RESEARCH REVIEW 2022

Carnegie Mellon University
Software Engineering Institute

Automating Mismatch Detection and Testing in ML Systems

NOVEMBER 14, 2022

Grace Lewis
Principal Researcher and TAS Initiative Lead

Carnegie Mellon University Software Engineering Institute

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.

DM22-0808

Organizations Struggle with Moving Machine Learning (ML) Components into Production Systems

Challenges include the following:

- ML component performs poorly because model training data is different from production data.
- Large amounts of glue code need to be developed because ML component input/output does not match with system components.
- Production environments and tools are not set up to detect model problems or to collect the necessary data for model troubleshooting and retraining.
- Systems perform poorly (or are unable to deploy) because available computing resources are insufficient to support model inference requirements.
- Organizations acquire ML components that they do not know how to test properly.

Different Teams and Workflows

Data Scientists, ML Engineers

Software Engineers

Operations Staff

The development of ML-enabled systems* typically involves three separate activities and workflows.

- model development
- model integration and testing
- model operation

… performed by three different and separate teams

- data science or ML engineering
- software engineering
- operations

… and often no systems context.

* We define an ML-enabled system (or ML system for short) as a software system that includes one or more machine learning (ML) components.

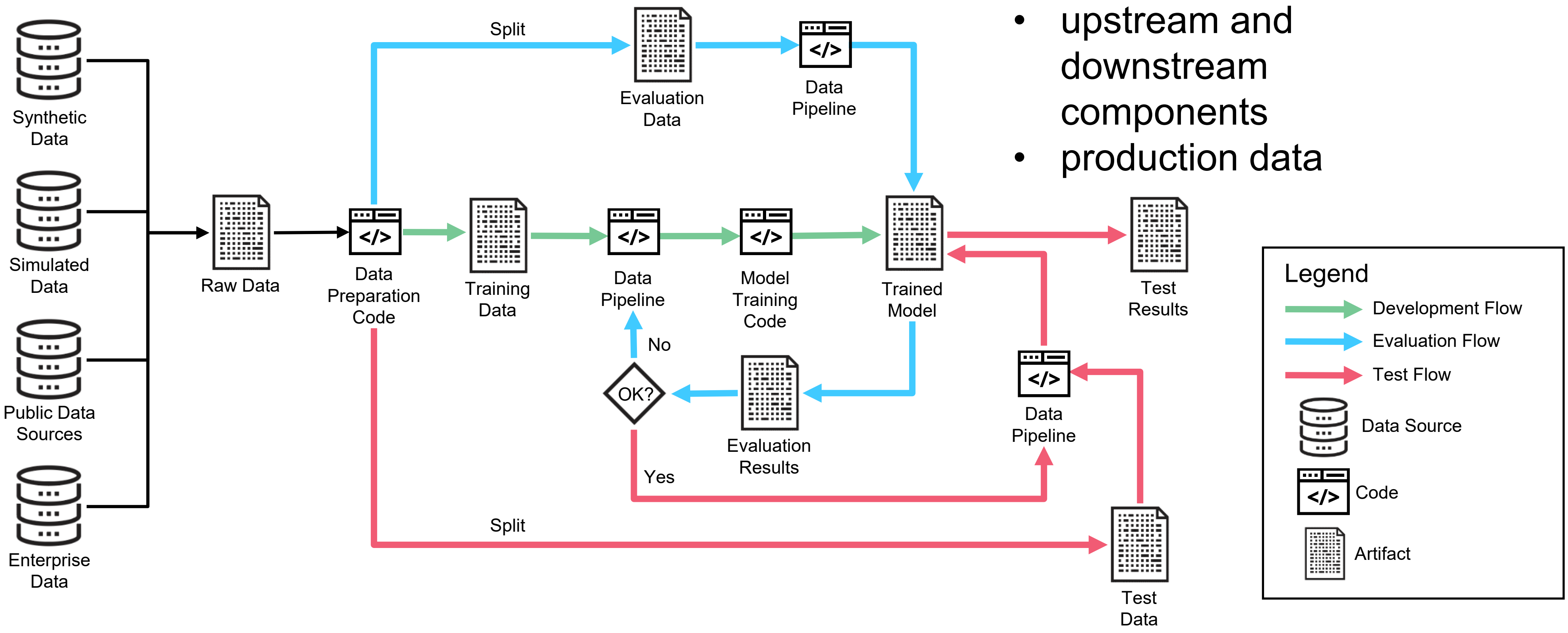
Model Development Workflow

Examples of information required to make better decisions:

- system context
- upstream and downstream components
- production data

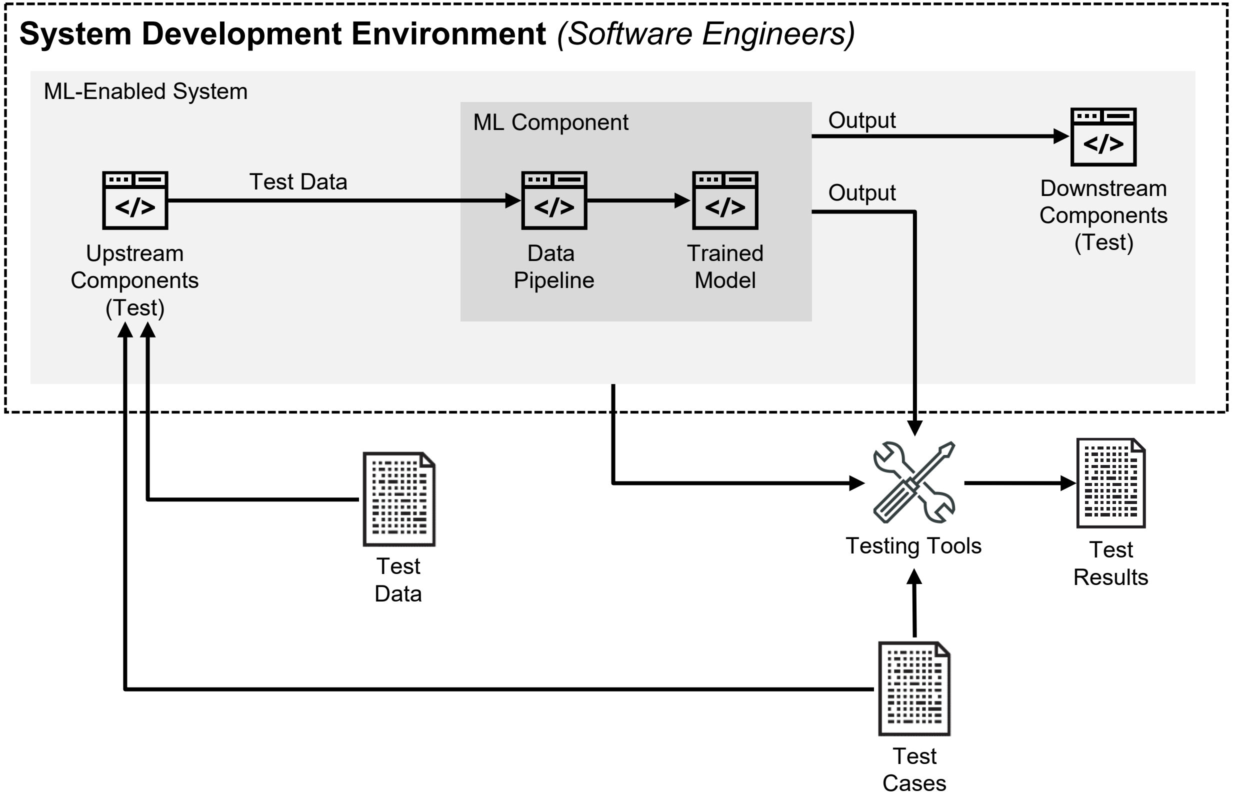
Carnegie Mellon University Software Engineering Institute

Model Development Environment *(Data Scientists, ML Engineers)*



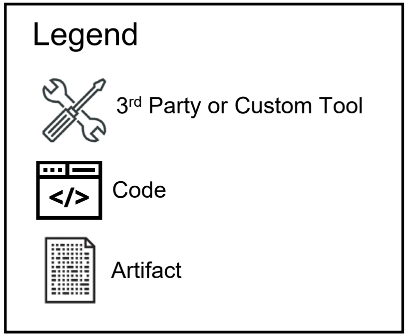
Carnegie Mellon University Software Engineering Institute

Model Integration and Testing Workflow



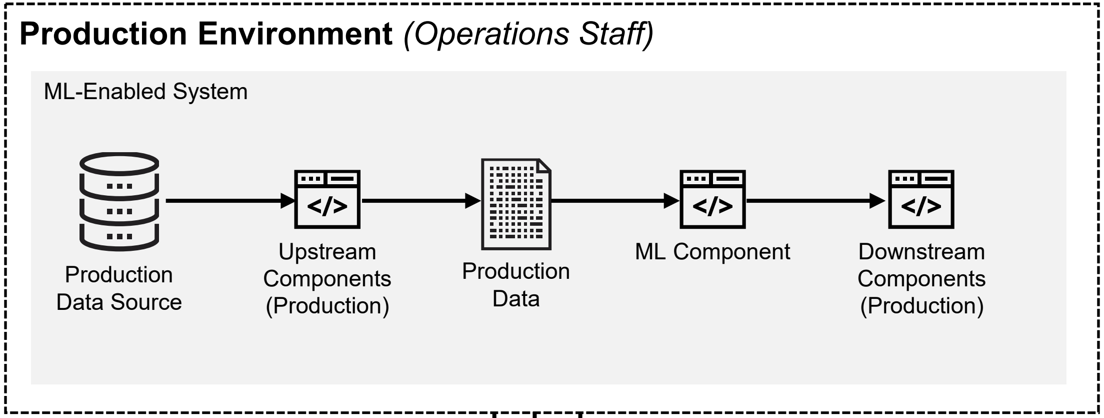
Examples of information required to make better decisions:

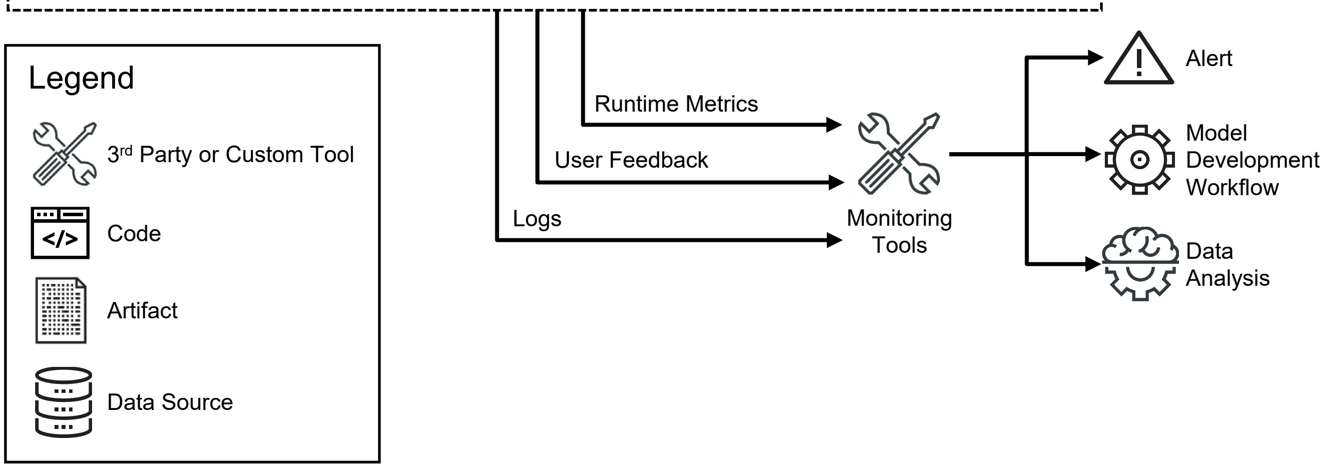
- system context
- test cases
- test data
- production environment



Carnegie Mellon University Software Engineering Institute

Model Operations Workflow





Examples of information required to make better decisions:

- system context
- monitoring requirements
- resource requirements

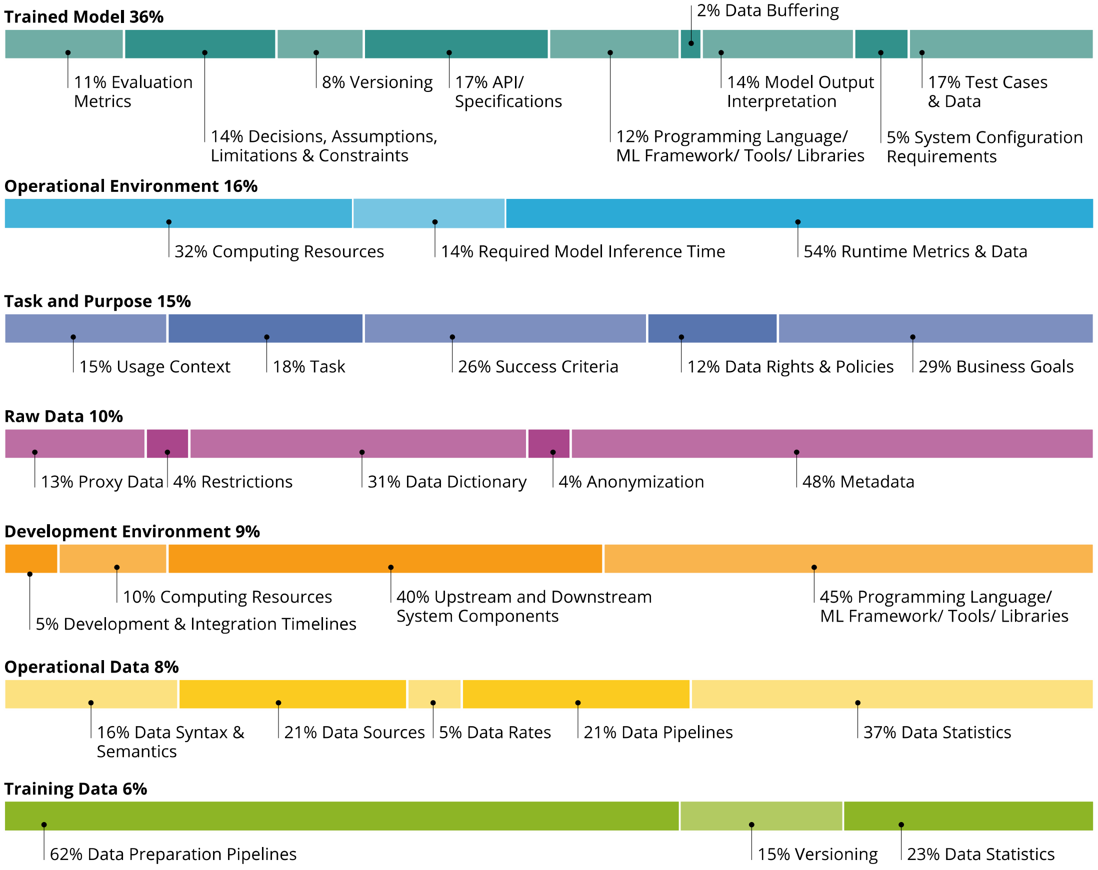
ML Mismatch

Carnegie Mellon University Software Engineering Institute

ML mismatch is a problem that occurs in the development, deployment, or operation of an ML-enabled system when different stakeholders—data scientists, ML engineers, software engineers, operations, system owners—make **incorrect assumptions** about systems elements that result in a negative consequence.

As the DoD adopts machine learning to solve mission-critical problems, the inability to detect and avoid ML mismatch creates delays, rework, and failure in the development, deployment, and evolution of ML systems.

Previous Work: Descriptors for ML Systems



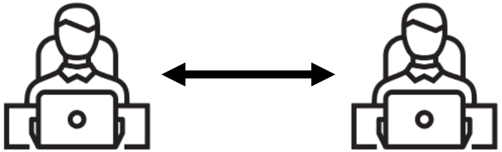
We developed a set of machine-readable descriptors (JSON schema) that define system attributes that need to be specified to avoid mismatch.

- system context
- raw data
- training data
- data pipeline
- trained model
- development environment
- production environment
- production data

Project Objectives

Develop a suite of tools to

- automate ML mismatch detection.
- demonstrate extending and using descriptors to support testing production readiness of ML components.
- validate research results with DoD systems and collaborators.
    - We are actively looking for project collaborators.

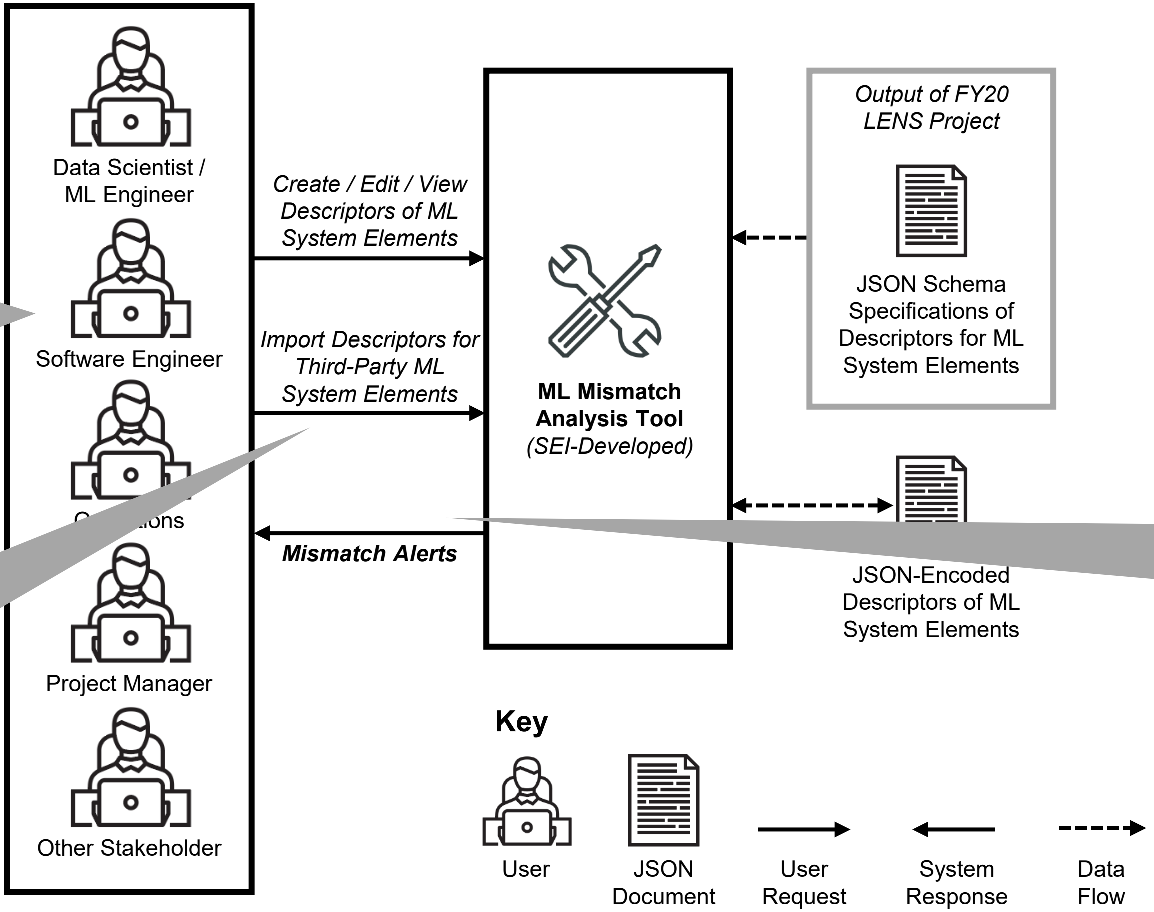


Collaborators Wanted!

Carnegie Mellon University
Software Engineering Institute

TEC: ML Mismatch Detection Tool

Stakeholders develop and share descriptors for their parts of the system and can view any descriptors.



As system elements are acquired (if applicable), descriptors are imported.

Mismatch rules encoded in the tool serve to generate alerts.

TEC: ML Mismatch Detection Tool — Demo

Carnegie Mellon University Software Engineering Institute

SEI

TEC

- Home
- My Projects
- New Project
- Resources

Home

Welcome to TEC, the ML Mismatch Detection Tool

The development and operation of ML-enabled systems involves three workflows

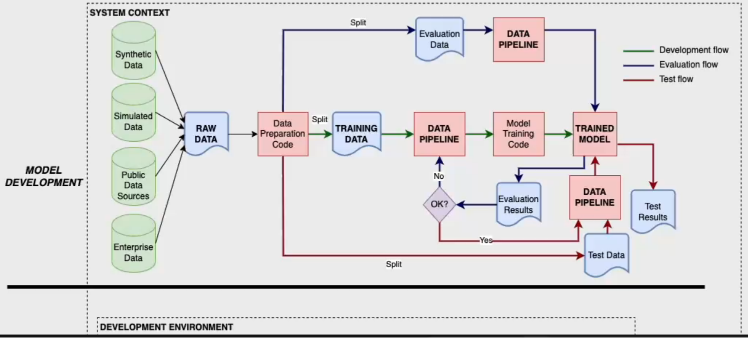
- The *Model Development* workflow produces a trained model and is typically executed by data scientists or ML engineers with a background in statistics and machine learning
- The *Model Integration* workflow takes the data pipeline and trained model produced in the previous workflow, packages them as an ML component, and integrates it into an ML-enabled system — typically executed by software engineers and developers with a background in traditional software development and testing
- The *Model Operation* workflow oversees the operation and monitoring of the production ML-enabled system — typically executed by IT personnel with a background in traditional IT operations

These workflows are often executed by three different teams with three different backgrounds, tools, and even different vocabularies, which can lead to ML Mismatch.

We define ML Mismatch as a problem that occurs in the development, integration, deployment, and operation of an ML-enabled system due to incorrect assumptions made about system elements by different stakeholders that results in a negative consequence.

TEC supports the explicit recording of these assumptions in a set of **eight descriptors**, shown in the diagram below in bold caps letters. The goal of the descriptors is to support the **model development to operations process**.

- System Context: Business goals, task to perform, success criteria, usage context, risks, and other business elements that influence model development, integration, deployment, operation, and evolution
- Raw Data: Unprocessed data sources from which training data is derived
- Data Pipeline: Code that prepares data for processing by the Trained Model
- Training Data: Data for model training
- Trained Model: Trained model to be deployed in a production ML-enabled system
- Development Environment: Development and computing environment in which the ML Component (data pipeline and trained model) will be tested and integrated into the ML-enabled system
- Production Environment: Computing environment in which the ML Component (data pipeline and model) will execute as part of an ML-enabled system
- Production Data: Data that is processed by the ML Component in production



ML Component Testing

Testing of ML components is a known challenge, especially for organizations that acquire ML components.

- Results from our ML mismatch study show that one of the top causes for mismatch is lack of information on how to test ML components.
- Results from our study on collaboration challenges in ML systems development show that teams do not exchange information required for testing, leading to problems in production.
- These two findings are consistent with published practitioner studies, which highlight the need for tools and realistic techniques for testing ML-enabled software systems.

Testing Production Readiness of ML Components

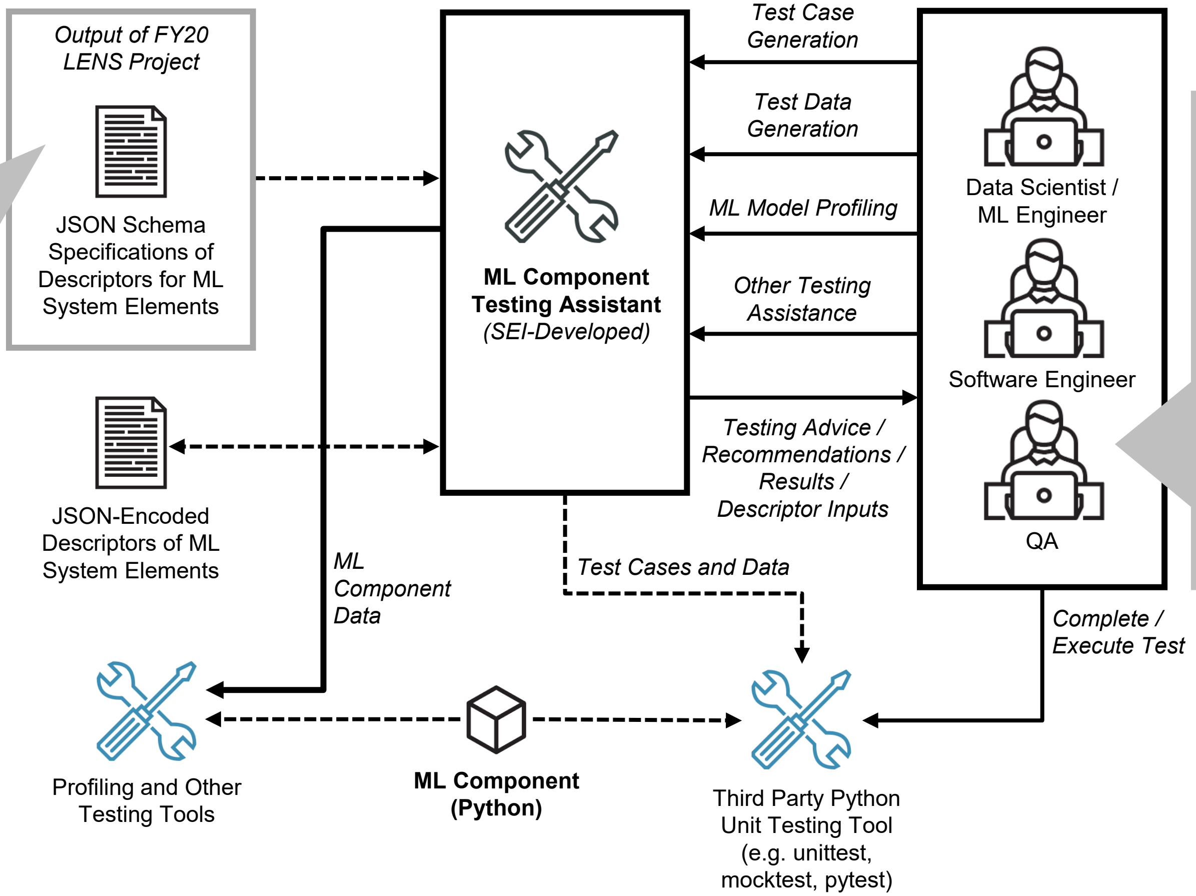
We define production readiness based on these ML component attributes:

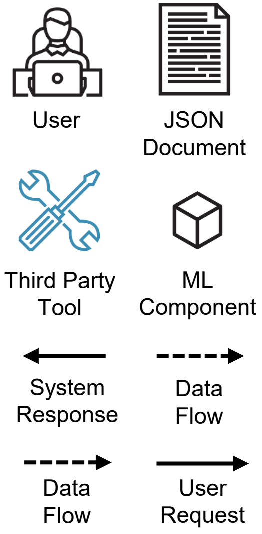
- **Ease of Integration**: ML component inputs and outputs are compatible with upstream and downstream components in the production system.
- **Testability**: The ML component provides either (1) specifications, test cases, or test data that enable testing by software developers or external QA teams or (2) evidence of testing or evidence that teams have followed best practices.
- **Monitorability**: The ML component produces information that monitoring components can use in the production system to detect potential problems.
- **Maintainability**: The ML component defines (or produces) data that teams can use for model retraining and troubleshooting.
- **Quality**: The ML component meets quality requirements.
  - model requirements (e.g., accuracy)
  - system requirements (e.g., inference time, resource consumption)

Carnegie Mellon University Software Engineering Institute

ML Component Testing Assistant

Extension of descriptors to support testing for **production readiness** of ML components





Initial testing capabilities targeted at acquisition organizations or QA teams as they perform **acceptance or independent testing** of ML components — where the least amount of understanding and tool support exists

Carnegie Mellon University Software Engineering Institute

Implementing and Evaluating Several Approaches to Testing for Production Readiness

Model Quality	**Completed**: tool for train-test leakage detection
Ease of Integration / Testability	In progress: black box testing of ML component capabilities
Production Readiness	In progress: evidence-based process and tool for independent testing of ML component production readiness • mapping practitioner interview data from our two studies to production-readiness attributes • mapping existing SE practices and tools presented as ML testing tools to production-readiness attributes • refining definition of production readiness based on findings and collaborator discussions

Train-Test-Leakage Detection

Trust in models developed by external teams was identified as a collaboration challenge for organizations building ML systems.

- Model developers often report high performance during test and evaluation because models are overfit—inadvertently or intentionally.
- A cause for overfitting is train-test leakage—leaking information about test data into training data.

Developed a static analysis tool that uses data flow analysis and pointer analysis for detection of the following three types of train-test leakage:

- **Overlap**: Test data is directly used as input for training or hyper-parameter tuning.
- **Multi-Test**: Test data is used repeatedly for evaluation.
- **Preprocessing**: Test data and training data are preprocessed together (e.g., normalization, feature selection, vectorization).

Looking for Collaborators

Carnegie Mellon University Software Engineering Institute

As an applied R&D organization, we need collaborators to inform and validate our research through participation in different types of activities.

Near-term, we are looking for

- organizations willing to use and evaluate the automated mismatch detection tool and provide feedback.
- organizations or teams (e.g., DT&E, OT&E) tasked with testing ML components (or ML systems) developed by other organizations to
  - participate in discussions related to definition of production readiness and solution brainstorming.
  - evaluate and use developed processes and tools and provide feedback.

Longer term, we are looking for organizations to integrate SEI-developed approaches into their ML system development and testing processes and tool chains.

Team

SEI Team

Contact us at
info@sei.cmu.edu

Grace A. Lewis, Ph.D. (PI)
Principal Researcher and TAS Initiative Lead

Rachel Brower-Sinning, Ph.D.
Machine Learning Research Scientist

Alex Derr
Associate Software Engineer

Collaborators

Christian Kästner, Ph.D.
Associate Professor
Carnegie Mellon University
School of Computer Science

Chenyang Yang
Ph.D. Student
Carnegie Mellon University
School of Computer Science

Publications

ML Mismatch Study: Lewis, G. A., Bellomo, S., & Ozkaya, I. (2021, May). Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. In 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN) (pp. 133-140). IEEE. https://arxiv.org/abs/2103.14101

Collaboration Challenges Study: Nahar, N., Zhou, S., Lewis, G., & Kästner, C. (2022). Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. Proceedings of the 44th International Conference on Software Engineering (ICSE 2022). https://arxiv.org/abs/2110.10234

Train-Test Leakage Detection: Yang, C., Brower-Sinning, R., Lewis, G. A., Kästner, C.(2022). Data Leakage in Notebooks: Static Detection and Better Processes. Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE 2022). https://arxiv.org/abs/2209.03345