What is Really Different in Engineering AI-Enabled Systems?

Ipek Ozkaya

Technical Director, Engineering Intelligent Software Systems

Carnegie Mellon University Software Engineering Institute

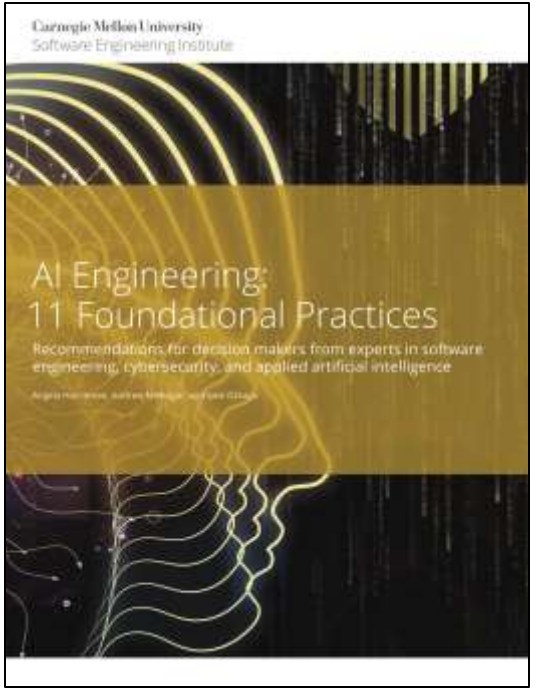
ozkaya@sei.cmu.edu

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.

DM22-0491

AI-enabled systems are software systems!



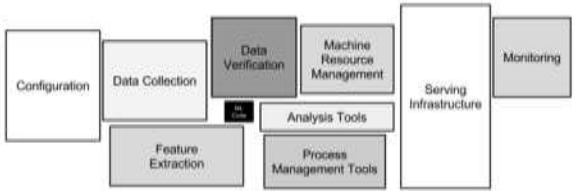
A. Horneman, A. Mellinger, I. Ozkaya. *AI Engineering: 11 Foundational Practices*. Pittsburgh: Carnegie Mellon University Software Engineering Institute, 2019.

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.

Challenges of Predictable Design and Analysis of AI-Enabled Systems

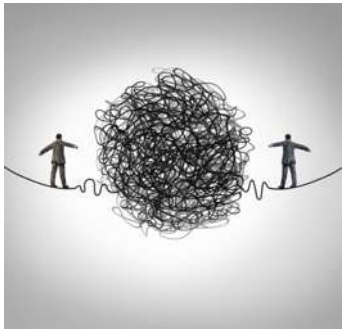
What changes are induced with maintenance and evolution of ML models?



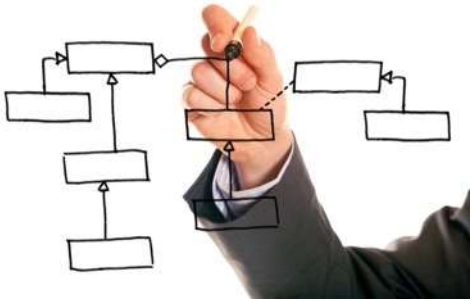
What are ML components' dependencies?

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?

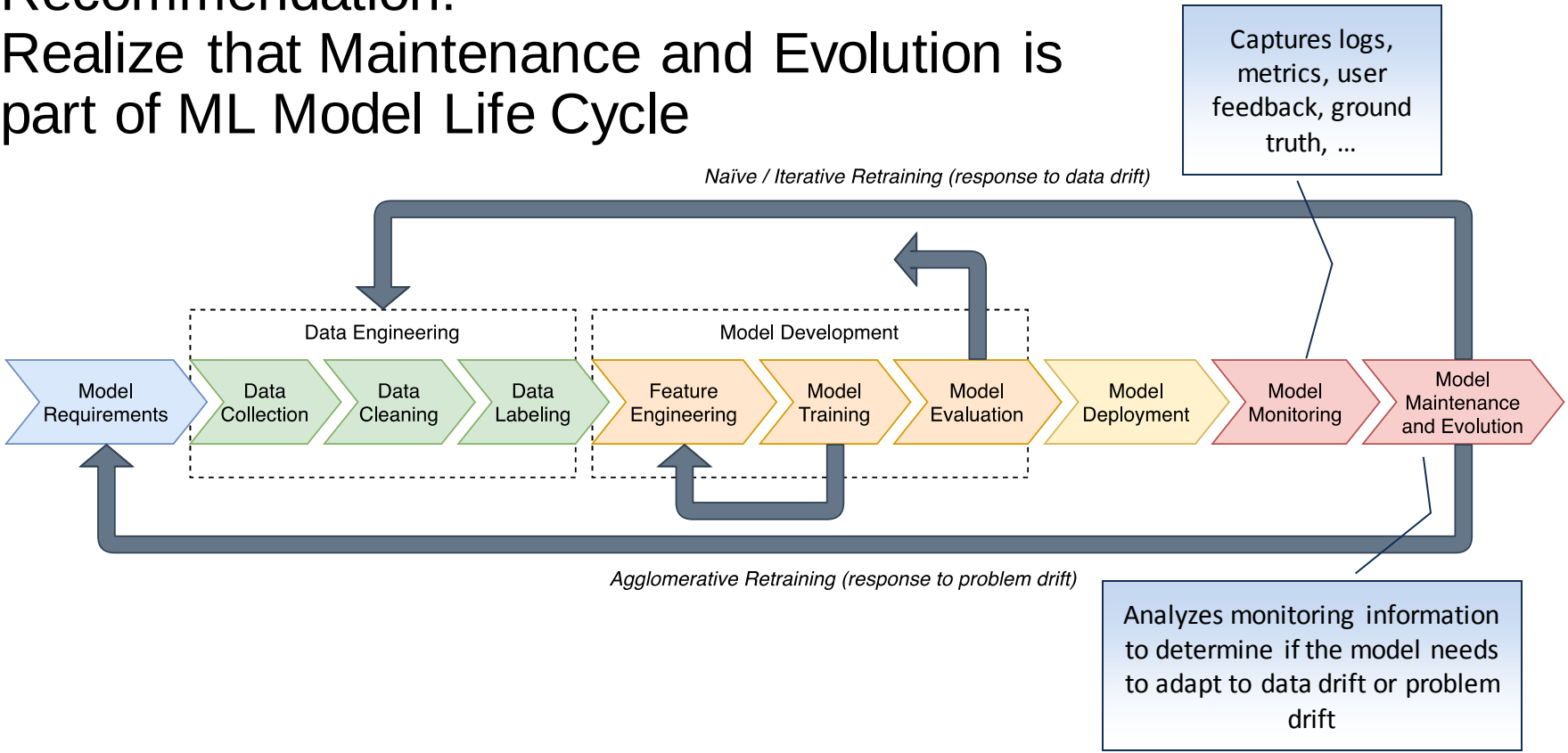


How can co-architecting and co-versioning needs be managed between the data pipelines and the system?

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

Much attention is paid to model development and evaluation, but not enough to putting models into production, and maintaining and evolving AI systems.

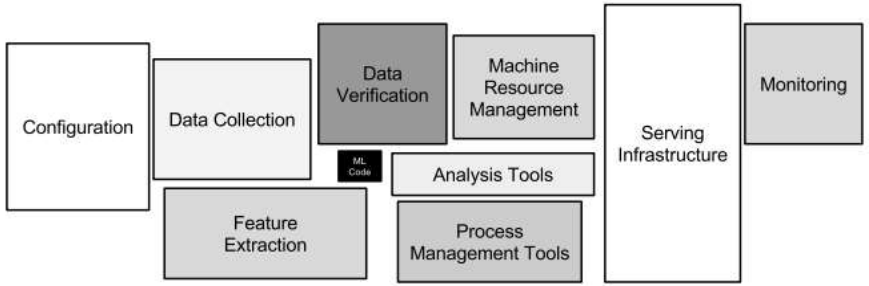
Recommendation: Realize that Maintenance and Evolution is part of ML Model Life Cycle



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*

Challenge #2: Lack of Systems Perspective

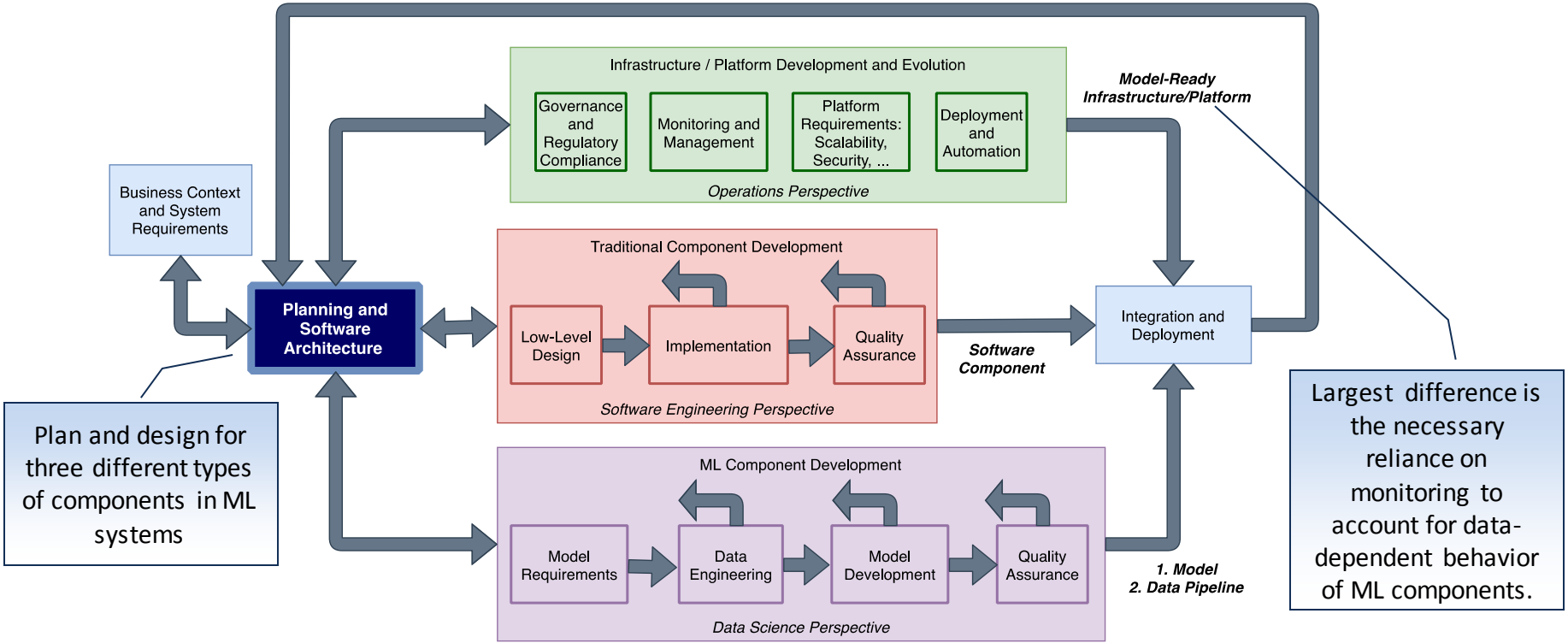
We fail to elicit, design for, and sustain the vast amount of other software components that AI components need to interact with



"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 Component, Data, and Architectural Dependencies



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

Challenge #3: Inability to separate data and system attributes

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

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

Key data concerns create unanticipated AI failure

- ethics, bias, fairness
- incorrect outcomes

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 degrade at a different rate than the rest of the 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 #1: 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 the relationship of monitorability and 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)

* 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.

Recommendation #2: Integrate the analyses performed by the Data Scientist into the MLOps pipeline

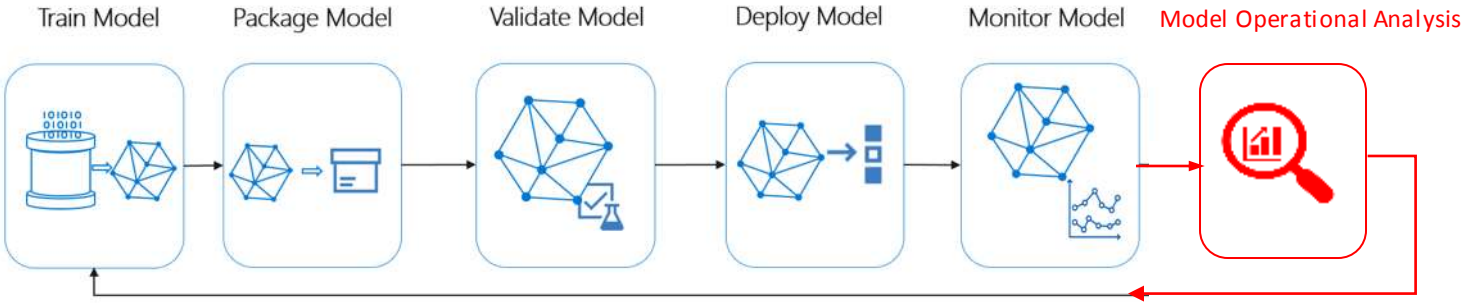


Diagram Adapted from MS Azure MLOps Pipeline

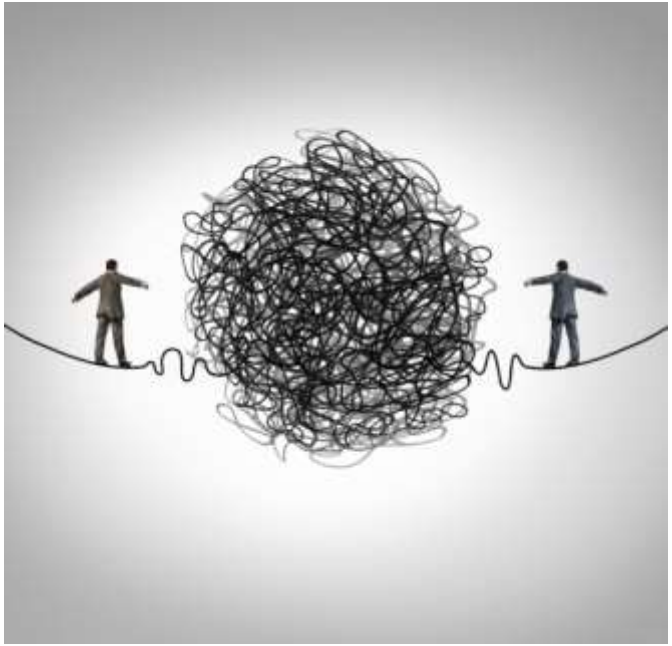
MLOps automates model deployment, but creates a model retraining problem

- 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

https://insights.sei.cmu.edu/blog/improving-automated-retraining-of-machine-learning-models/

Challenge #5: Different Paces of Change

AI systems have several kinds and rates of change that we do not yet fully understand and have techniques for to manage.



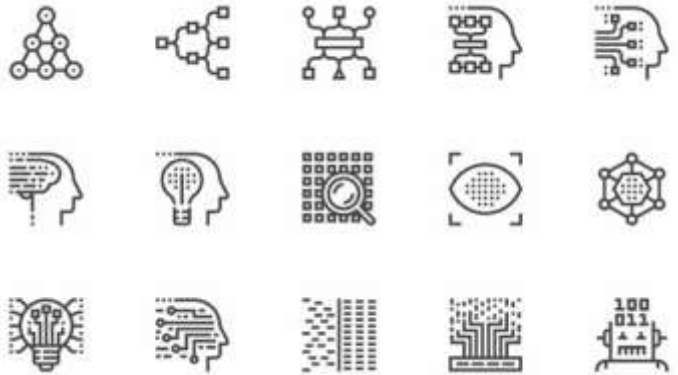
Recommendation: Embrace Changing Anything Changes Everything Principle*

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: Failure to Recognize Two Related Systems in Play

An ML system may have two software systems that need to be developed and sustained in sync.

- The pipeline that produces the ML model (or other AI components if applicable)
- The system that 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

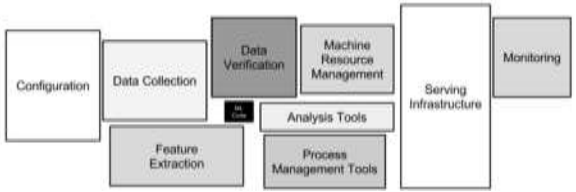
Co-Versioning: Versioning in ML systems needs to provide traceability from training dataset to parameters to model to evaluation dataset to results to deployed model

- Key practice for maintenance and evolution

Grace A. Lewis, Ipek Ozkaya, Xiwei Xu:
Software Architecture Challenges for ML Systems. ICSME 2021: 634-638

Summary Recommendations

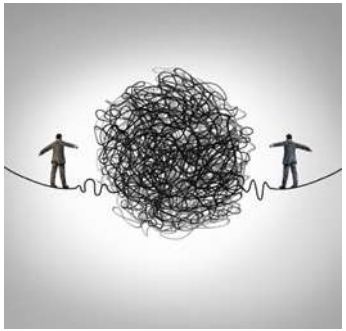
Realize that maintenance and evolution is part of ML Model Life Cycle



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

Ipek Ozkaya

Technical Director

Engineering Intelligent Software Systems

Software Engineering Institute

Carnegie Mellon University

ozkaya@sei.cmu.edu