Are You Ready to **Engineer** and **Sustain** AI Systems?

Dr. Ipek Ozkaya

Technical Director

Engineering Intelligent Software Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM20-0876

Agenda

CMU and SEI Overview

Foundational AI Practices: Overview and Examples

Machine Learning Mismatches

Misconceptions for AI Systems

What Can You Do Today?

Carnegie Mellon University Software Engineering Institute

CMU – Global Research University

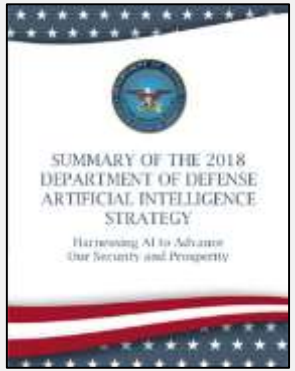
- CMU challenges the curious and passionate to imagine and deliver work that matters
- 1,442 total faculty, 13,285 students, 130 research centers
- Ranked #17 U.S. university, #1 for Computer Science, #4 for College of Engineering[1]
- Main campus and research centers in Pittsburgh, PA; Silicon Valley,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

AI-enabled systems are software systems!



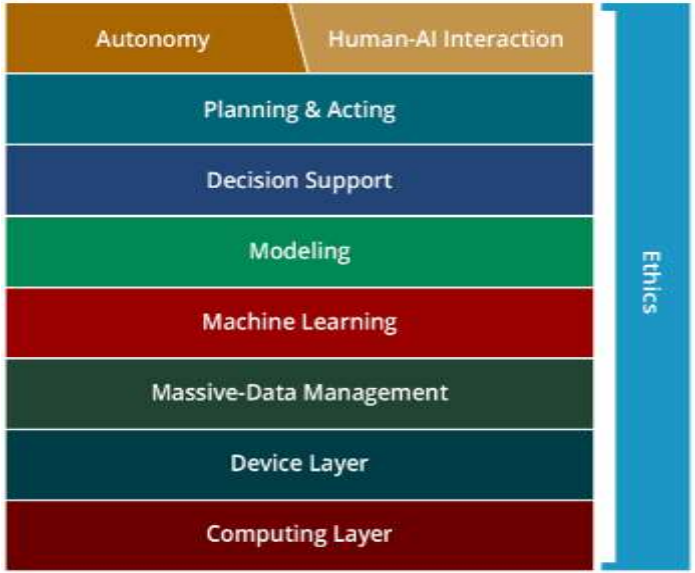
"AI refers to the ability of machines to perform tasks that normally require human intelligence – for example, recognizing patterns, learning from experience, drawing conclusions, making predictions, or taking action – whether digitally or as the smart software behind autonomous physical systems."

Source: *2018 DoD AI Strategy*

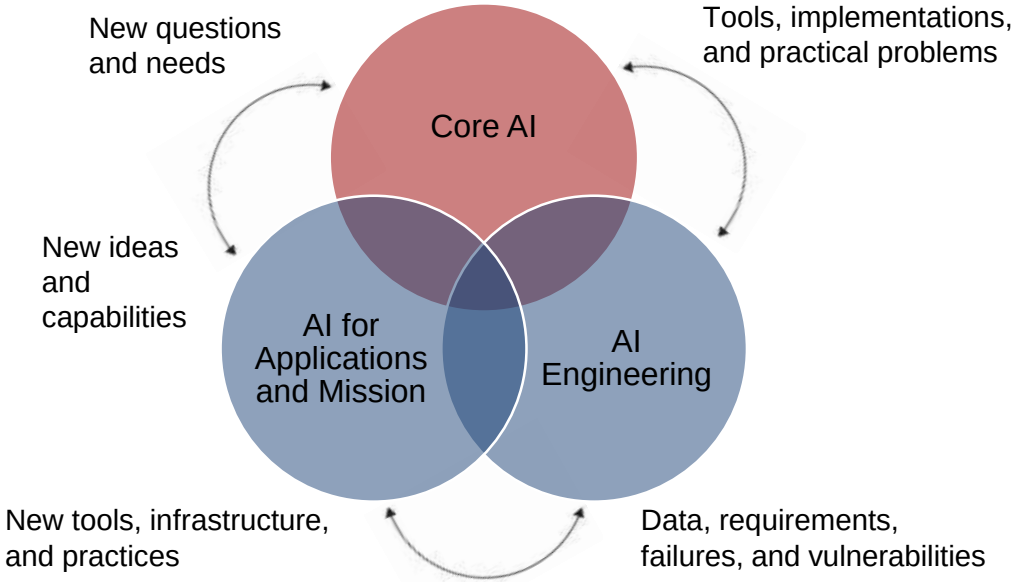
An **AI-enabled system** is a **software system** with one or more **AI component(s)** that need to be developed, deployed, and sustained along with the other software and hardware elements of the system.

- **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) requires software design and architecture approaches to be incorporated early and continuously.

AI at CMU and AI at the SEI



CMU AI Stack



AI at the SEI

AI Challenges in Government

Managing Data

- Lack of sufficient, appropriately labeled data
- Low and uneven data quality of data even when data is available (pixel, quality, metadata quality)
- Architecting data pipelines to enable incorporating rate of change in data

Evaluating and applying rapid advances to high-stakes government environments

- Knowing what approach to apply and when
- Applying heterogeneous algorithms to improve decision accuracy

Assuring AI-enabled systems

- Traceability of the recommendations to improve explainability, security, and ethics

Ensuring that the right resources are in place

- Modern tool and infrastructure availability
- Workforce development

Developing policy

- Understanding gaps in existing policies
- New policies related to data ownership

AI Engineering Framework

AI Technologies and Components

Knowledge representation
Modeling and abstraction
Algorithms
Scalability and performance
Robust component design
Component V&V
Novelty and uncertainty

AI Systems

Architecture, analysis and design
Model-based engineering
Virtual integration
Robustness and resiliency
Secure AI
System V&V

Human-Machine Teaming

Interpretability and explainability
Human-machine trust
Machine-human trust
Co-learning
Communicating uncertainty
Data presentation

Process and Policy

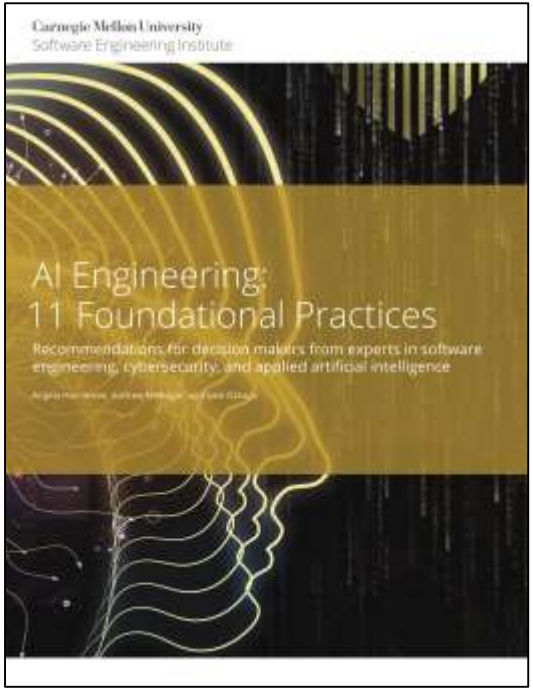
AI system acquisition
Modern software practices for AI
ML lifecycle management
Data lifecycle management
Standards
Benchmarks
Socio-technical issues
Economic considerations
AI-centric threat modeling
Risk and resiliency
Security coordination (AI C/PSIRT)
Ethics
Bias
Privacy

Infrastructure

Data, data management, data pipelines, data synthesis
Scalability, performance, and evaluation
Computational resources and considerations (CSWaP)
Processing placement and deployment, data locality
Platform choices and transferability
Tooling
Sensors and actuators
Future architectures

Safe ● Secure ● Robust ● Reliable ● Resilient

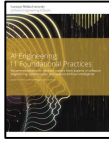
AI Engineering: Foundational Practices



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

Developing, deploying, and sustaining AI-enabled systems requires planning and ongoing resource commitment to take full advantage of the benefits.

Do you have an AI problem?



Ensure that you have a problem that both can and should be solved by AI.

- Your problem might have better, simpler solutions.

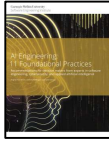
Do you know what measures of success look like for your problem?

- How do you model risk and uncertainty?
- Does your context tolerate and handle uncertainty of results?

If the solution requires data, do you have the necessary data?

- Do you have means to collect, store, and manage data in the long term as well as to secure it and determine who owns it?

Do you have the right team?



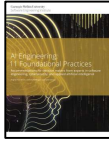
Successful AI system development is a team effort that includes experts in

- software engineering
- data science
- data architecting
- software and system architecting
- continuous integration and deployment
- operations
- the domain

Do you rely on external parties for data?

Are you ready to bring your contractors on board?

Data, data, data, data …



Resources spent in data management WILL consume your project.

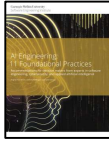
- *Do you have data? Do you have access rights to data? Do you have permission to use the data you have access to? How often does your data change? How will you collect data?....*

Automation and modern infrastructures are critical for successful AI adoption.

Ensure that your data management processes account for

- changes in environment
- adversarial exploitation
- biases in data

Do you understand change?

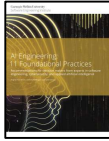


Teams need to plan and design for constant change to manage the “changing anything changes everything” principle*

- Architecting data pipelines to enable incorporating rate of change in data, retraining models, and incorporating new data from training to deployment to maintenance/evolution
- Designing instrumentations to enable traceability of the recommendations to improve explainability
- Changing environment conditions

* Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden Technical Debt in Machine Learning Systems. In *Proc. 28th Int. Conf. Advances in Neural Information Processing Systems, Vol. 2* (pp. 2503-2511). ACM, 2015.

Do you understand your quality attributes?



Quality attributes are properties of work products or goods by which stakeholders judge their quality.

Software cannot be designed to optimize for all quality attributes.

- Software engineers must select which attributes are most important for the stakeholder needs.
- Stakeholder needs will vary with the context in which a software system is deployed.

The degree to which a software system meets its quality attribute requirements depends on its architecture.

- Architectural decisions are made to promote various quality attributes.

L. Bass, P. Clements, R. Kazman. *Software Architecture Principles and Practices*, 3rd ed. Addison-Wesley, 2012.

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

High-Priority Quality Attributes for AI-Enabled Systems

AI engineering software design challenge	If you desire…	At a minimum, you need to …
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 over time.	data centricity	• ensure that uncertainty, availability, and scalability of data are key architecture drivers for system design • implement monitoring components to observe and manage data changes over time
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

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.

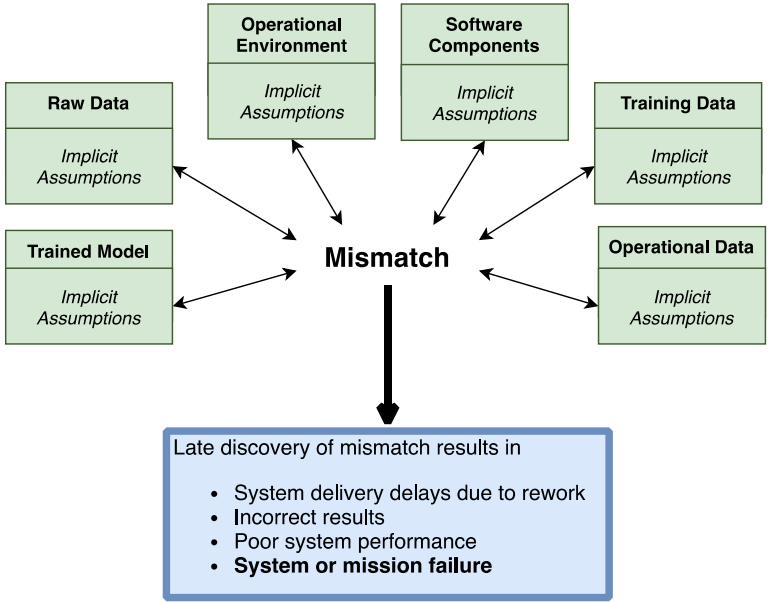
Mismatches Hindering Effective Design and Deployment of ML Systems

with Dr. Grace Lewis and Stephany Bellomo

Mismatch in ML-Enabled Systems

ML mismatch is 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 result in a negative consequence.

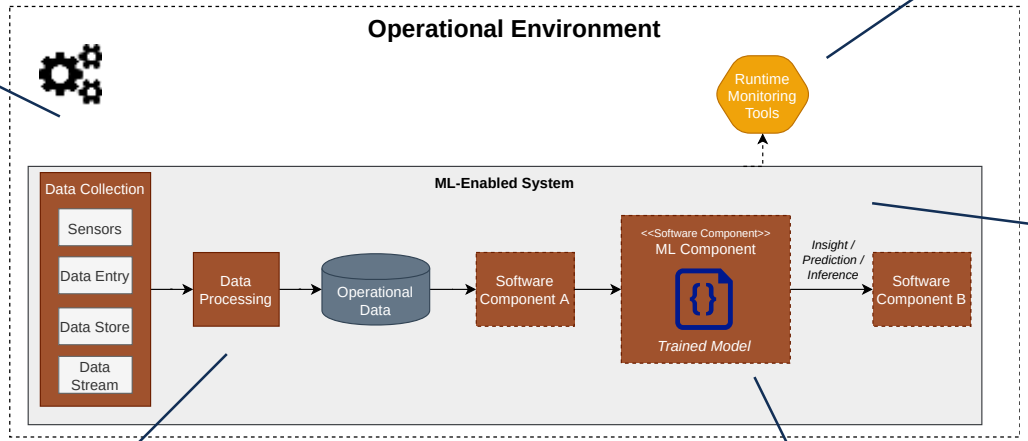
ML mismatch can also be traced back to information that would have helped avoid the problem if it had been shared with stakeholders.



Grace A. Lewis, Stephany Bellomo, April Galyardt. Component Mismatches Are a Critical Bottleneck to Fielding AI-Enabled Systems in the Public Sector. *Association for the Advancement of Artificial Intelligence AI in Public Sector Workshop.* Washington, DC, November 7–9, 2019.

Examples of Mismatch

Poor system performance because computing resources for model testing differ 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**)

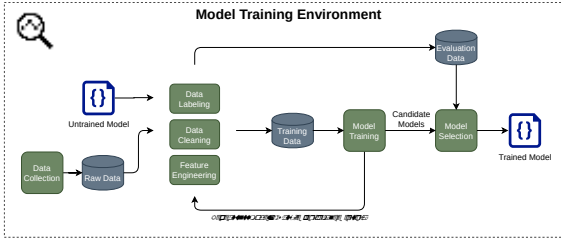
System failure due to poor testing—developers not able to replicate testing done during model training (**test data mismatch**)

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

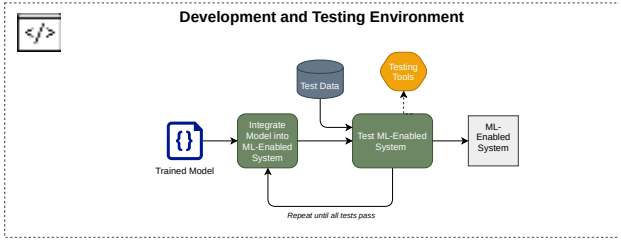
Large amounts of glue code because trained model input/output is very different from operational data types (**API mismatch**)

Problem: Multiple Perspectives

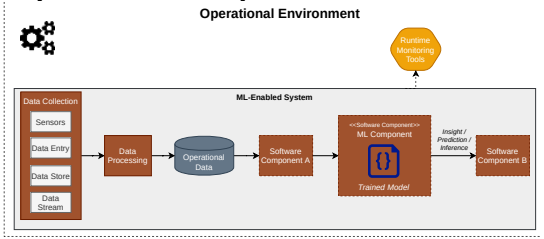
Data Scientist Perspective



Software Engineer Perspective



Operations Perspective



ML-enabled systems typically involve three different and separate workflows:

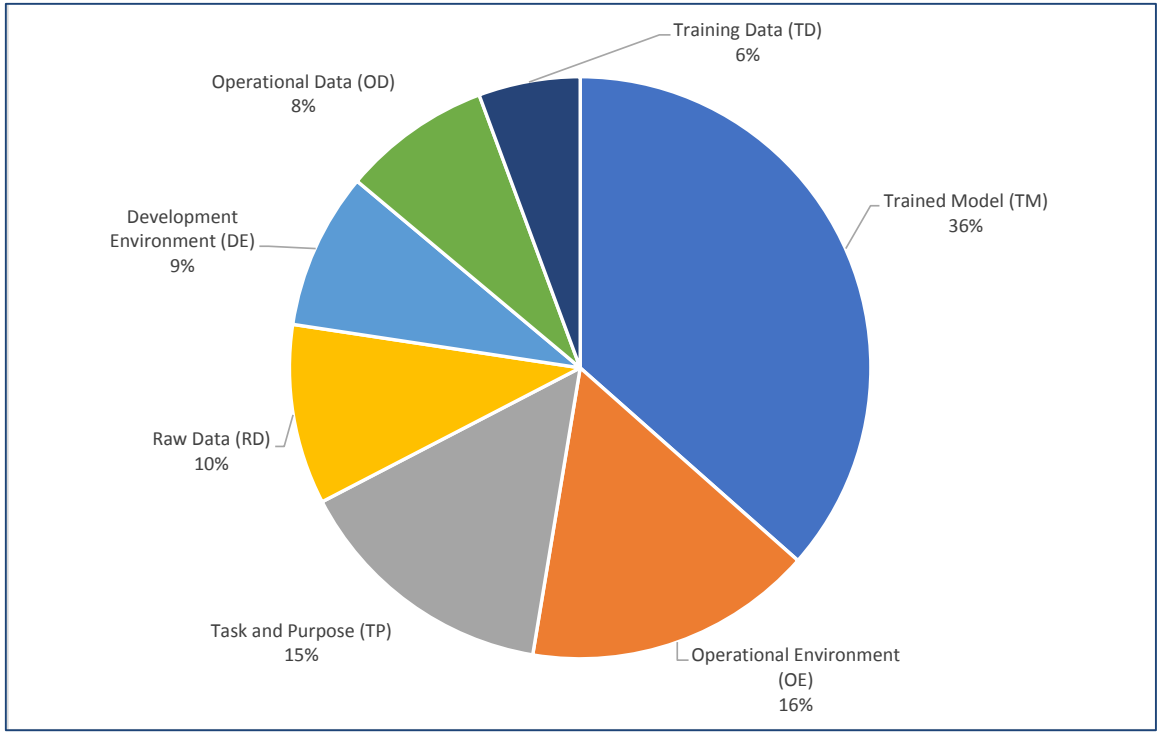
- model training
- model integration and testing
- model operation

… performed by three different sets of stakeholders:

- data scientists
- software engineers
- operations staff

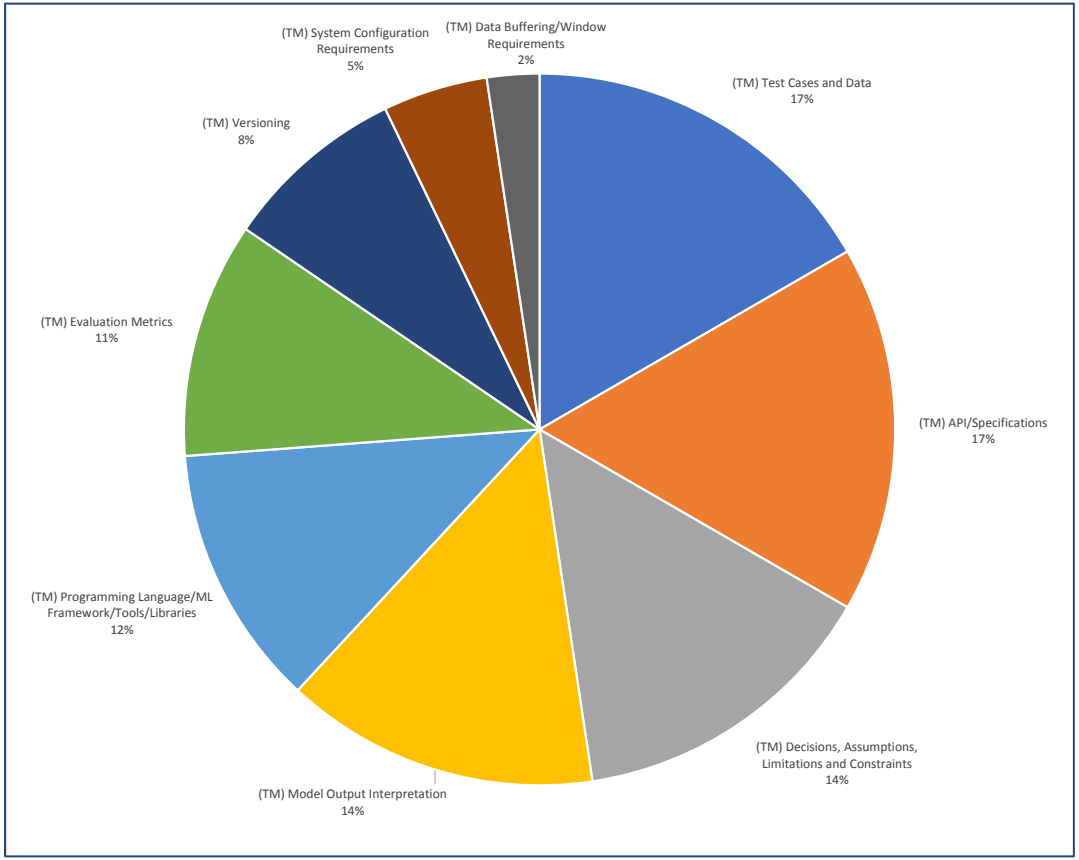
… with three different perspectives

Mismatched Categories



Most mismatches (36%) are due to incorrect assumptions made about the trained model, followed by incorrect assumptions made about the operational environment (16%)

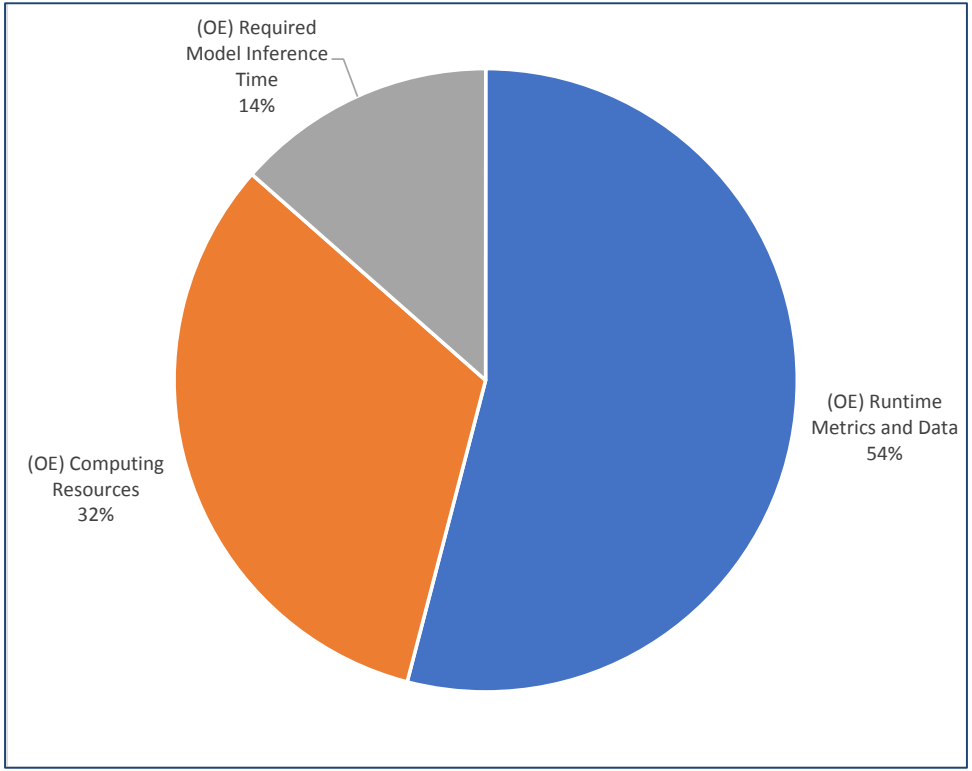
Trained Model Mismatches – Subcategories



Trained-model mismatches are mostly associated with lack of test cases and test data (17%) and lack of model specifications and APIs (17%)

"I had many attempts but was never able to get from the quants a description of what components exist, what are their specifications, what would be some reasonable test we could run against them so we could reproduce all those tests."

Operational Environment – Subcategories



Operational-environment mismatches are mostly associated with unavailable runtime metrics and data (54%) and unawareness of computing resources available for model serving (32%)

"A typical thing that might happen is that in production environment, something would happen. We would have a bad prediction, some sort of anomalous event. And we were asked to investigate that. Well, unless we have the same input data in our development environment, we can't reproduce that event."

Software Design Misconceptions for AI Systems

Software Design Misconceptions in AI Engineering – 1

We cannot specify AI systems.

- The level of specification of AI systems depends on the level of uncertainty in the discovery process. Sometimes uncertainty is low to none.
- AI specifications are specifications of problems, not systems.

We can avoid hidden dependencies.

- Understanding data and shared resource dependencies at runtime is not only an AI system problem but also a software system problem.
- Understanding data pipelines will lower the risks posed by hidden data dependencies.
- Existing software dependency analysis techniques can serve as a starting point.

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

Software Design Misconceptions in AI Engineering – 2

We can manage system change propagation.

- In AI systems, change management should be a top design driver.
- Decouple routine change from highly coupled data changes, which often are the source for unpredictable change propagation.

Frameworks and containers do it all.

- Generating an initial ML model DOES NOT imply designing and deploying a long-lived AI system.
- 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).*

Recommendations – Foundations

Ensure that you have a problem that both can and *should* be solved by AI.

Take your data seriously to prevent it from consuming your project.

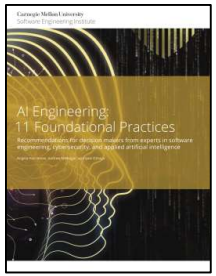
Design for interpretation of the inherent ambiguity in the output.

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.

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

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



Recommendations – Software Architecture and Design

Start with key architecturally significant requirements that will drive the AI system and underlying model(s).

Understand key areas of uncertainty and rates of change that the system will need to be designed for. Design for uncertainty.

Apply highly integrated monitoring and mitigation strategies for enforcing security by design.

Automate anything and everything possible to improve consistent update and monitoring mechanisms.

Embrace runtime monitoring as the only way to guarantee consistent model performance over time.

Contact Information

Ipek Ozkaya

Technical Director

Engineering Intelligent Software Systems

ozkaya@sei.cmu.edu