating characteristic curve (ROC) and area under curve (AUC) with user-specified thresholds to decide whether a model is accurate enough to deploy.
3. If the model is deemed suitable, the **model deploy task** calls Michelangelo to deploy the model. The model deploy task can also deploy the same model to use different sharding configurations, such as those specific to cities where Uber operates.
4. Finally, a **monitoring task** is typically added to collect serving metrics such as ROC and AUC, comparing them with their training equivalents and continuously monitoring model performances.

Model Unique Identifier

Accuracy Metrics

Search String: trained model "machine learning" description


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.

# Descriptor(s) Example(s)

Sample descriptor created from a text example in [Mitchell 2019]
- Not representative of our research results
- Example in paper is not machine readable (i.e., in XML)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<descriptor name="TrainedModel">
    <model-details>
        <name>Smiling Detection in Images</name>
        <developer>Google and the University of Toronto</developer>
        <date>2018</date>
        <version>v1</version>
        <type>Convolutional Neural Net</type>
        <description>Pre-trained for face recognition then fine-tuned with cross-entropy loss for binary smiling classification</description>
    </model-details>
    <intended_use>
        <primary-intended-uses>
            <primary-use>Fun applications, such as creating cartoon smiles on real images</primary-use>
            <primary-use>Augmentative applications, such as providing details for people who are blind</primary-use>
            <primary-use>Assisting applications such as automatically finding smiling photos</primary-use>
        </primary-intended-uses>
        <primary-intended-users>
            <user>Younger audiences</user>
        </primary-intended-users>
        <out-of-scope-uses>
            <out-of-scope-use>Emotion detection or determining affect</out-of-scope-use>
            <out-of-scope-use>Smiles were annotated based on physical appearance, and not underlying emotions</out-of-scope-use>
        </out-of-scope-uses>
    </intended_use>
    <factors>
        <groups>
            <group>gender</group>
            <group>age</group>
            <group>race</group>
            <group>Fitzpatrick skin type</group>
        </groups>
        …
</descriptor>
```


© 2019 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.