Detection and Reporting Preparation to Support JPSS-2

Dr. Jon Neff
Aerospace Civil Systems Group

Approved for public release. OTR 2020-00362.

Desired End State

For JPSS-2 operations

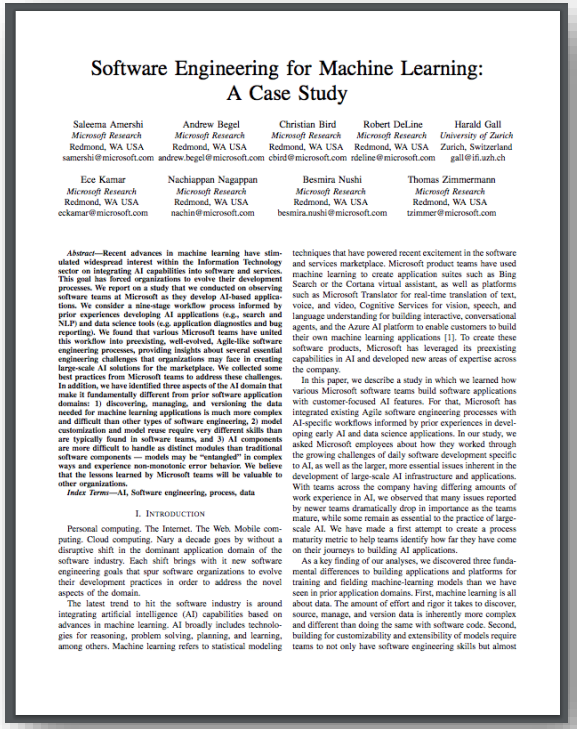
- Pre-screen large telemetry volume: ~80,000 mnemonics, 28 GB/day
- Detect potential anomalies when telemetry is still within limits.
- Give engineers time back rather than consuming their time.
- Completely automated data pipeline.
- Support multiple users.
- Everything is resilient and works.
- Displays for telemetry, anomaly detection, etc. Max latency 10 seconds.
- New telemetry source can be implemented in hours.
- Plug and play different anomaly detection algorithms (as containers).
- Engineers can quickly demonstrate proficiency.

D&R should be easy to use. Data pipeline should be simple to maintain.

Data Pains? We Are Not Alone…

From Recent Microsoft Internal Survey of 551 AI/ML Software Engineers:*

- Common Challenges
  - *End-to-end tool fragmentation*
  - ***Data collection, cleaning, management***
- Desires and Suggestions
  - *"Focus on building a **super solid data pipeline which continuously loads and massages data**, which enables us to try different AI algorithms with different hyper parameters, etc. without much hassle."*
  - *"**Pay a lot of attention to data**."*
  - *"**Put more efforts on data collection and annotation**"*
  - *"Be relaxed about framework / machine learning code, but **careful and deliberate about data** & objectives."*
  - *"Ensure **complete traceability of all training and test data**, …"*
  - *"**Center development around data** (sharing, provenance, versioning)"*



Software Engineering for Machine Learning:
A Case Study

Saleema Amershi, Microsoft Research, Redmond, WA USA, samershi@microsoft.com
Andrew Begel, Microsoft Research, Redmond, WA USA, andrew.begel@microsoft.com
Christian Bird, Microsoft Research, Redmond, WA USA, cbird@microsoft.com
Robert DeLine, Microsoft Research, Redmond, WA USA, rdeline@microsoft.com
Harald Gall, University of Zurich, Zurich, Switzerland, gall@ifi.uzh.ch
Ece Kamar, Microsoft Research, Redmond, WA USA, eckamar@microsoft.com
Nachiappan Nagappan, Microsoft Research, Redmond, WA USA, nachin@microsoft.com
Besmira Nushi, Microsoft Research, Redmond, WA USA, besmira.nushi@microsoft.com
Thomas Zimmermann, Microsoft Research, Redmond, WA USA, tzimmer@microsoft.com

***Abstract*—Recent advances in machine learning have stimulated widespread interest within the Information Technology sector on integrating AI capabilities into software and services. This goal has forced organizations to evolve their development processes. We report on a study that we conducted on observing software teams at Microsoft as they develop AI-based applications. We consider a nine-stage workflow process informed by prior experiences developing AI applications (e.g., search and NLP) and data science tools (e.g. application diagnostics and bug reporting). We found that various Microsoft teams have united this workflow into preexisting, well-evolved, Agile-like software engineering processes, providing insights about several essential engineering challenges that organizations may face in creating large-scale AI solutions for the marketplace. We collected some best practices from Microsoft teams to address these challenges. In addition, we have identified three aspects of the AI domain that make it fundamentally different from prior software application domains: 1) discovering, managing, and versioning the data needed for machine learning applications is much more complex and difficult than other types of software engineering, 2) model customization and model reuse require very different skills than are typically found in software teams, and 3) AI components are more difficult to handle as distinct modules than traditional software components — models may be "entangled" in complex ways and experience non-monotonic error behavior. We believe that the lessons learned by Microsoft teams will be valuable to other organizations.**

***Index Terms*—AI, Software engineering, process, data**

I. Introduction

Personal computing. The Internet. The Web. Mobile computing. Cloud computing. Nary a decade goes by without a disruptive shift in the dominant application domain of the software industry. Each shift brings with it new software engineering goals that spur software organizations to evolve their development practices in order to address the novel aspects of the domain.

The latest trend to hit the software industry is around integrating artificial intelligence (AI) capabilities based on advances in machine learning. AI broadly includes technologies for reasoning, problem solving, planning, and learning, among others. Machine learning refers to statistical modeling techniques that have powered recent excitement in the software and services marketplace. Microsoft product teams have used machine learning to create application suites such as Bing Search or the Cortana virtual assistant, as well as platforms such as Microsoft Translator for real-time translation of text, voice, and video, Cognitive Services for vision, speech, and language understanding for building interactive, conversational agents, and the Azure AI platform to enable customers to build their own machine learning applications [1]. To create these software products, Microsoft has leveraged its preexisting capabilities in AI and developed new areas of expertise across the company.

In this paper, we describe a study in which we learned how various Microsoft software teams build software applications with customer-focused AI features. For that, Microsoft has integrated existing Agile software engineering processes with AI-specific workflows informed by prior experiences in developing early AI and data science applications. In our study, we asked Microsoft employees about how they worked through the growing challenges of daily software development specific to AI, as well as the larger, more essential issues inherent in the development of large-scale AI infrastructure and applications. With teams across the company having differing amounts of work experience in AI, we observed that many issues reported by newer teams dramatically drop in importance as the teams mature, while some remain as essential to the practice of large-scale AI. We have made a first attempt to create a process maturity metric to help teams identify how far they have come on their journeys to building AI applications.

As a key finding of our analyses, we discovered three fundamental differences to building applications and platforms for training and fielding machine-learning models than we have seen in prior application domains. First, machine learning is all about data. The amount of effort and rigor it takes to discover, source, manage, and version data is inherently more complex and different than doing the same with software code. Second, building for customizability and extensibility of models require teams to not only have software engineering skills but almost

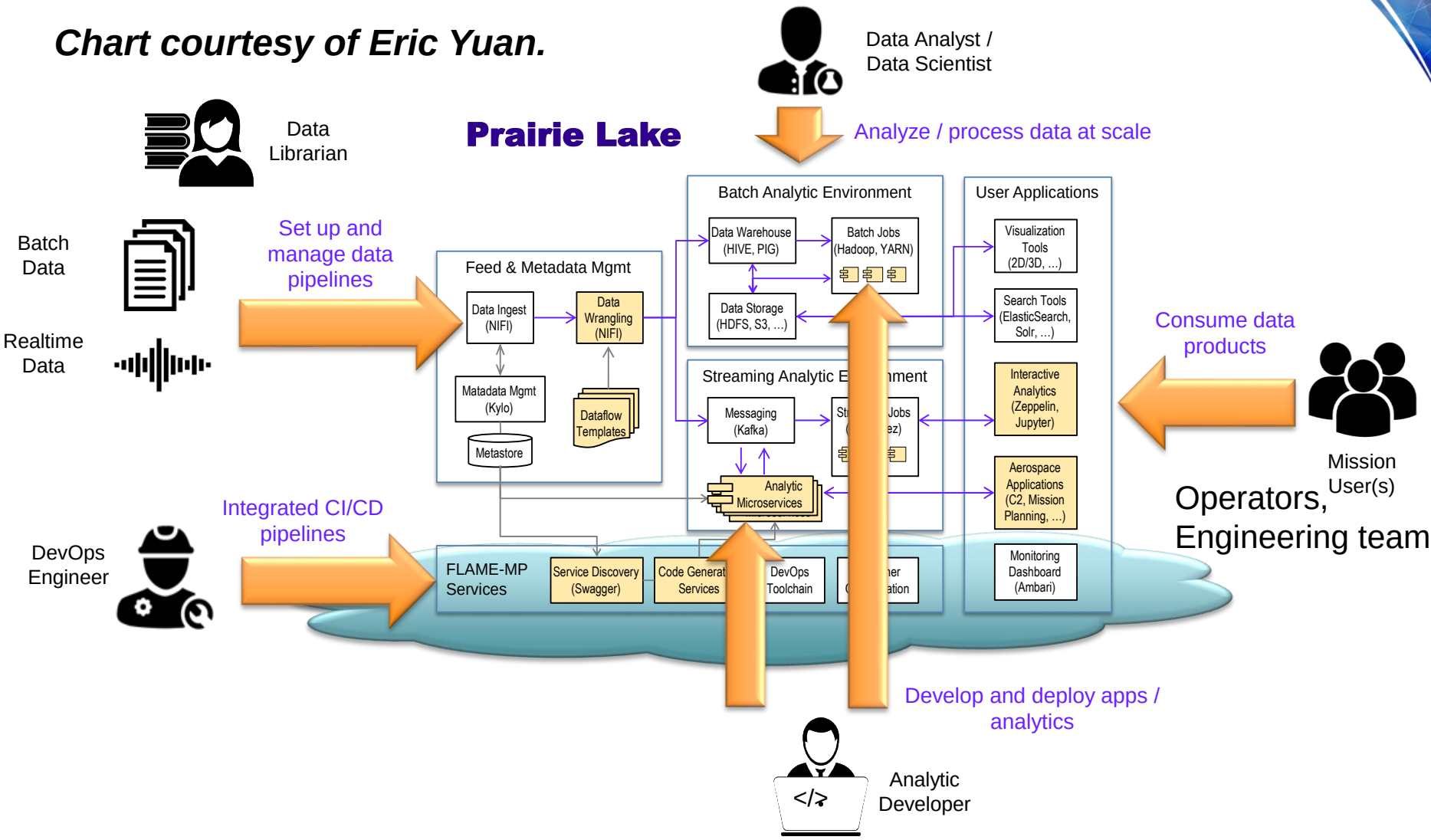
* Saleema Amershi et al., "*Software Engineering for Machine Learning: A Case Study*", ICSE'2019

Another study says 8 of 10 enterprise AI projects are stalled by data problems.

Prairie Lake Architecture and Roles

A platform for automated data pipelines based on open-source software

Chart courtesy of Eric Yuan.



Prairie Lake's separation of concerns and roles makes it much easier to maintain and run D&R.

Implementation Phases

Phase 1
Current: basic prototype

Gov't Resource for Algorithm Verification, Independent Test, & Evaluation

GRAVITE

rsync

Telemetry HDF5 format

HPC Cluster

scp

Large files

Windows PC

Filter data & train D&R nets

- **Basic Prototype**
- **Measure of Performance**

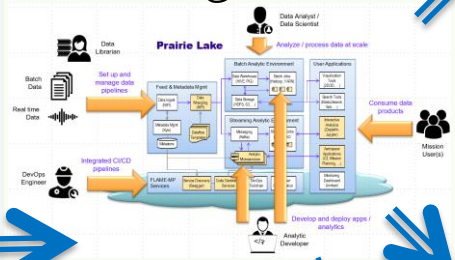
Phase 2
Advanced prototype

GRAVITE

OR

STA

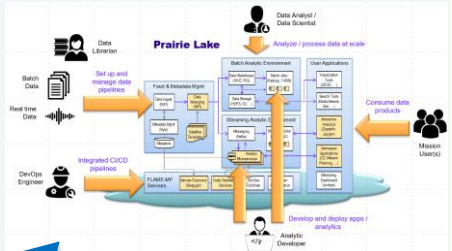
D&R @ ADVC



- **Improve Performance**
- **Attribution**
- **Improve Efficiency & Ease of Use**
- **Resolve Data Source for Telemetry& Anomalies**

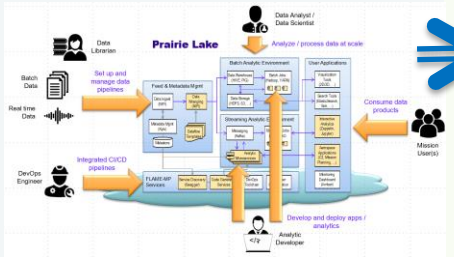
Phase 3
Migrate to host/T&E

D&R @ NSOF on-prem



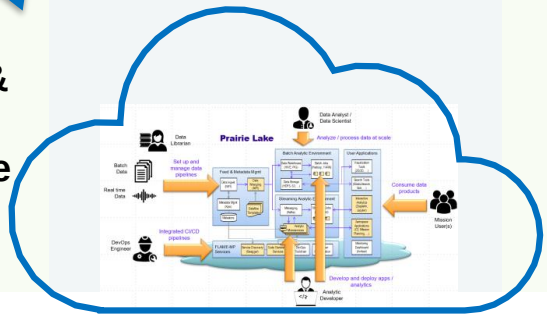
OR

D&R @ Aerospace on-prem



OR

D&R @ Gov Cloud



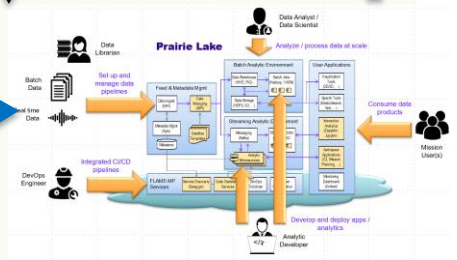
Phase 4
Operations

JPSS SIGNAL HANDLING (JSH)

Telemetry ARF Format

FOT

Alerts



- **Support operations**