MLOps, the hard way

A journey on integrating AI and Software Engineering

Prof. Michele Ciavotta
michele.ciavotta@unimib.it
University of Milano - Bicocca

Before we start

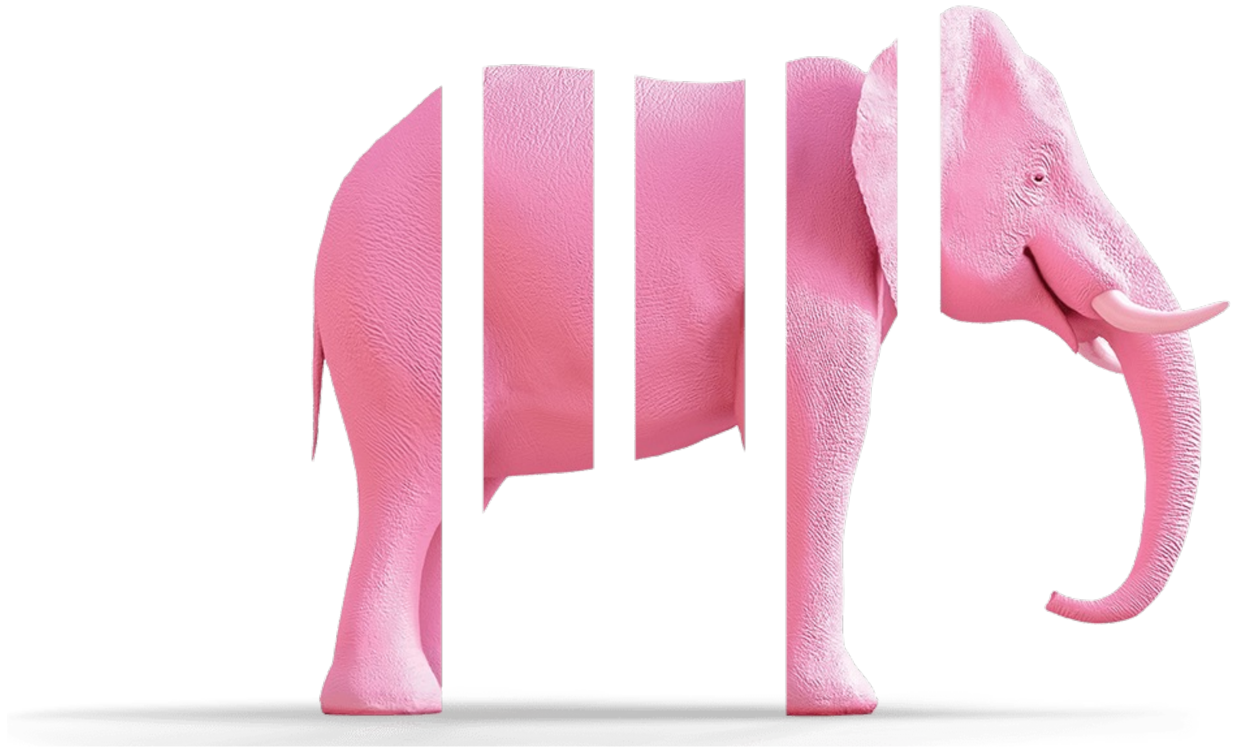
Objective (MLOps is hard)

Present and comment on some mistakes (technical dept and antipatterns) made during a project to create an AI-based solution for finance.

Client: SME in Fintech

UniMiB served as a consultant in SE and AI

Project Duration Jan '20 - Jul '23



Before we start

Objective (MLOps is hard)

Present and comment on some mistakes (technical dept and antipatterns) made during a project to create an AI-based solution for finance.

Client: SME in Fintech

UniMiB served as a consultant in SE and AI

Project Duration Jan '20 - Jul '23

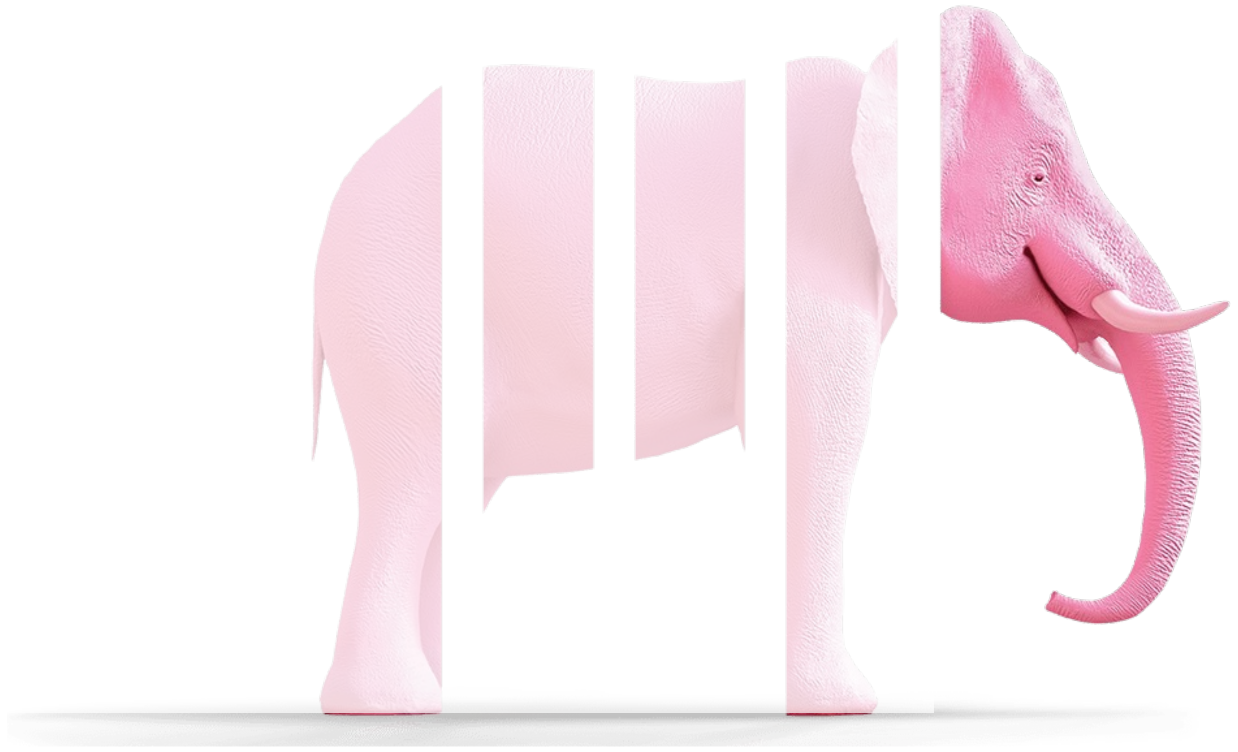
Disclaimer

Anecdotical, 1 Data point

Italian SME with specific context

    Brownfield - Greenfield

Presentation time is limited



Context: AI-based solution

Hofmann et al. (2020) distinguish three AI solution types according to the role, scope, and value contribution of AI:

- **AI-enabled solutions** use AI to improve or simplify the input and output interfaces to the user. They support interaction, often based on natural language processing. One example is ChatGPT.
- **AI-based solutions** use AI to implement the actual **core task** in processes and thus create new insights. One example is learning-based credit risk assessments.
- **Complete AI solutions** use AI to support both input and output as well as the actual task processing.

P. Hofmann, J. Jöhnk, D. Protschky, and N. Urbach, “Developing Purposeful AI Use Cases – A Structured Method and Its Application in Project Management,” presented at the WI2020

Context: AI-based solution

Hofmann et al. (2020) distinguish three AI solution types according to the role, scope, and value contribution of AI:

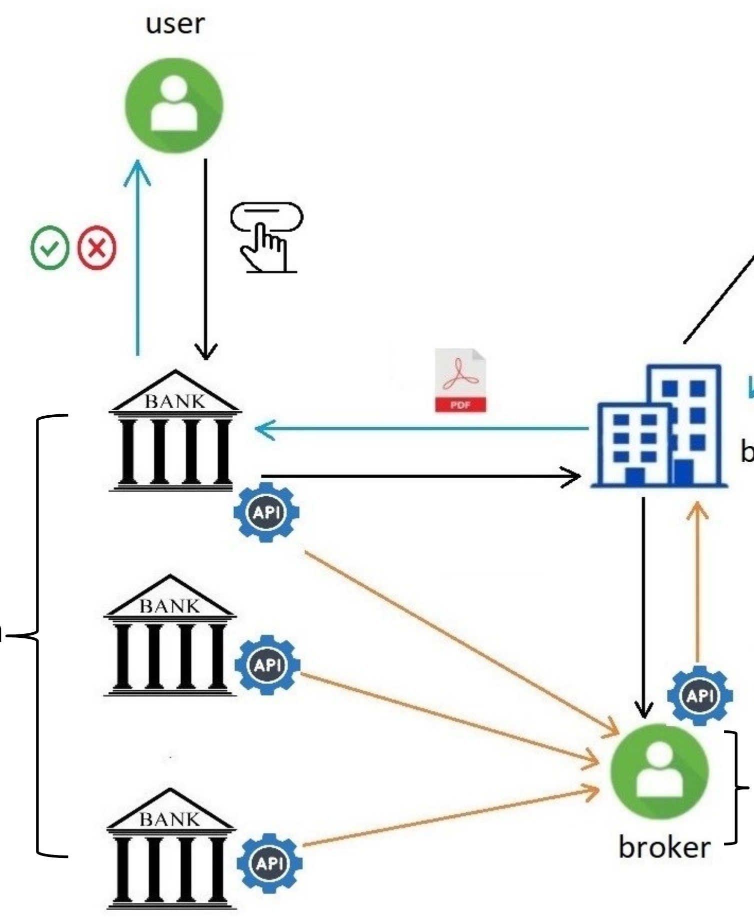
- **AI-enabled solutions** use AI to improve or simplify the input and output interfaces to the user. They support interaction, often based on natural language processing. One example is ChatGPT.

- **AI-based solutions** use AI to implement the actual **core task** in processes and thus create new insights. One example is learning-based credit risk assessments.

- **Complete AI solutions** use AI to support both input and output as well as the actual task processing.

P. Hofmann, J. Jöhnk, D. Protschky, and N. Urbach, "Developing Purposeful AI Use Cases – A Structured Method and Its Application in Project Management," presented at the WI2020

Context: Credit score assessment service



Service

executed within a virtual machine

SME

passes the data received from the broker to the service

Provide access to customer data, with the explicit consent of the user, to authorized third parties

Consortium members

Authorized third-party

downloads transactions recorded on the various customer bank accounts

harmonizes movements according to internal categories

→	request
→	response
→	download

If we had only known earlier

At the project's kick off (Jan '20)

- We had substantial experience in Data Science projects
- Even if there was already much emphasis on AI and MLOps but we were not aware of systematic studies on antipattern and technical debt in the intersection between AI and SE

Sculley, David, et al. "Machine learning: The high interest credit card of technical debt." SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)

If we had only known earlier

At the project's kick off (Jan '20)

- We had substantial experience in Data Science projects
- Even if there was already much emphasis on AI and MLOps but we were not aware of systematic studies on antipattern and technical debt in the intersection between AI and SE

1. H. Muccini and K. Vaidhyanathan, "*Software Architecture for ML-based Systems: What Exists and What Lies Ahead*," in 2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN), Madrid, Spain, 2021 pp. 121-128.

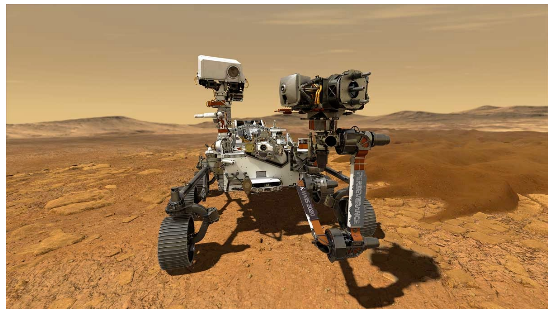
2. Serban, K. van der Blom, H. Hoos, and J. Visser, "*Adoption and Effects of Software Engineering Best Practices in Machine Learning*", Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–12, Oct. 2020.

3. J. Bogner, R. Verdecchia and I. Gerostathopoulos, "*Characterizing Technical Debt and Antipatterns in AI-Based Systems: A Systematic Mapping Study*" 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021, pp. 64-73.

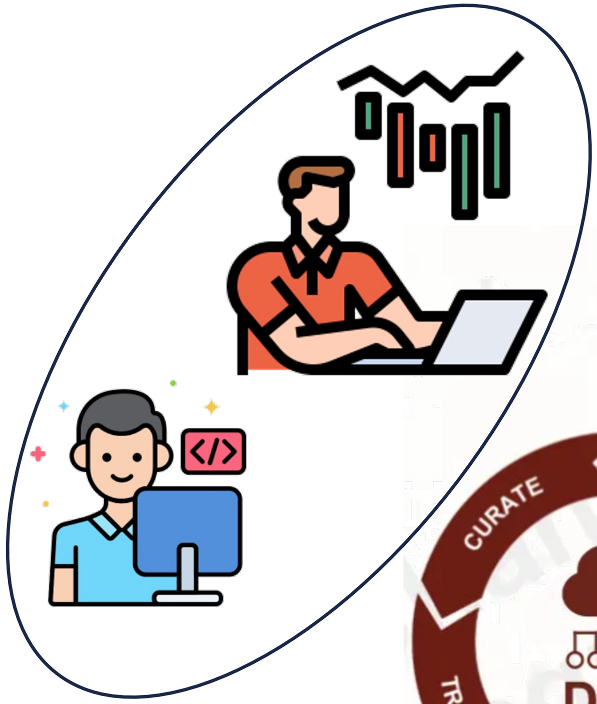
4. K. Shivashankar, and A Martini. "*Maintainability Challenges in ML: A Systematic Literature Review.*" 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2022.

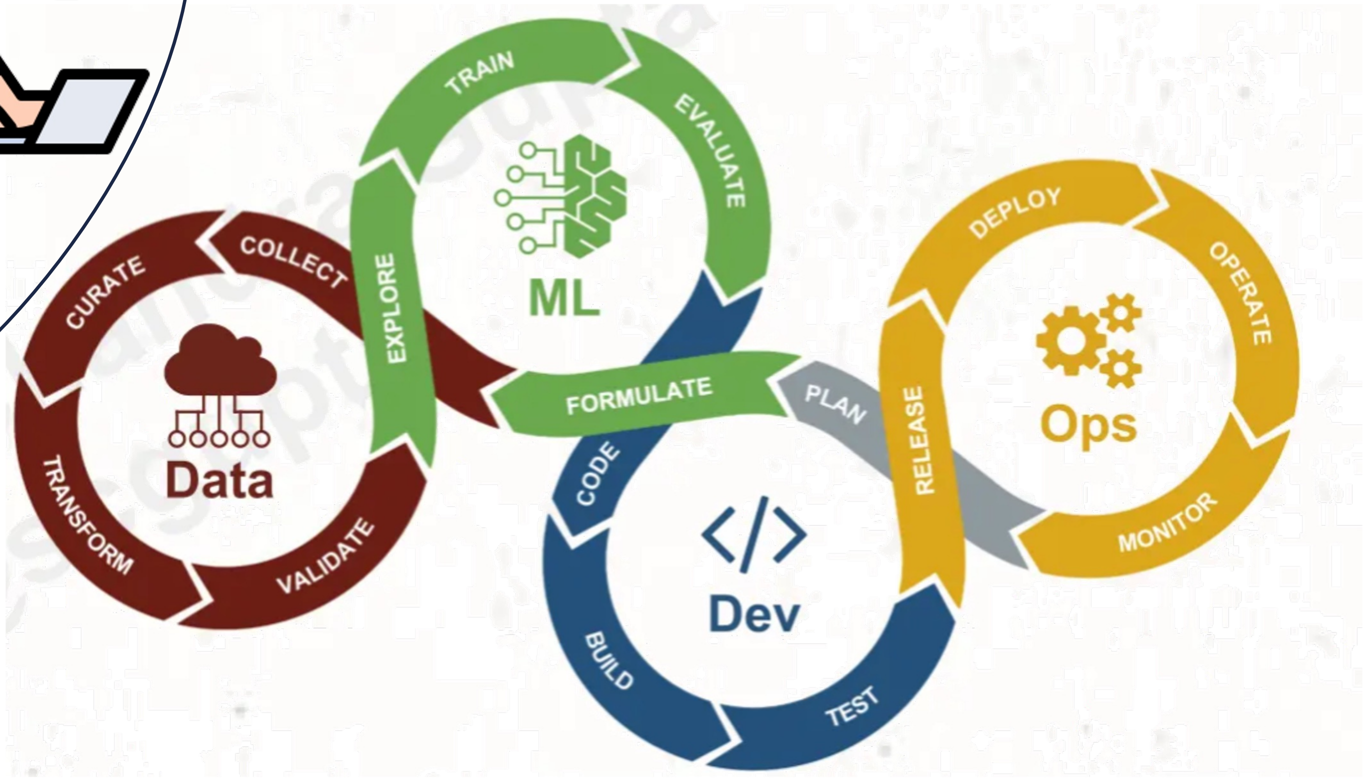
5. N. Muralidhar, et al. "*Using antipatterns to avoid mlops mistakes*". arXiv preprint arXiv:2107.00079 (2021).

Wait, what? Did you say MLOps?

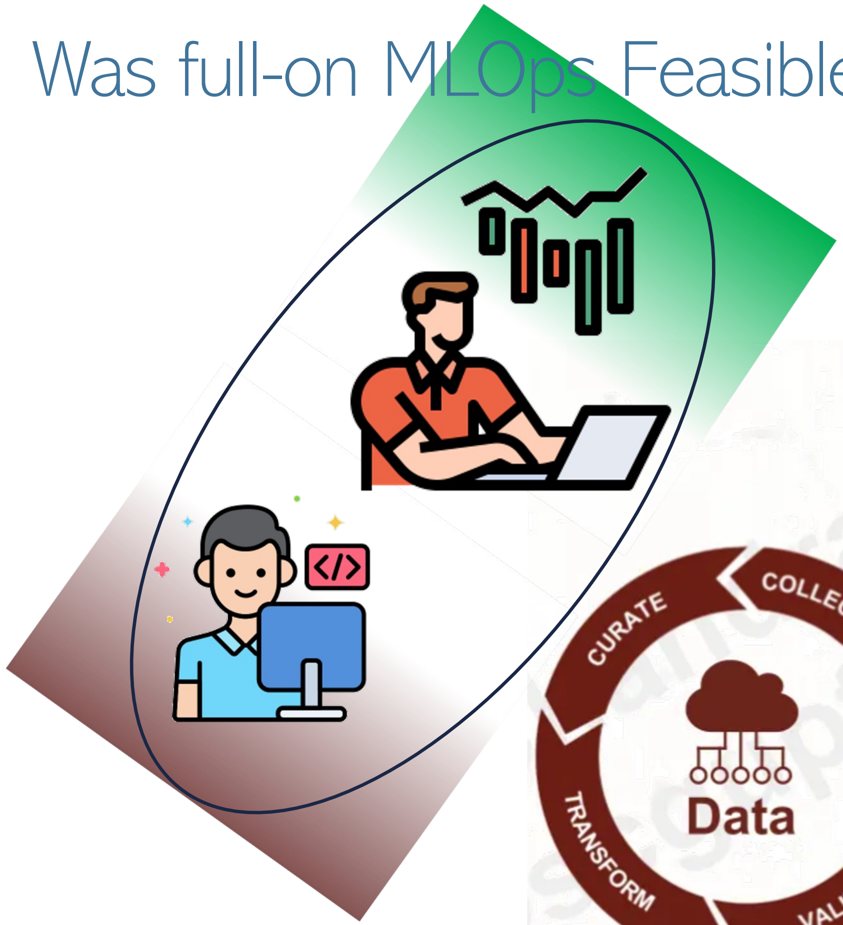


Was full-on MLOps Feasible?



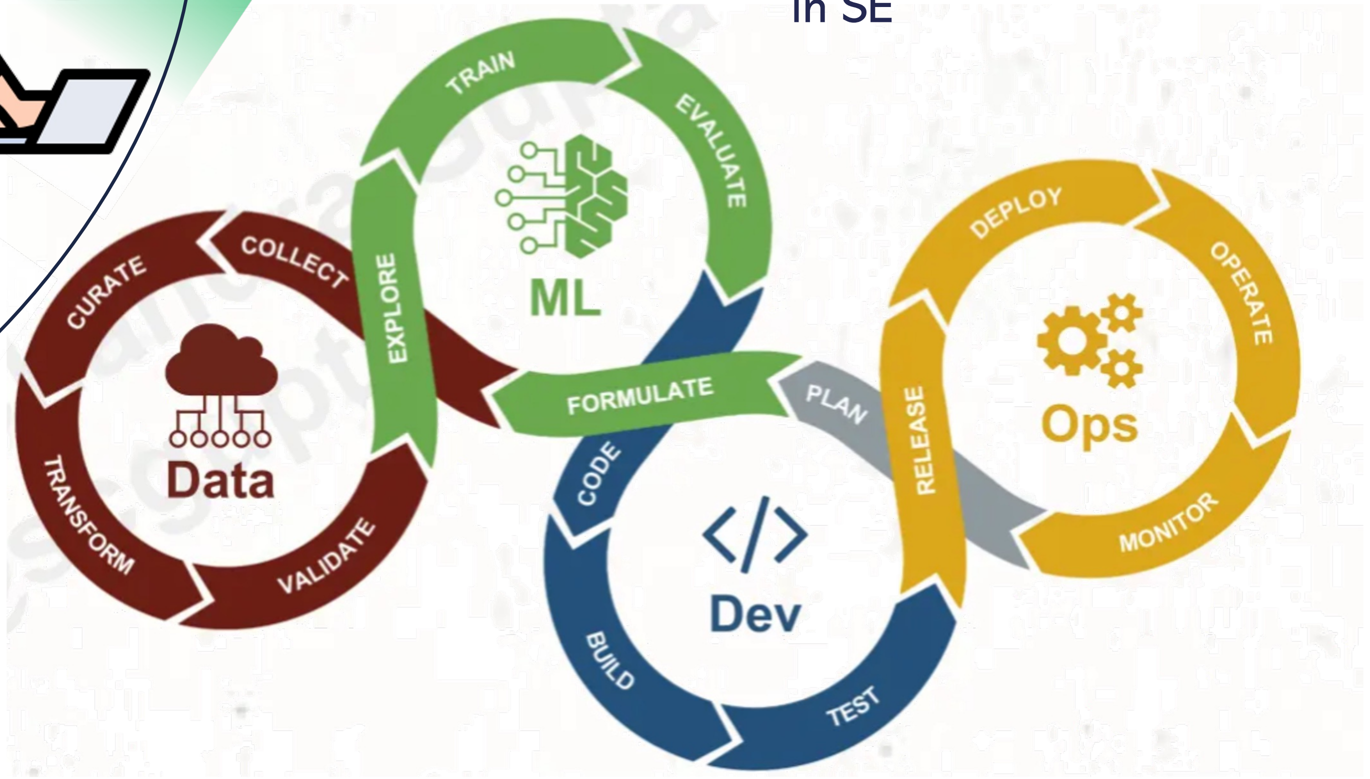


Was full-on MLOps Feasible?



Brownfield – Greenfield scenario:

- Developers used to do work in on more classical task
- Freshly graduate young data scientists with no expertise in SE

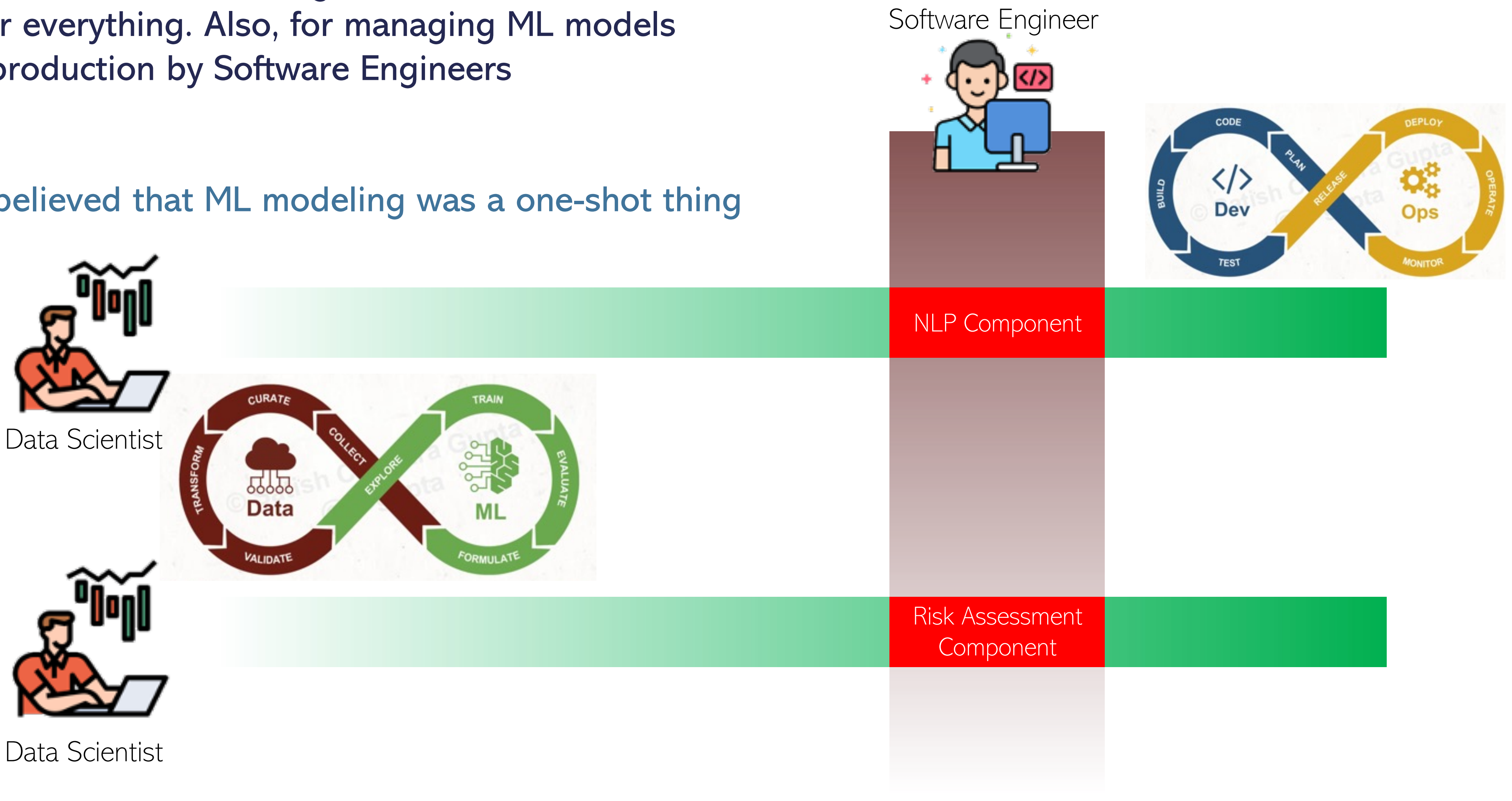


The origin of a dooming decision…

So, what happened? We didn't fight the client's requirements:

1. Go for the monolith. Let's start small, right?
2. Go for Java. Yes, for everything. Also, for managing ML models
3. Models are put in production by Software Engineers
4. On premises

Deep down the client believed that ML modeling was a one-shot thing

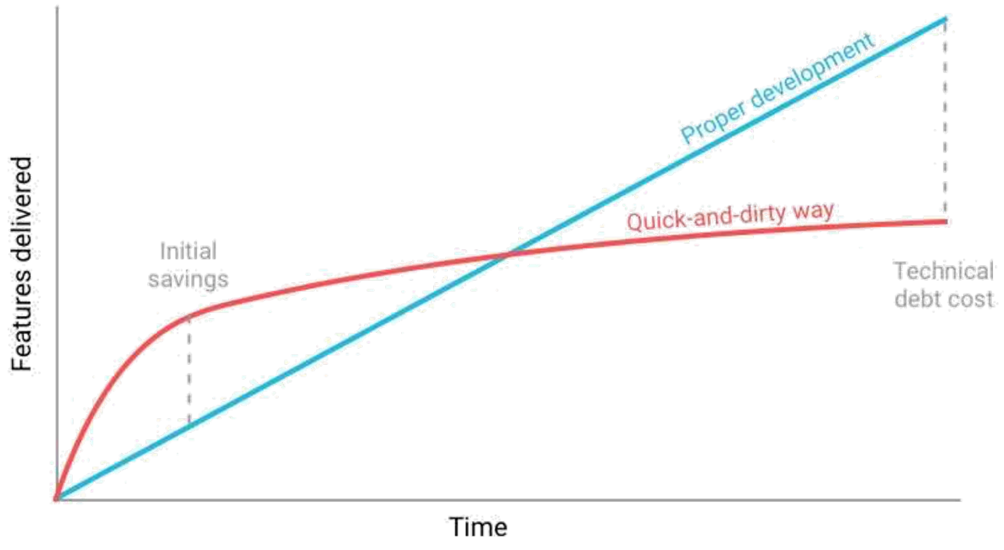


The origin of a dooming decision…

It was bad! We accumulated Technical Debt

No separation of concerns in the code
- intertwined codebase with preprocessing, feature code, model loading - unloading – serving

No production-ready MLOps tools to use
- No easy way to automatize model testing and retraining
- No easy way to check for data/concept drift



Source: https://accesto.com/

TD is specified as "*design or implementation constructs that are expedient in the short term, but set up a technical context that can make a future change more costly or impossible*" Dagstuhl seminar 16162.

The “just a binary” antipattern (2021)

In this antipattern paradigm, the data scientist develops a model offline and hands it over to IT for deployment. IT treats the model like any other third-party software library by writing against its application programming interfaces (APIs).

This antipattern is appealing:

1. **It isolates data science from IT**. Most data scientists lack the software engineering background for production. IT lacks the background in ML to understand model development. If these teams work in isolated departments, the antipattern allows for a smooth handoff.
2. **It simplifies accountability**. IT (understandably) doesn’t want to be accountable for the output or accuracy of a model.
3. **It avoids new infrastructure investment**. It gives IT the ability to deploy the ML in any modern infrastructure. No new tools or cloud platforms are necessary.

Nowhere to be found in the literature

10.13.2021

Kevin Dewalt

CHIEF EXECUTIVE OFFICER & CO-FOUNDER

PROLEGO

Why this antipattern is dangerous

Technical debt & slow iteration

- A focus on "the model" usually indicates that an engineering team has **limited experience with ML**. Risk to underestimate how much additional infrastructure is needed to support MLOps.

- The antipattern may work for a **beta deploy for relatively simple, static problems**. The engineering team begins to accumulate technical debt.

MLOps require:

1. Ongoing access to fresh training data
2. Feature scaling and reuse
3. Input data monitoring
4. Model output monitoring
5. Model orchestration and versioning
6. Pipelines

A few examples:

- The model drifts because it runs on data that evolved after training. Not easy to recognize without supporting infrastructure (test debt)
- The data scientists and engineering team lose track of model versions and the associated training data (version debt - Data Crisis as a Service Antipattern)
- Data scientists add new features to a model, but the feature engineering pipeline isn't put into production (code debt)
- Data scientists build models and experiment in notebooks, but the code isn't migrated into more robust software packages (code debt)

The “Act now, Reflect Never” Antipattern*

Once models are placed in production, predictions are sometimes used as-is without any reflection, or even periodic manual inspection.

- Like us for the first year and a half until the rollout to production
- End users used as testers

It is quite often the case that the statistical properties of the training dataset and the target variable change over time

- Prior shift
- Covariate shift
- Concept shift

Test debt, ranging from naive omission of basic sanity checks to the lack of more sophisticated tests to assess data quality or distributions

It is important to have (explainable) systems in place that can monitor, track, and debug deployed models.

*N. Muralidhar, et al. “*Using antipatterns to avoid mlops mistakes*”. arXiv preprint arXiv:2107.00079 (2021).

Lessons learned

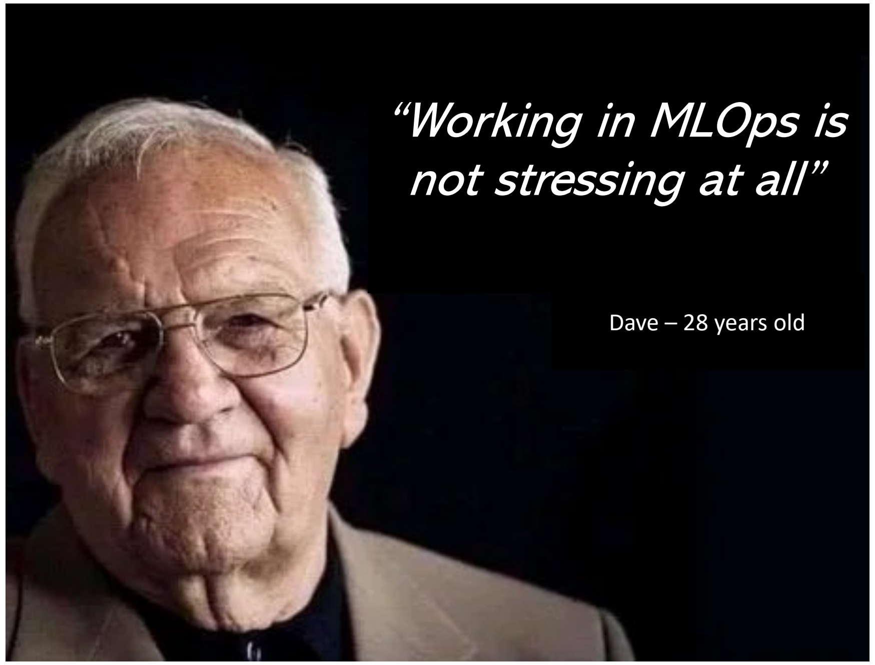
Cost of wrong early decisions can be extremely high

Support infrastructure for MLOps, dev culture, and separation of concerns are costly but necessary

ML is not fire-and-forget but it requires commitment (human-in-the-loop)

The bottom line is that the main issues that hinder the adoption of MLOps are:

- Lack of knowledge - MLOps requires both Dev and ML skills
- Lack of interest in the “other side” – “MLOps is seen as more work for developers”
- Lack of communication, trust, purpose
- Lack of innovation culture



Thank you!