What is Really Different in Engineering AI-Enabled Systems?

Dr. Ipek Ozkaya

Technical Director, Engineering Intelligent Software Systems

Software Solutions Division

Carnegie Mellon University Software Engineering Institute

ozkaya@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

DM21-1048

Agenda

CMU and SEI Overview

National Agenda for Software Engineering

Foundational selected AI practices

- Characterizing and detecting mismatch in ML-enabled systems
- Software architecture for ML-enabled systems
- Role of MLOps in continuous monitoring and evolution of ML-enabled systems

Misconceptions for AI systems

What can we do today?

About me


Istanbul, Turkey


Managing Technical Debt
Reducing Friction in Software Development
AI Engineering: 11 Foundational Practices


Software Engineering Institute


Pittsburgh, PA


Software
Software
Software
AUTOMATIC PROGRAM REPAIR
Software
Software
Software

Carnegie Mellon University Software Engineering Institute

CMU – Global Research University

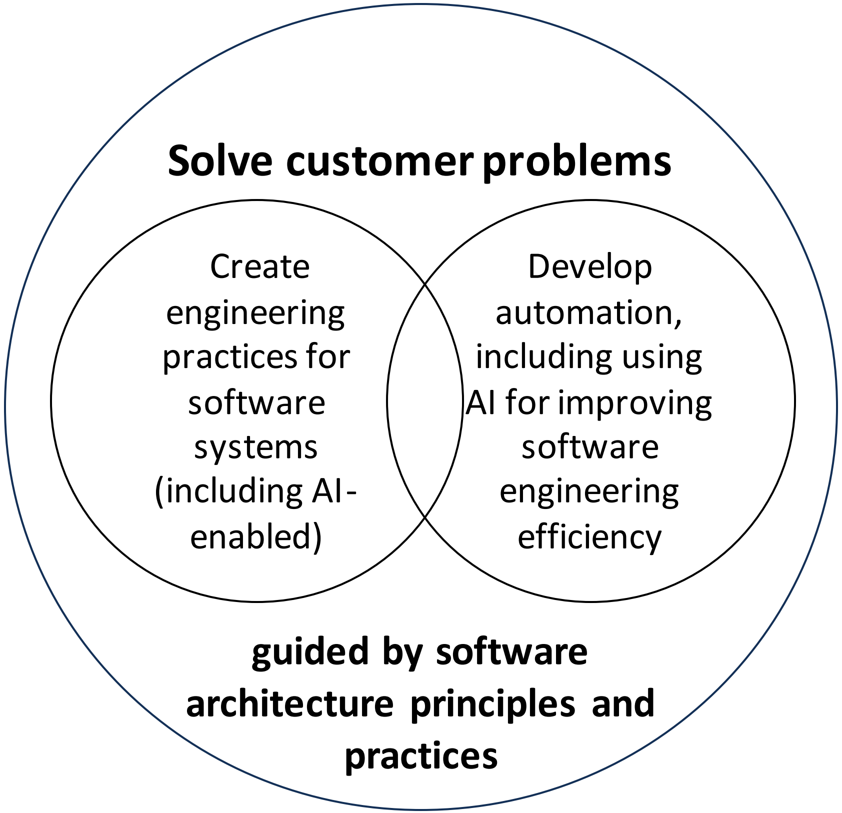
- CMU challenges the curious and passionate to imagine and deliver work that matters
- 1,442 total faculty, 13,285 students, 130 research centers
- Ranked #17 U.S. university, #1 for Computer Science, #4 for College of Engineering[1]
- Main campus and research centers in Pittsburgh, PA; Silicon Valley, CA; and Doha, Qatar

CMU – Software Engineering Institute

- Founded in 1984 as a DoD R&D Federally Funded Research and Development Center
- Focused on software, cyber, and AI
- 730 employees
- HQ in Pittsburgh, PA; other offices in DC and CA
- ~$145M annual funding / ~$21M DoD (USD R&E) 6.2 and 6.3 Line funding

[1] 1 2018 *US News and World Report*

Engineering Intelligent Software Systems – 1



A team of 26 engineers, researchers, data scientists

We develop and apply range of techniques and practices applicable at different points in the software development lifecycle.

- Domains of expertise include IT, C2, tactical, avionics, and health informatics
- Technology expertise includes IoT, big data, digital twin, cloud, and machine learning

Engineering Intelligent Software Systems – 2

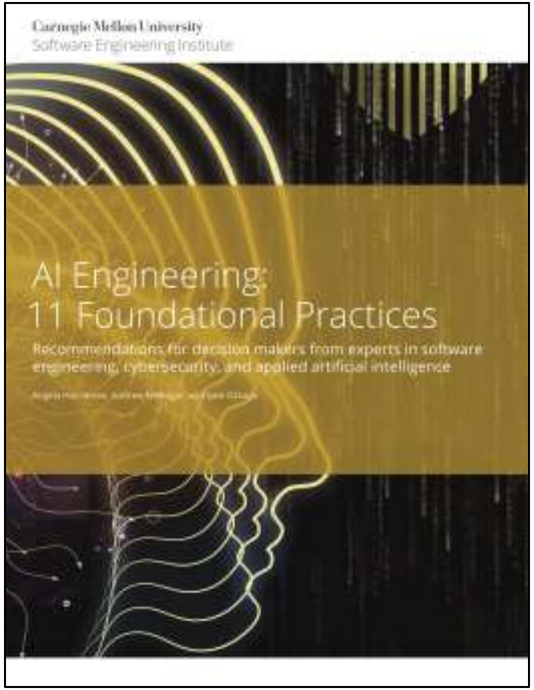
10+ courses in software architecture, technical debt, big data, available in a mix of public, on-site, and eLearning options

Educator’s Workshop every year to give back to the community.

https://resources.sei.cmu.edu/news-events/events/software-engineering-workshop/

The SEI Pearson Addison-Wesley Series on Software Architecture

AI-enabled systems are software systems!



A. Horneman, A. Mellinger, I. Ozkaya. *AI Engineering: 11 Foundational Practices*. Pittsburgh: Carnegie Mellon University Software Engineering Institute, 2019.

An **AI-enabled system** is a **software system** with one or more **AI component(s)** that need to be developed, deployed, and sustained along with the other software and hardware elements of the system.

- **Disciplined software engineering** and **cybersecurity practices** are essential starting points in adopting AI.
- The interaction between **software, data, and AI components** (e.g., ML models) creates unique challenges and requires software design and architecture approaches to be incorporated early and continuously.

SEI National Agenda for Software Engineering

Led by Anita Carlton, SEI SSD Division Director
https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=741193

Developed in collaboration with industry, government and the software engineering research community, in close collaboration with a diverse advisory board:

- Deb Frinkle, Oak Ridge National Lab (chair)
- Sara Manning Dawson, Microsoft
- Yolanda Gil, Unv. of Southern California
- Vint Cert, Google
- Penny Compton, Lockheed Martin
- Tim McBride, Zonic Labs
- Michael McQuade, CMU VP for Research
- Nancy Pendleton, Boeing
- Tim Dare, Booz Allen
- William Scherlis, DARPA

Emerging Vision of the Future of Software Engineering

The current notion of software development will be replaced by one where **the software pipeline consists of humans & AI as trustworthy collaborators that rapidly evolve systems based on programmer intent.**

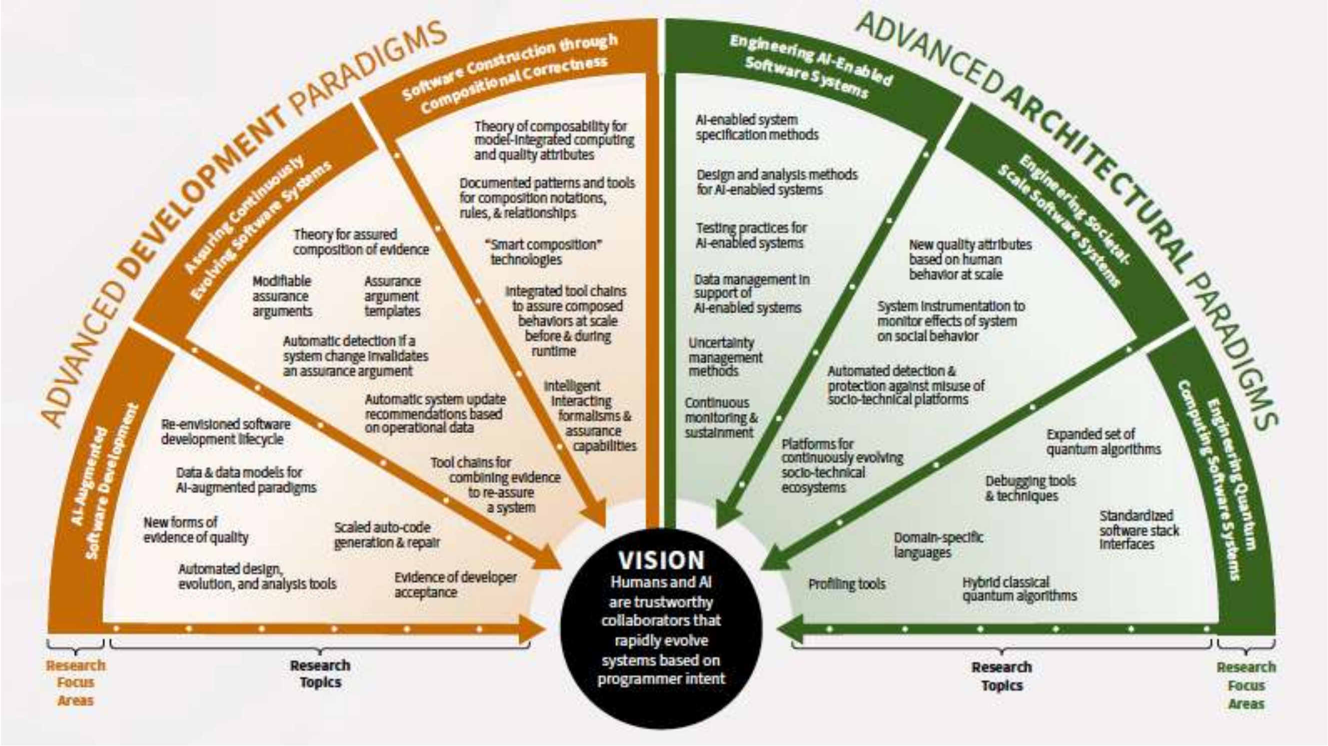
Advanced development paradigms lead to efficiency & trust at scale.

- Humans leverage trusted AI as a workforce multiplier for all aspects of software creation & sustainment.
- Formal assurance arguments are combined & analyzed to assure & efficiently (re)assure continuously evolving software.
- Enhanced software composition mechanisms enable predictable construction of systems at increasingly large scale.

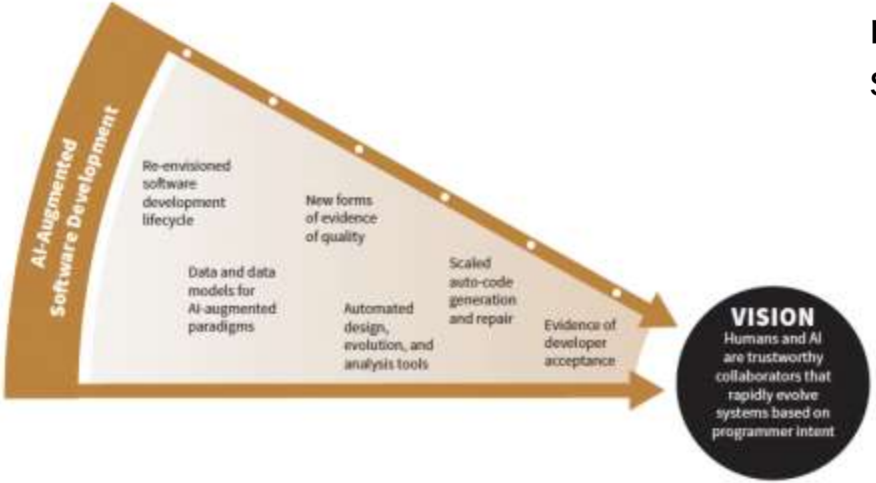
Advanced architectural paradigms enable the predictable use of new computational models.

- Theories & techniques drawn from social sciences are used to design large-scale socio-technical systems, yielding more predictable outcomes.
- AI & non-AI components interact in predictable ways to achieve enhanced mission, societal, & business goals.
- New analysis & design methods facilitate the development of quantum-enabled systems.

Research Focus Areas



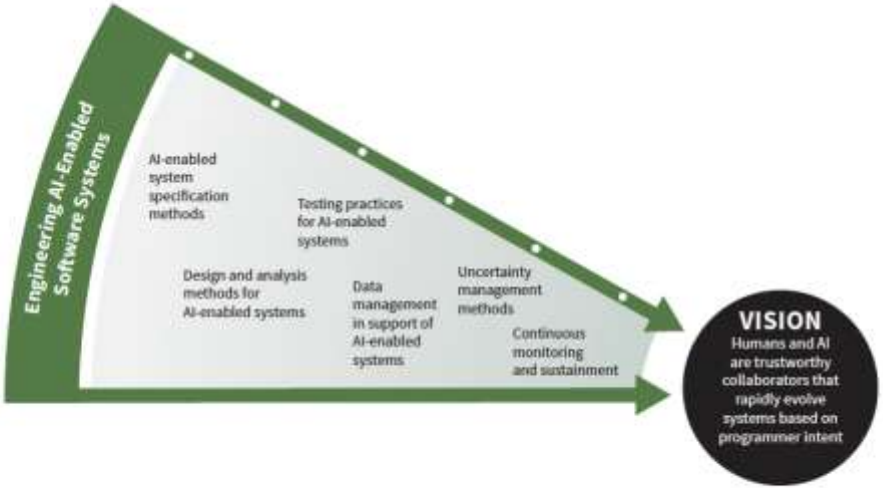
AI-Augmented Software Development



AI4SE has become an umbrella term to refer to research that uses AI approaches to tackle software engineering challenges.

- There is already progress in improving developer tools to eliminate subtle mistakes that later become hard to detect and propagate fixes for.
  - e.g. Github Copilot by Microsoft, "AI pair programmer"
- Availability of appropriate data sets is a critical barrier
  - e.g. Project Codenet by IBM (https://arxiv.org/abs/2105.12655)

Engineering AI-Enabled Software Systems



Advances in ML algorithms and the increasing availability of computational power are already resulting in huge investments in systems that aspire to exploit AI.

- Application of software engineering to AI problems
- Reinvigoration of data architecting
- Development of the new discipline of AI engineering will drive progress

Studies increasingly are all emphasizing the disconnect between ML model development and operations of systems in the field (Lwakatare 2019, Serban 2020, Giray 2021)

SEI Pillars of Work in AI Engineering

AI Engineering is a field of research and practice that combines the principles of systems engineering, software engineering, computer science, and human-centered design to create AI systems in accordance with human needs for mission outcomes.

Human-centered AI

how AI systems are designed to align with humans, their behaviors, and their values

Scalable AI

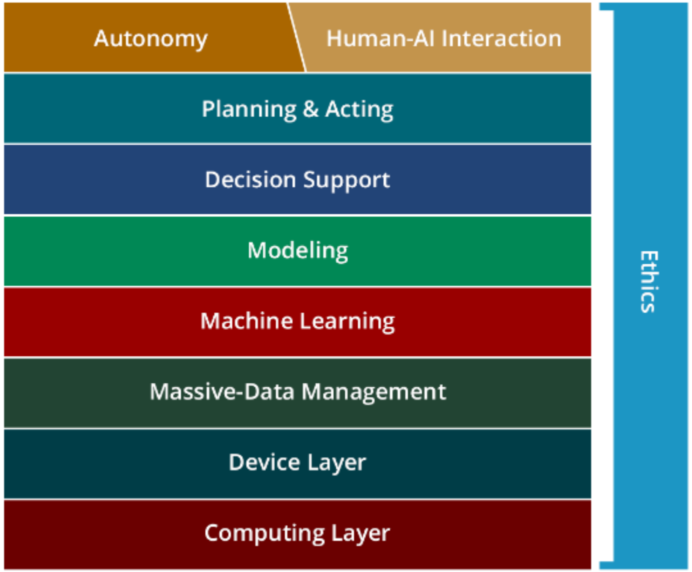
how AI infrastructure, data, and models may be reused across problem domains and deployments.

Robust and Secure AI

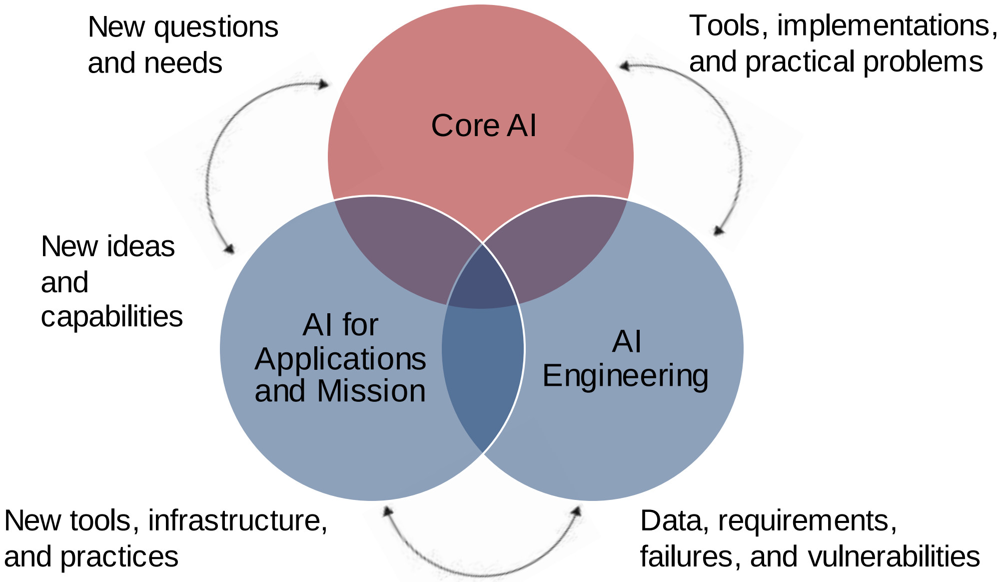
how we develop and test resilient AI systems.

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=735452

AI at CMU and AI at the SEI



CMU AI Stack*

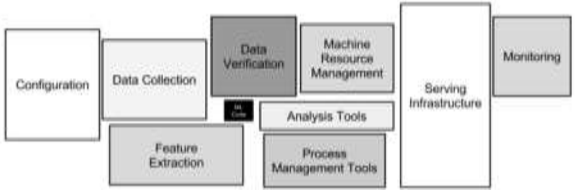


AI at the SEI

* A. W. Moore, M. Hebert, S. Shaneman, "The AI stack: a blueprint for developing and deploying artificial intelligence," Proc. SPIE 10635, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX, 106350C (4 May 2018); https://doi.org/10.1117/12.2309483

Predictable Design and Analysis of AI-Enabled Systems Rely on Software Engin

What changes are induced with maintenance and evolution of ML models?



What are ML components' architectural dependencies? What are driving patterns?

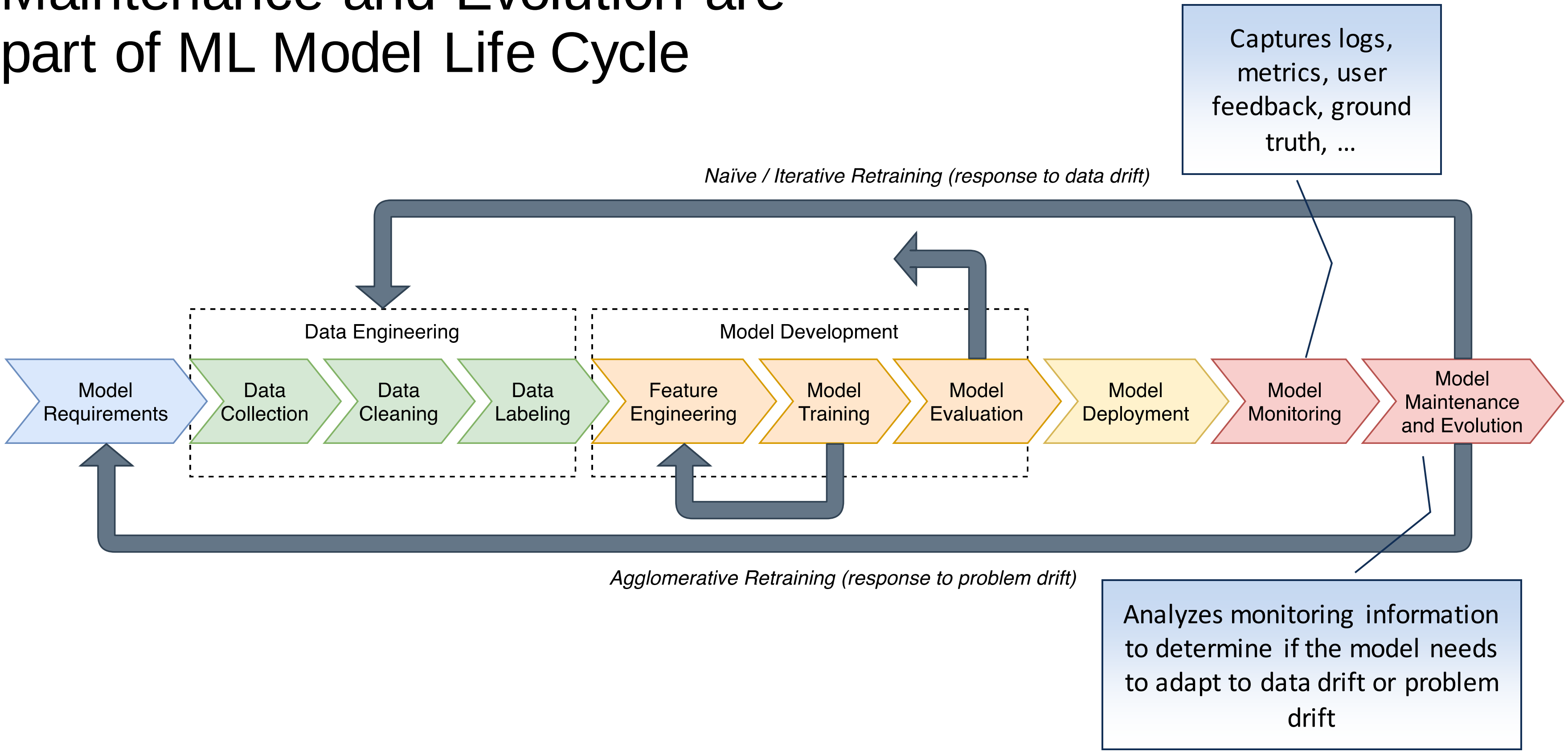
How to model and analyze high-priority quality attributes of AI-enabled systems

How can different aspects of monitorability inform ML-enabled system evolution?

How can we model for changing anything changes everything principle?

How can the essential but separate AI-enabled co-architecting and co-versioning needs be managed?

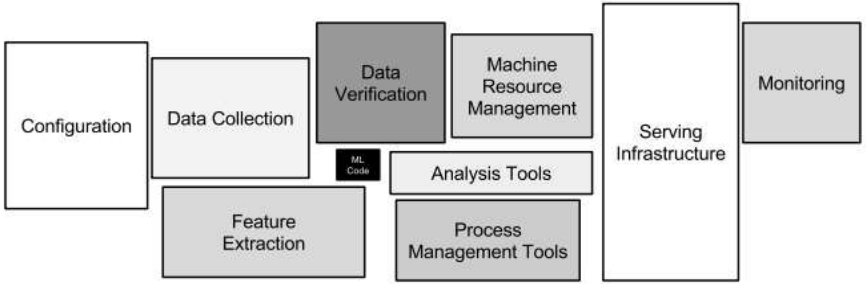
Maintenance and Evolution are part of ML Model Life Cycle



Source: Adapted from *S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software Engineering for Machine Learning: A Case Study. In2 019 IEEE/ACM 41st ICSE-SEIP. IEEE, 2019*

Systems Perspective is Essential for AI Systems

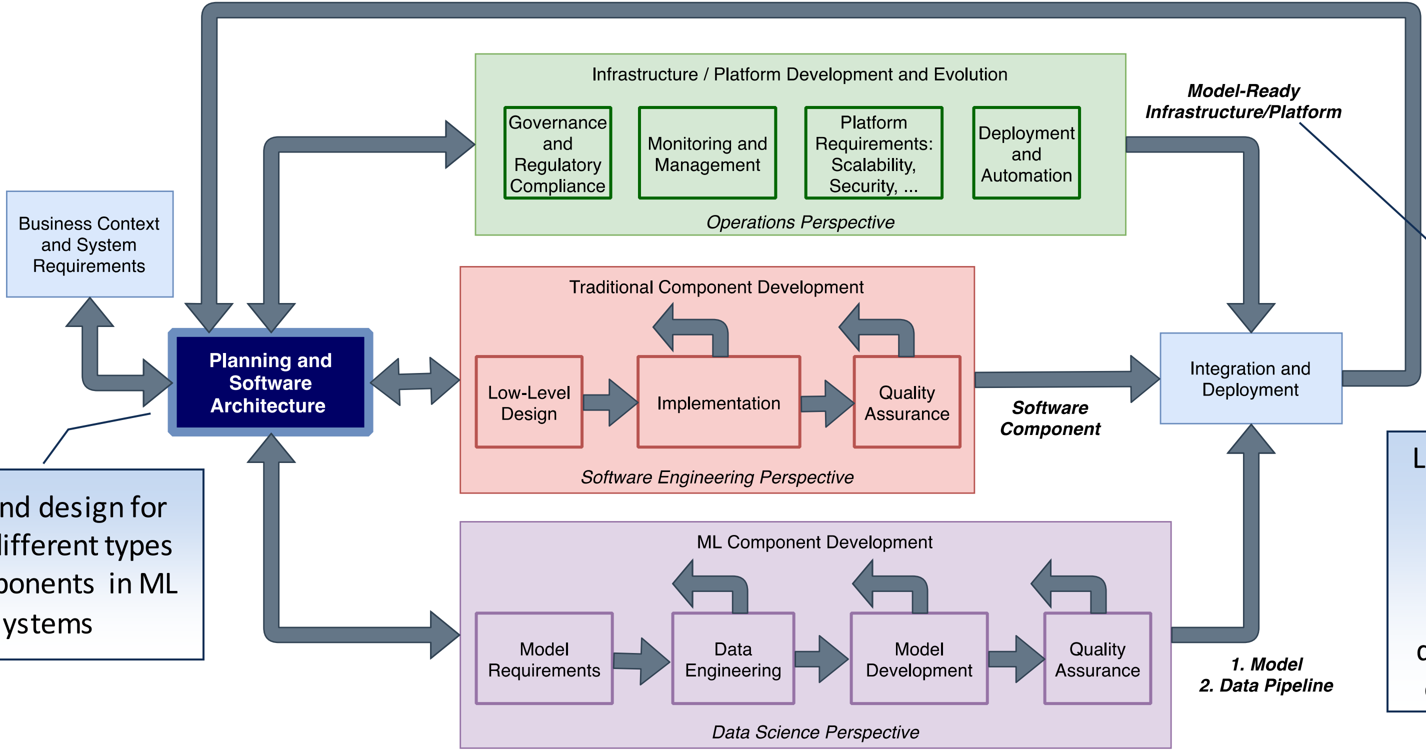
Failing to elicit, design for, and sustain the vast amount of other software components that AI components need to interact with results in not architecting the systems appropriately and failed AI system development and deployment.



"Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex." [Sculley 2015]

Source: Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. In Advances in neural information processing systems (pp. 2503-2511). http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

Manage Architectural Dependencies of AI Components



Plan and design for three different types of components in ML systems

Largest difference is the necessary reliance on monitoring to account for data-dependent behavior of ML components.

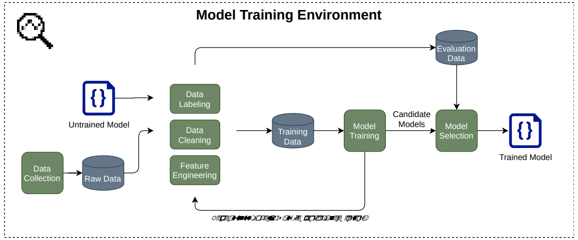
Source: Adapted from *"On the Process for Building Software with ML Components"* available at https://ckaestne.medium.com/on-the-process-for-building-software-with-ml-components-c54bdb86db24

Characterizing and Detecting Mismatch in ML-Enabled Systems

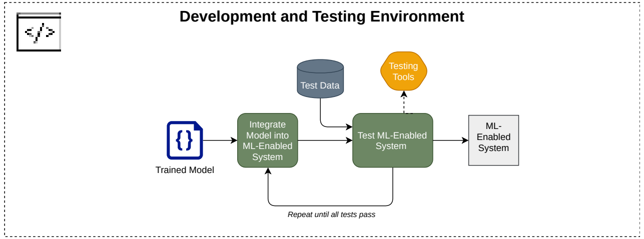
Grace Lewis, Stephany Bellomo, Ipek Ozkaya

Problem: Multiple Perspectives

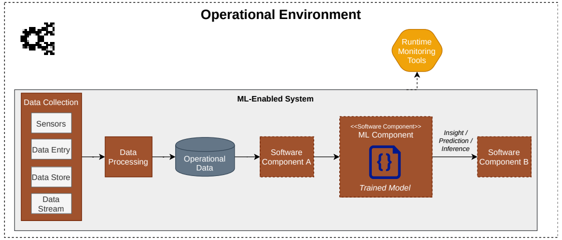
Data Scientist Perspective



Software Engineer Perspective



Operations Perspective



ML-enabled systems typically involve three different and separate workflows

- Model training
- Model integration and testing
- Model operation

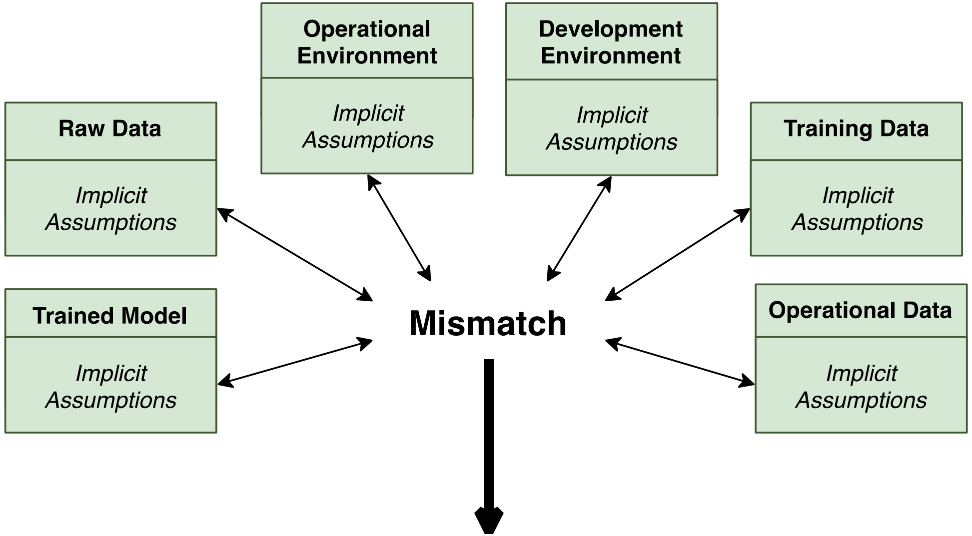
… performed by three different sets of stakeholders …

- Data scientists / ML engineers
- Software engineers
- Operations staff

… with three different perspectives

Grace A. Lewis, Stephany Bellomo, Ipek Ozkaya: **Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems.** WAIN@ICSE 2021: 133-140

Problem: Mismatch between Assumptions made by each Perspective



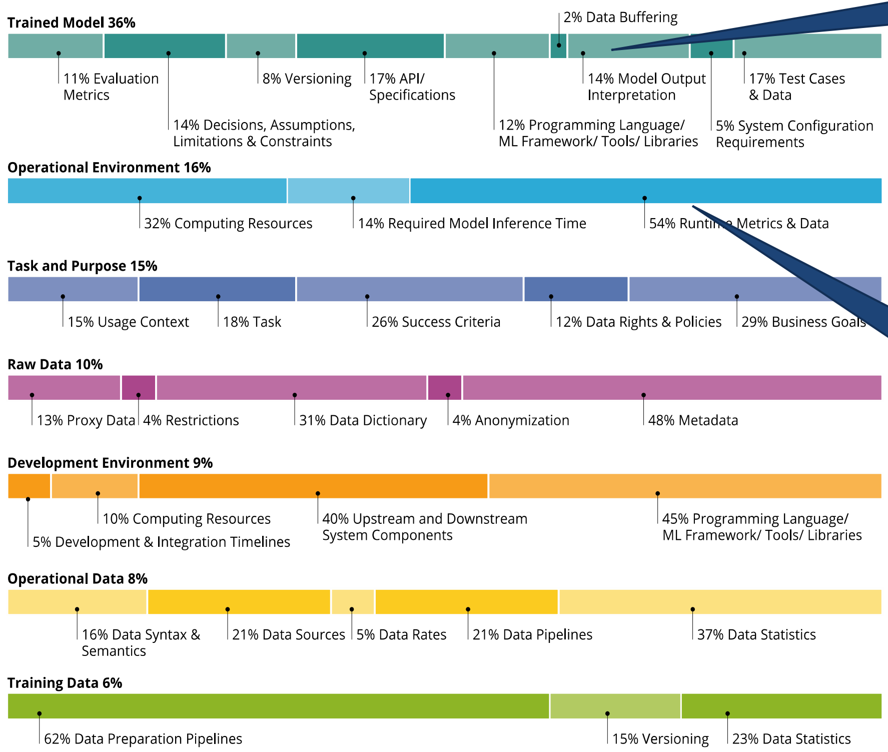
Late discovery of mismatch results in

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

We define an **ML mismatch** as a problem that occurs in the development, 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.

We also posit that ML mismatch can be traced back to information that could have been shared between stakeholders that would have avoided the problem.

Characterizing and Detecting ML Mismatch



Test cases & data mismatches make up the majority of the observed challenges (monitoring, component dependencies)

interviews to identify

- examples and consequences of mismatch
- information that should be shared between system stakeholders in order to avoid that mismatch

missing information into 7 and 34 system attributes

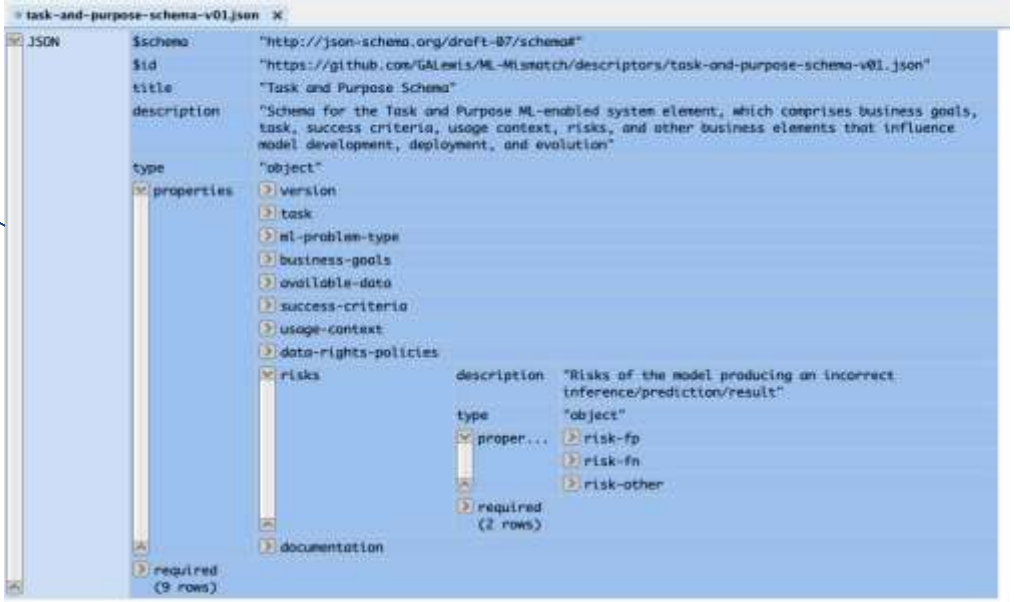
Operational Environment mismatches include poor system performance because computing resources for model testing different from operational computing resources

Study replication package and paper pre-print available at https://github.com/GALewis/ML-Mismatch

Descriptors for ML System Elements

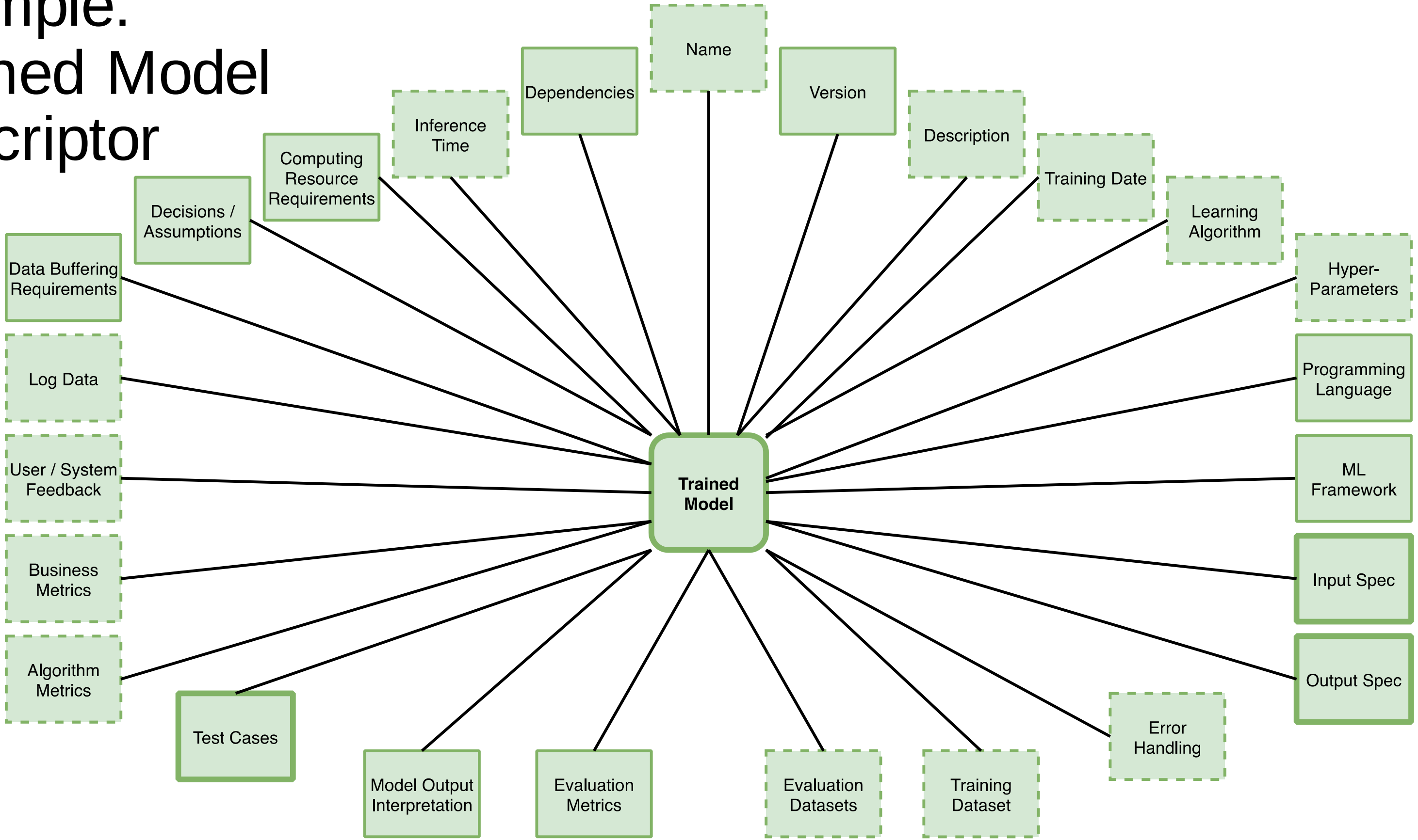
Details of mismatch examples and attributes extracted from literature review were used to develop set of seven machine-readable descriptors (JSON Schema) that define system attributes that need to be specified in order to avoid mismatch

- Task and Purpose
- Raw Data
- Training Data
- Trained Model
- Development Environment
- Production Environment*
- Production Data*



* Operational Environment and Operational Data were renamed Production Environment and Production Data, respectively, based on survey feedback

Example: Trained Model Descriptor



Bold borders indicate top attributes from interviews and surveys. Dashed borders indicate attributes added from the literature review and gap analysis.

Failures Related to Architecturally Significant Requirements

Key AI-specific concerns, when not approached with a systems perspective, create unanticipated system-level failures, e.g.

- data-dependent behavior
- shared resource dependencies
- misaligned runtime environments for AI components

L. Pons, I. Ozkaya. [Priority Quality Attributes for Engineering AI-enabled Systems.] *Association for the Advancement of Artificial Intelligence AI in Public Sector Workshop.* Washington, DC, November 7-9, 2019.

Quality attributes drive software architectures

Architecture permits or precludes the *achievement of a system's desired quality attributes*. The strategies for achieving these requirements entail thinking about the structure and behavior of the system.

If you desire…	At a minimum, you need to …
high performance	minimize the frequency and volume of inter-element communication
modifiability	limit interactions between elements
security	manage and protect inter-element communication
availability	determine the properties and behaviors that elements must have and the mechanisms you will employ to address fault detection, fault prevention, and fault recovery
extensibility	limit interactions between elements, isolate data types, and abstract common services

Recommendation: Understand High-Priority Quality Attributes of ML-Enabled Systems

ML-related software design challenge	You will desire…	At a minimum, you need to …
ML components need to be designed such that attributes can be observed	monitorability	• include monitoring components to observe and manage data changes over time • identify attributes to expose
AI introduces new attack surfaces.	security	• decouple model changes from the rest of the system • build in capabilities to modify the systems to ease deploying retrained models
Tight coupling of data and models may limit implementing privacy protections.	privacy	• decouple data stores and their interactions with other systems as much as possible • isolate changes and updates to as few locations as possible
Software update cycles may not adequately address data changes and their impact.	data centricity	• ensure that uncertainty, availability, and scalability of data are key architecture drivers for system design
Output of AI components is not human interpretable.	explainability	• decouple model changes from changes to the rest of the system • introduce observability mechanisms into the system
Rate of change that impacts software and AI components can vary significantly.	sustainability	• express rate of change as an architectural concern • build in monitoring components for both the system and the AI components

Monitorability

AI components may degrade at a different rate than the rest of the system components.

Monitoring changes in data and its impact to the rest of the system adds levels of complexity for both AI components and and other system components:

- Components that are responsible for detecting, e.g. ML model performance degradation, need to be clearly identified and designed
- Components that incorporate user feedback for ground truth need to be included
- Other system monitoring components may need to be adjusted

Recommendation: Decouple Different Aspects of Monitorability

Understand what different monitoring techniques will be needed for data quality vs. model quality vs. software quality vs. service quality

Explore relationship between monitorability to self-adaptation in ML systems*

- *of* ML — ML models self-adapt to system changes (one of the goals of MLOps)
- *for* ML — ML system adapts to changes that affect quality of service (QoS)
- *by* ML — system uses ML techniques to adapt (some of this research is already happening in the self-adaptive systems community)

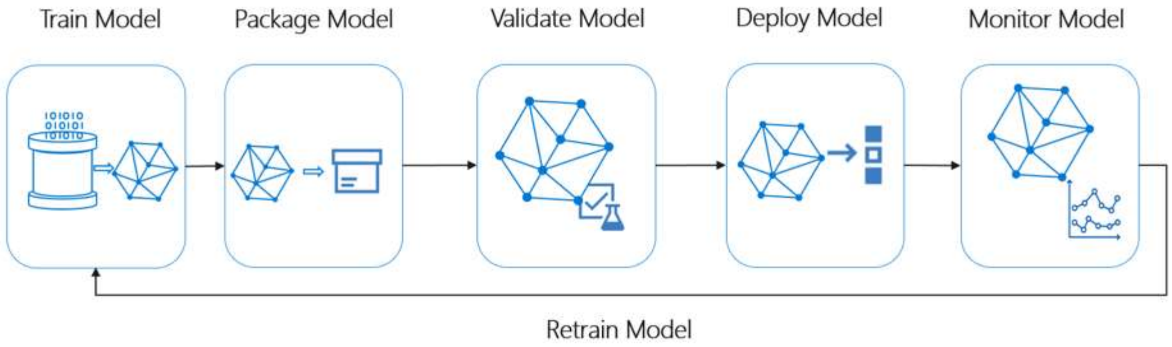
Understand how architectural elements that enable monitorability could also provide information to handle the inherent uncertainty of ML systems

* H. Muccini and K. Vaidhyanathan. Software Architecture for ML-based Systems: What Exists and What Lies Ahead. In 1st Int. Workshop on Software Engineering - AI Engineering (WAIN). IEEE, 2021.

Role of MLOps in Continuous Monitoring and Evolution of ML-Enabled Systems

Rachel Brower Sinning, Jeff Hansen, Jeff Hammet, Jeff Charabaszcs

MLOps State-of-the-Practice



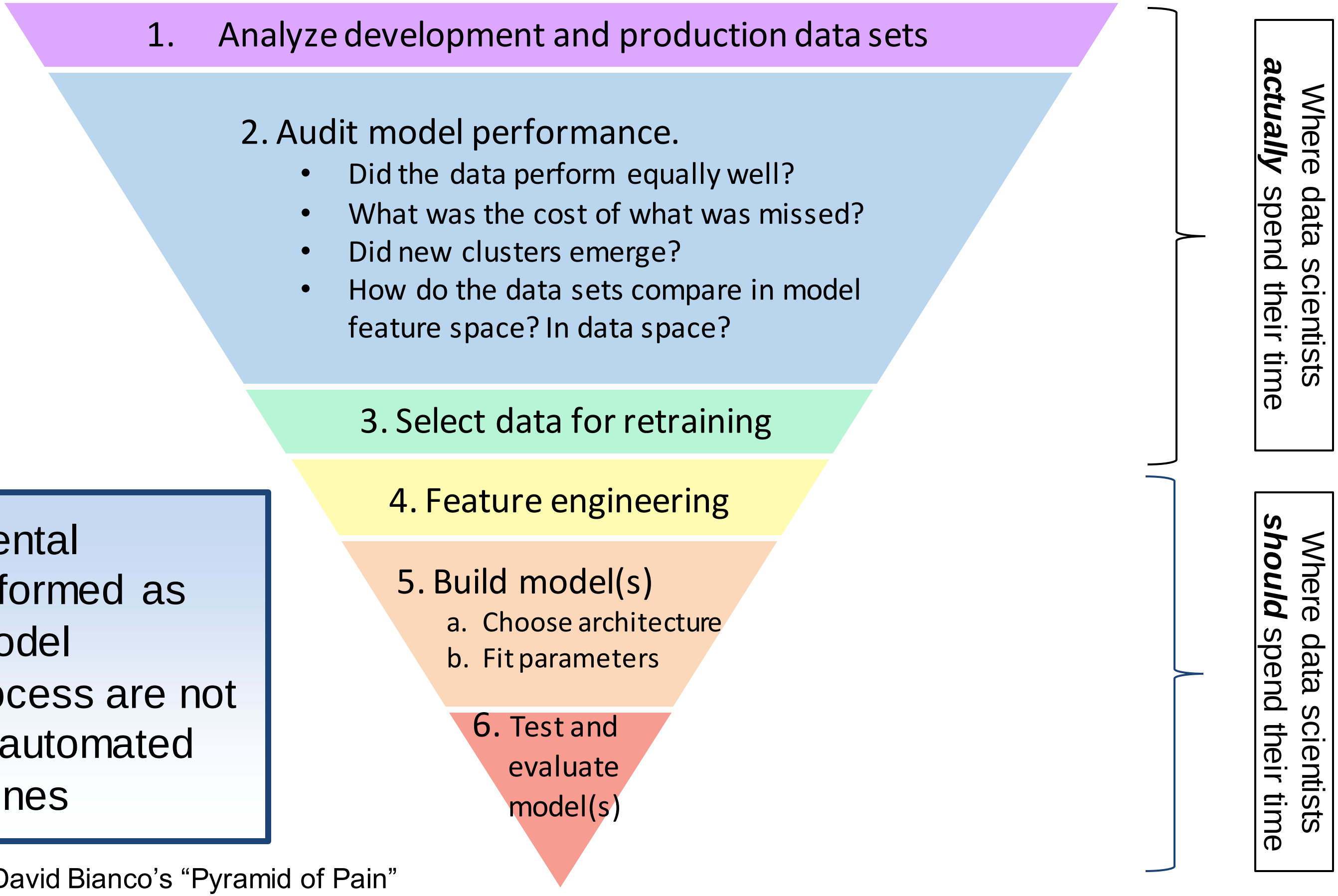
Problem:
Automated model retraining leverages only data changes — refitting of the production model to the new data

MLOps automates model deployment, but creates a model retraining problem

- Assumes new training data should be treated the same as the initial training data
- Assumes model parameters are constant and should be the same as those identified on the initial training data
- Has no information to understand why the model performed as it did
- Has no informed procedure of how to combine the production and development data set into a new training data set

Diagram Source: MS Azure MLOps Pipeline

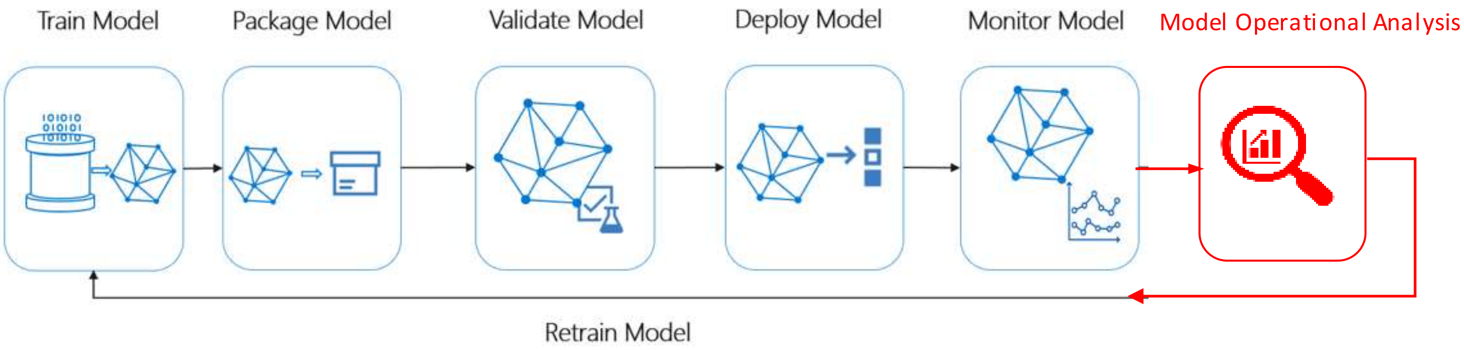
Model Retraining Process Performed by a Data Scientist



The experimental analyses performed as part of ML model retraining process are not replicated in automated MLOps pipelines

Graphic modeled after David Bianco's "Pyramid of Pain"

Solution: Integrate the analyses performed by the Data Scientist into the MLOps pipeline



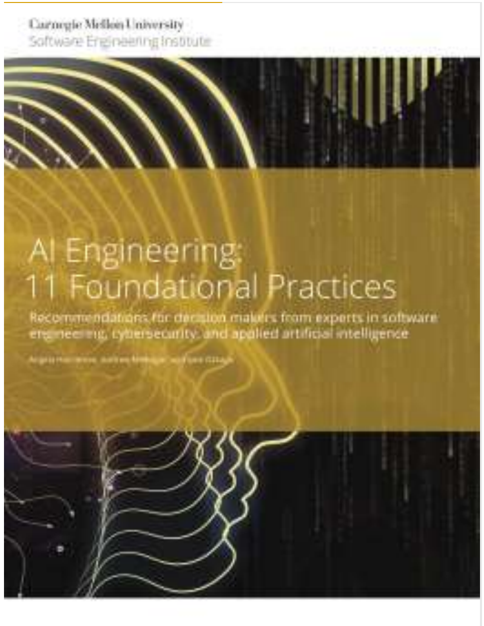
Model Operational Analysis should perform the first three steps of the model retraining process

1. [Analyze] Statistical analysis between the production data and development data
2. [Audit] Audit model performance
3. [Select] Integration of development and production data into a new development data set, with weights

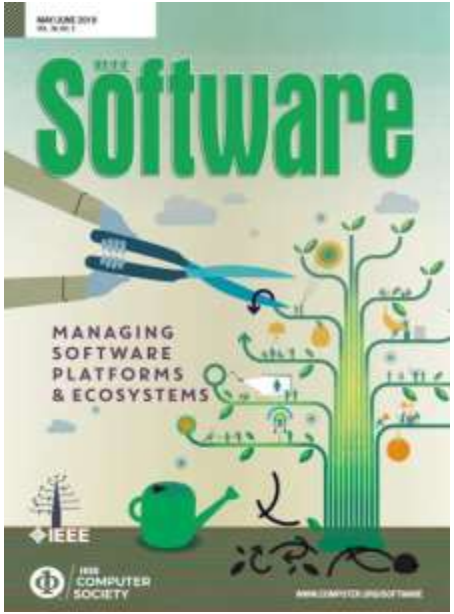
Goal is to perform informed retraining and reduce the time spent by data scientists in selecting new training data

Diagram Adapted from MS Azure MLOps Pipeline

Architecture Allows Improving Predictability of Data and Other System Component Interactions



A. Horneman, A. Mellinger, I. Ozkaya. *AI Engineering: 11 Foundational Practices*, *Sept*. 2019



I. Ozkaya. *Ethics Is a Software Design Concern.* *IEEE Softw. 36(3): 4-8 (2019).*

Take your data seriously to prevent it from consuming your project – data pipelines will require architecting.

Localize uncertainity.

Incorporate user experience and interaction to constantly validate and evolve models and architecture.

Treat ethics as both a software design consideration and a policy concern.

AI Exacerbates Existing SE Challenges

Specifying systems:

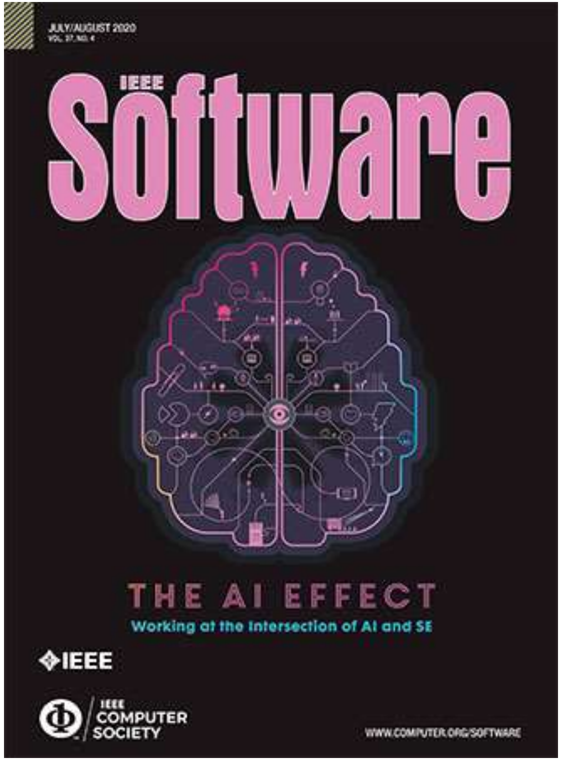
- The level of specification of AI systems depends on the level of uncertainty in the discovery process. Sometimes uncertainty is low to none.

Avoiding hidden dependencies:

- Understanding data and shared resource dependencies is not only an AI system problem, but also a software system problem.

Relying too much on frameworks and tools:

- Existing frameworks, model libraries, tools, and deployment environments help, but do not replace designing for scalability, observability, and sustainability of AI systems.



Ipek Ozkaya. *What Is Really Different in Engineering AI-Enabled Systems? IEEE Softw. 37(4): 3-6 (2020).*

References

L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic. A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation. In International Conference on Agile Software Development, pages 227–243. Springer, Cham, 2019.

A. Serban, K. van der Blom, H. Hoos, and J. Visser. Adoption and effects of software engineering best practices in machine learning. In Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2020.

S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software Engineering for Machine Learning: A Case Study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pages 291–300. IEEE, 2019.

G. Giray A software engineering perspective on engineering machine learning systems: State of the art and challenges. J. Syst. Softw. 180: 111031 (2021).

A. Horneman, A. Mellinger, I. Ozkaya. *AI Engineering: 11 Foundational Practices*. Pittsburgh: Carnegie Mellon University Software Engineering Institute, 2019.

References

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. In Advances in neural information processing systems (pp. 2503-2511).

Grace A. Lewis, Stephany Bellomo, Ipek Ozkaya: Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. WAIN@ICSE 2021: 133-140

L. Pons, I. Ozkaya. Priority Quality Attributes for Engineering AI-enabled Systems. *Association for the Advancement of Artificial Intelligence AI in Public Sector Workshop.* Washington, DC, November 7-9, 2019.

Contact Information

Ipek Ozkaya

Technical Director

Engineering Intelligent Software Systems

Software Engineering Institute

Carnegie Mellon University

ozkaya@sei.cmu.edu