Overcoming Software Architecture Challenges for ML-Enabled Systems

Dr. Ipek Ozkaya

Technical Director

Engineering Intelligent Software Systems

ozkaya@sei.cmu.edu

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.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

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.

DM21-0850

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, #1 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] 2021 *US News and World Report*

AI-enabled systems are software systems!

SUMMARY OF THE 2018
DEPARTMENT OF DEFENSE
ARTIFICIAL INTELLIGENCE
STRATEGY

Harnessing AI to Advance
Our Security and Prosperity

THE NATIONAL
ARTIFICIAL INTELLIGENCE
RESEARCH AND DEVELOPMENT
STRATEGIC PLAN: 2019 UPDATE

A Report by the
SELECT COMMITTEE ON ARTIFICIAL INTELLIGENCE
of the
NATIONAL SCIENCE & TECHNOLOGY COUNCIL

JUNE 2019

"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*

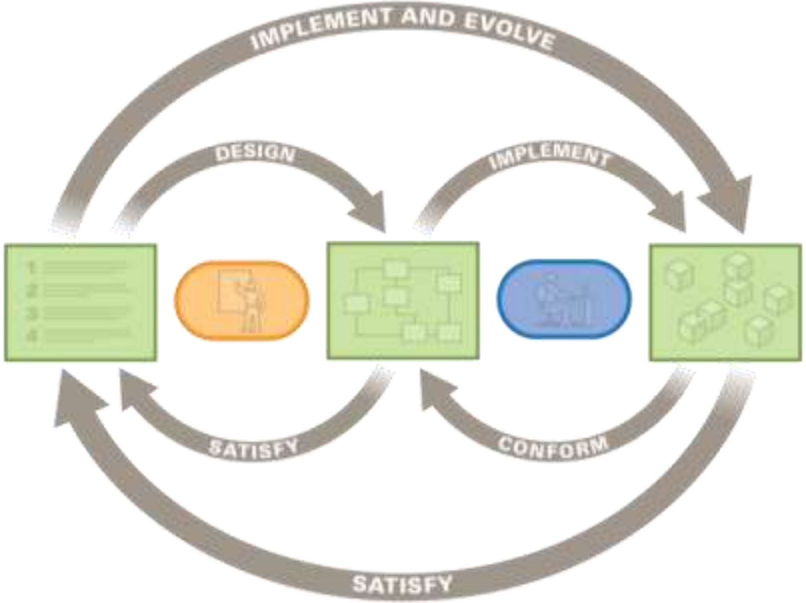
An **AI-enabled system** is a **software system** with one or more **AI component(s)** that need to be developed, deployed, and sustained along with the other software and hardware elements of the system.

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

Software Architecture Drives System Structure and Behavior

The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software components, relations among them and properties of both.

Architecture permits or precludes the *achievement of a system's desired quality attributes,* e.g. architecturally significant requirements.



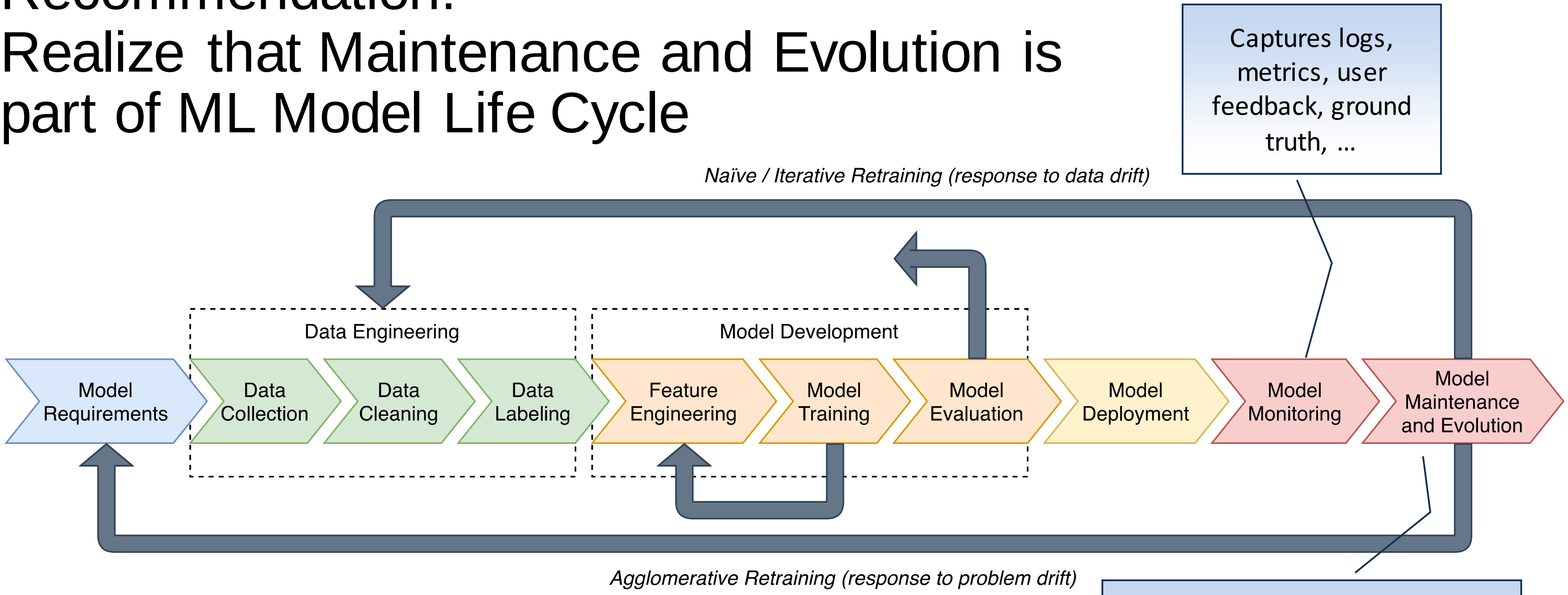
L. Bass, P. Clements, R. Kazman. *Software Architecture Principles and Practices*, 4th ed. Addison-Wesley, 2021.

Challenge #1:
Lack of Understanding of ML Model Life Cycle

While much attention has been paid to model development and evaluation, much less attention has been paid to putting models into production, and in particular model maintenance and evolution.

- This results in inability to understand architecturally significant attributes that need to be exposed for AI components and consequently inability to deploy reliable AI systems timely.

Recommendation:
Realize that Maintenance and Evolution is part of ML Model Life Cycle

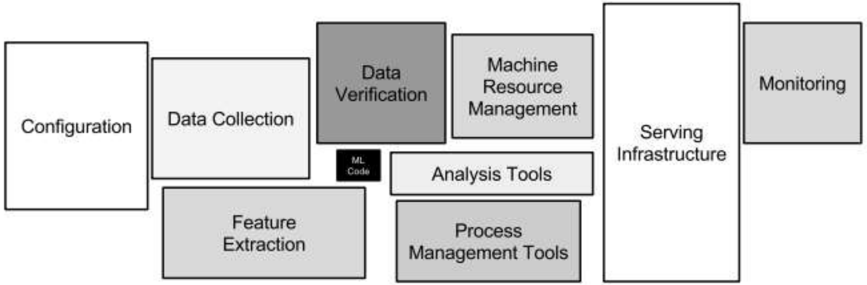

Captures logs, metrics, user feedback, ground truth, …
Naïve / Iterative Retraining (response to data drift)
Data Engineering
Model Development
Model Requirements
Data Collection
Data Cleaning
Data Labeling
Feature Engineering
Model Training
Model Evaluation
Model Deployment
Model Monitoring
Model Maintenance and Evolution
Agglomerative Retraining (response to problem drift)

Analyzes monitoring information to determine if the model needs to adapt to data drift or problem drift

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*

Challenge #2: Lack of Systems Perspective

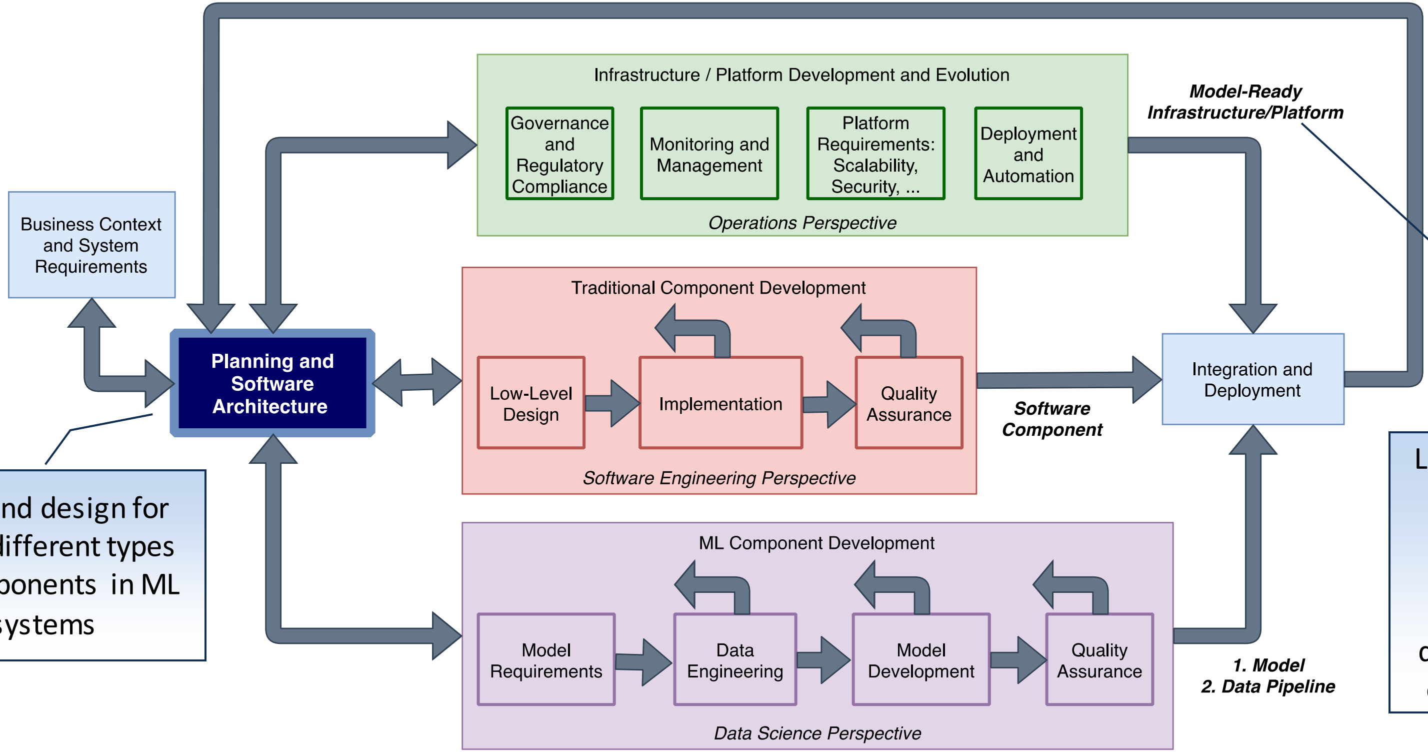
Failing to elicit, design for, and sustain the vast amount of other software components that the AI components need to interact with results in not architecting the systems appropriately and failed AI system development and deployment.



“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

Recommendation: Manage AI Components' Architectural Dependencies


Infrastructure / Platform Development and Evolution
Governance and Regulatory Compliance
Monitoring and Management
Platform Requirements: Scalability, Security, ...
Deployment and Automation
Operations Perspective
Model-Ready Infrastructure/Platform
Business Context and System Requirements
Planning and Software Architecture
Traditional Component Development
Low-Level Design
Implementation
Quality Assurance
Software Engineering Perspective
Software Component
Integration and Deployment
ML Component Development
Model Requirements
Data Engineering
Model Development
Quality Assurance
Data Science Perspective
1. Model
2. Data Pipeline

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

Challenge #3: Failures Related to Architecturally Significant Requirements

Key AI-specific concerns, when not approached with 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.

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

Challenge #4: Lack of 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)

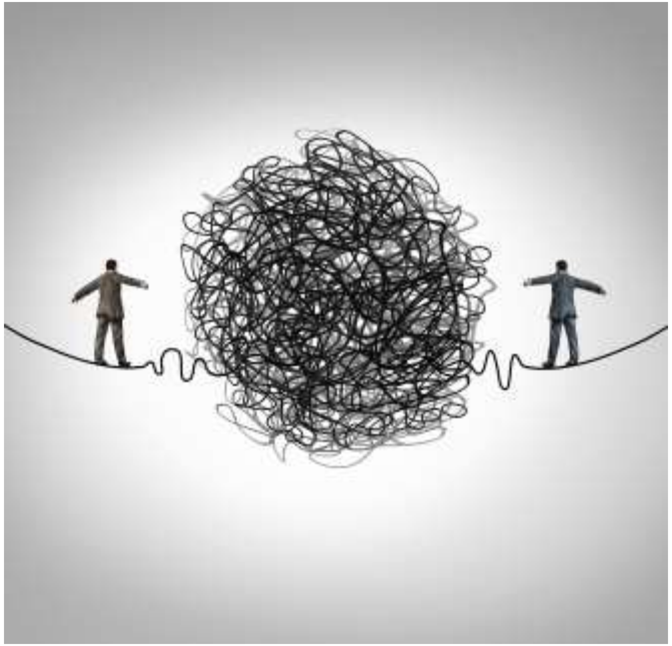
Understand how architectural elements that enable monitorability could also provide information to handle the inherent uncertainty of ML systems

* 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.

Challenge #5: Different Paces of Change

AI systems have several kids of rates of change and drivers of uncertainty.

Not managing for uncertainty and change in the architecture results in unintended system entanglement and significant resource spent in AI system development.



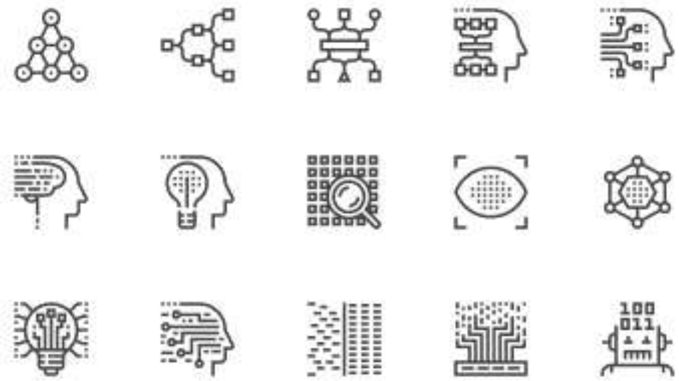
Recommendation: Embrace Changing Anything Changes Everything Principle*

Architect data pipelines to support incorporating rate of change in data, retraining models, and incorporating new data from training to deployment to maintenance and evolution.

Decouple routine change from highly-coupled data changes, which are often the source for unpredictable change propagation.

Design instrumentations to enable traceability of recommendations to improve explainability.

Do not forget changing environment conditions.



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

* 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.

Challenge #6: Lack of Recognizing Two Related Architectures in Play

An ML system has two software architectures that need to be developed and sustained in sync.

- Architecture of the pipeline that produces the ML model (or other AI components if applicable)
- Architecture of the system which uses and interacts with the AI/ML component

Recommendation: Recognize the Need for Co-Architecting and Co-Versioning

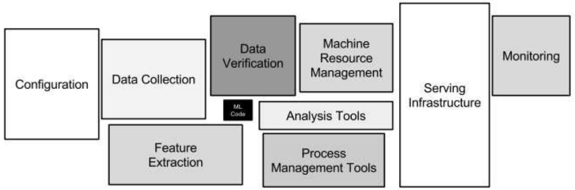
Co-Architecting: Both pipeline and system architectures need to be developed and sustained in sync such that design decisions are driven by both system and model requirements

Co-Versioning: Versioning in ML systems needs to provide traceability from training dataset to parameters to model to evaluation dataset to results to deployed model

- Key practice for maintenance and evolution

Summary Recommendations

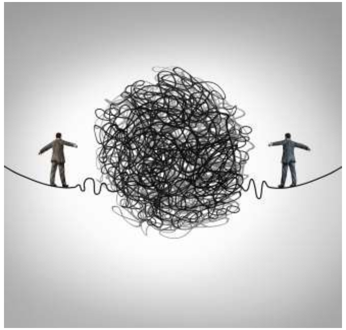
Realize that maintenance and evolution is part of ML Model Life Cycle



Manage ML components’ architectural dependencies

Understand high-priority quality attributes of AI-enabled systems

Decouple different aspects of monitorability



Embrace changing anything changes everything principle

Recognize the need for co-architecting and co-versioning

Contact Information

Dr. Ipek Ozkaya

Technical Director

Engineering Intelligent Software Systems

Software Engineering Institute

Carnegie Mellon University

ozkaya@sei.cmu.edu