Software Architecture Best Practices for Enterprise Artificial Intelligence

Yannick Martel[1], Arne Roßmann[2], Eldar Sultanow[3], Oliver Weiß[4], Matthias Wissel[5], Frank Pelzel[6], Matthias Seßler[7]

Abstract: AI systems are increasingly evolving from laboratory experiments in data analysis to increments of productive software products. A professional AI platform must therefore not only function as a laboratory environment but must be designed and procured as a workbench for the development, productive implementation, operation and maintenance of ML models. Subsequently, it needs to integrate within a global software engineering approach. This way, Enterprise Architecture Management (EAM) must implement efficient governance of the development cycle, to enable organization-wide collaboration, to accelerate the go-live and to standardize operations. In this paper we highlight obstacles and show best practices on how architects can integrate data science and AI in their environment. Additionally, we suggest an integrated approach adapting the best practices from both the data science and DevOps.

Keywords: Software Architecture; Enterprise Architecture; Machine Learning; Artificial Intelligence; MLOps

1 Introduction

Despite gaining more and more popularity (and hype), AI technology is still in an infancy stage and may be intimidating to companies with no AI know-how and experienced personnel. AI use cases involve many challenges and often fail to scale and translate into productive solutions [Wh19]. In addition, specifically data science projects are very deep and challenging in terms of content. Their level of industrialization is not comparable to what we have reached for other business applications. While being at the core, the machine learning part used in the enterprise environment represents only a fraction of the overall system, but machine learning still put some strong constraints on the overall architecture. This can

[1] Capgemini, 147 Quai du Président Roosevelt, 92130 Issy les Moulineaux, France, yannick.martel@capgemini.com
[2] Capgemini, Bahnhofstraße 30, 90402 Nuremberg, Germany, arne.rossmann@capgemini.com
[3] Capgemini, Bahnhofstraße 30, 90402 Nuremberg, Germany, arne.rossmann@capgemini.com
[4] Capgemini, Mainzer Landstraße 180, 60327 Frankfurt, Germany, oliver.weiss@capgemini.com
[5] Capgemini, Olof-Palme-Straße 14, 81829 Munich, Germany, matthias.wissel@capgemini.com
[6] IT Dept. of Germany's Federal Employment Agency, Südwestpark 26, 90449 Nuremberg, Germany, frank.pelzel@arbeitsagentur.de
[7] IT Dept. of Germany's Federal Employment Agency, Tafelhofstraße 4, 90443 Nuremberg, Germany, matthias.sessler@arbeitsagentur.de

doi:10.18420/inf2020_16

result in problems such as technical debts that prevent further development, modification or even operations of the system [Sc15].

Project participants and customers therefore face difficulties that they are not familiar with, for example, from "traditional" enterprise information system projects. These new challenges include:

(1) Model rules and logic exist outside the well-known code base. This makes it difficult to encapsulate and control the behavior of the whole system. In addition to code dependencies, dependency on data is also introduced when using Machine Learning Models.

(2) The model changes its behavior over time. This is referred to as model drift and can lead to a decrease of quality for predictions by a model that indeed is successfully deployed in production but is decaying in its abilities.

(3) A mix of frameworks, different programming concepts and languages is involved. The detection of semantic errors is much more difficult in such an environment, where each component has been selected as optimal for one function but might not fit well in the overall architecture.

Integrating AI into existing systems is a process that is complicated not only due to technical intricacies but also due to the organizational aspects. The data science team primarily aims at accelerating and simplifying experimentation and innovation based on machine learning and data analytics. They commonly work in a fast-paced prototyping and rapidly iterate, test new approaches and explore new areas. The focus of enterprise software development team, on the other hand lies on stable and reproducible software releases. Both disciplines follow different workflows and incorporate different tools.

It seems difficult to incorporate the dynamic nature of ML model development and training into an end-to-end software development cycle in a reproducible fashion. We commonly see a lack of expertise in software engineering in data science teams and experience for corresponding software architectures. On the other hand, IT departments struggle to understand the specificities of machine learning based software systems, and to integrate the exploratory and empirical approaches favored in data science. Some approach is needed to incorporate the two corpuses of knowledge and practices and to apply it to our new AI-based systems.

In our paper we first summarize related work examining the integration of ML components into large enterprise software systems in section 2. Section 3 then presents the basic methodology of comparing case studies which we will use to sharpen the problem domain, followed by a detailed analysis of several case studies in section 4. In section 5 we show a best practices approach for enterprise software development incorporating ML parts by discussing suitable processes, team lineup and eventually presenting our Machine Learning Reference Architecture. Finally, section 6 presents conclusions, limitations and future research by comparing the discussed approach with other solutions such as commercial platforms.

2 Related Work

The problem of moving machine learning into production has been mostly faced by the large web players of the AI generation, whose business model is built on large-scale automation and massive use of machine learning. Typically, Google, Amazon, LinkedIn, Facebook, Lyft and their colleagues have developed and approached software to face the challenges they encountered. Some have published their learnings, like Google in [Sc15], Facebook in [Ha18] or Uber in [HD17]. Most of the work is on large to very large-scale infrastructure, adapted to the volume of data and size of teams of these players.

MLOps is now one of the *5 Hands-on Skills Every Data Scientist Needs in 2020*[8], and is part of ML modernization movement. Many vendors such as DataRobot, Dataiku, SAS, SAP, ParalellM, and of course the cloud platform vendors such as AWS, Google Cloud Platform and Microsoft Azure have viewpoints, recommendations and reference architectures.

3 Methodology

We chose an exploratory approach and used a case study analysis to identify and categorize architectural shortcomings and problems which occurred in real practice of AI projects. These practical cases include:

- Federal Employment Agency (FEA): An operational analytics tool visualizes and predicts cpu and memory utilizations including peak and idle times [Ch19].
- BSH Hausgeräte GmbH: A novel approach for predicting and visualizing distributed product information queues at BSH Hausgeräte GmbH [Ch20].
- Federal Employment Agency (FEA): The use of Natural Language Processing (NLP) to classify documents. In the case, student certificates are evaluated by our machine learning models and the results are made available to the processor.
- Premium automotive OEM: Anomaly detection in a car fleet.
- Factoring branch of a French bank: Integration of scoring algorithms into productive environment
- A fraud detection startup in need of running machine learning models in a real time streaming environment.

[8] `https://www.kdnuggets.com/2020/01/odsc-5-skills-every-data-scientist-needs.html`

- Aircraft maintenance and material support analytics: Building a robust and stable data pipeline for many different data sources

To resolve this, we interviewed experts involved in these projects and based on their feedback identified and worked out the problem core in order to cluster problem domains which can be subsequently addressed through our solution approach.

4 Case Study Analysis

4.1 Federal Employment Agency (FEA): Operational Analytics

This case develops a novel tool for data center management that incorporates data visualization and machine learning capabilities for a large government agency in Germany, which hosts three highly available data centers containing more than 10,000 servers [Ch19]. The solution comprises a web-based 3D prototype running on Node.js. The tool provides a significantly better option and enabled visualization of historical data for all server instances at the same time, as well as real-time charts. It also uses the full potential of machine learning for time series forecasting.

There were quite a few challenges faced during implementing the ML-based tool. Setting up the Python environment including all needed dependencies was time intensive. Development of new functions which required new dependencies (which in worst case conflict with existing ones) would block other developers to restart the system after an update. Anaconda was not required at the beginning but later became essentially the only solution to get the system running. At the end containerization simplified a lot, as it allowed easy encapsulation of the dependencies and smooth installation.

4.2 BSH Hausgeräte GmbH: Predicting and visualizing distributed product information queues

The objective of the project was to present processing state and queues in a more understandable and predictable way and to visualize them in order to enable improved decision-making.

When implementing the solution, many challenges came up: there existed many versions of Python scripts, notebooks etc. Defining interfaces was not that easy as we know this in the JEE world. The production environment consisted of a complex infrastructure and many pieces had to be put together before having a running system. Sometimes a small change led to issues where other developers were unable to run the system (as it ran one day before) as they needed to adjust parameters and paths manually etc. The overall build process was not easy and transparent, for details see [Ch20].

4.3 Federal Employment Agency (FEA): Natural Language Processing (NLP) for Confirmation of Studies

There are vast amounts of text documents in public administration, which opens great potentials for the use of ML-algorithms. In our case, certificates, which are uploaded to claim child benefits, are now evaluated by machine learning models. Especially classification and automatic information retrieval foster a simpler, faster and more beneficial way of processing these data.

As this had been the first use case at FEA which had been deployed to production, there were some initial efforts needed. E.g. once we had to recode from Python to Java to hand over the model. Today, we develop and deploy in Python. Other efforts are organizing and implementing a monitoring and feedback loop for the first time, especially if this causes some modifications to operating applications. A further, at first glance trivial aspect, is the organisation of the collaboration of the numerous stakeholders for the first time - i.e. to enable IT-Experts, Business Analysts, Product Owners, Data Scientists etc. to work together with different access, views etc. and within the data science team with script and model management, versioning etc. Due to performance and scalability issues in other use cases, the need for upgrading the laboratory to a ML-factory by an on-premise stack become inevitably. The requirements to take a platform decision are listed in chapter 5.5

4.4 Premium Automotive OEM

Anomaly detection system: All vehicles during the ride are regularly sending data to the endpoint in the cloud. However, often due to technical problems, some data is lost or delayed. The customer wants to be informed as soon as possible about problems with collecting the data.

The solution consists of three models underneath, each one specialized in detecting different kind of anomalies: Linear Learner is an AWS SageMaker built-in model which was used for making prediction for next 24 hours, and each hour a check whether incoming data is within confidence interval of prediction was made. Autoencoder is a neural network, which finds the function would follow the general pattern and would be insensitive to sudden deviations. Then the difference between incoming data and denoised function is assessed using Local Outlier Factor to decide whether observation is anomaly. For retraining purposes, scripts for data preprocessing and training, which must be run by user, were prepared. In the future, there will be a module for automatic retraining.

4.5 Factoring branch of a French Bank

In this case the factoring branch of a French bank required a quick integration of two ML algorithms giving scorings used by a digital platform into production: One for fraud probability (using graph analysis; for instance if the bank client is linked to many known fraudsters, it may be a fraudster) and another about default of payment probability (if the amount of transaction of the client is very different of the usual amounts seen for this client, the score is higher).

Using a DevOps pipeline, each algorithm has been delivered on production four months after the delivery by the data scientist. The software engineering team decided not to reimplement all the preparation procedure but to reuse the SQL scripts already done and to optimize only the scripts that needed to. This way the work was easier because data scientists and software engineers used the same language. The result of the preparation process was exposed in a NoSQL Database to be used by a microservice with low latency. The prediction was done by a Java microservice merging the data from the NoSQL database with the arguments sent in a SOAP query to call a Python script giving the score as result. This mixed architecture introduced many difficulties due to the closely coupled Python-integration with Java. As key learning we would recommend exposing Python scorings using Python (micro-) services with frameworks like Django[9] for example. We also had difficulties to manage Python packages environments and we started to implement a dependency management automation process to avoid this issue on a follow-up project.

All logs (during the preparation process or during the microservice call) were sent to an ELK cluster and visualized via technical and business dashboards to follow the algorithm usage. A difficulty concerning this kind of monitoring was to train the hosting team to manage all the parts of the project. The team's background was mainly the management of batch loadings for BI and so we chose to hide the technical complexity of Big Data technologies using a framework. In addition, we created trainings for the ops team with reference to BI to give them some examples they already knew.

4.6 Fraud detection at Fintech startup

This case is an example of moving machine learning from the lab to a production environment. This young fintech has developed a machine learning model for detecting payment fraud in real time, based on behavior fingerprints of clients, accounts, banks and payments. This model has been proven in the lab on historical data, with big data tools and a mix of Python and Scala scripts.

Then the model had to be introduced in a real-time process, together with data preparation and feature engineering. Therefore, the fintech has developed a real-time streaming engine supported by a Kafka unified log, with the following steps:

1.) Collection and conversion of transactions and auxiliary data onto a common data model; 2.) Enrichment of transactions with auxiliary data – for instance addition of client profile and bank profile to the transaction; 3.) Calculation of features for the machine learning

[9] `https://www.djangoproject.com`

model – including short-term (1 minute, 1 hour, 1 day) and longer term (1 year, 1 month...) aggregates based on different keys (account, client, country...) and selection criteria (failed transactions, instant payment...). These behavioral features require the maintenance of pre-features in an in-memory database. This move from a batch lab environment to real-time has required redesign and redevelopment of all the algorithms; 4.) Application of machine learning models, using the same library as during the lab, but with its Scala bindings; 5.) Application of configurable business rules, to assess the risk and make a final decision.

This effort has been done in a separate environment, with independent technologies, and with a pure DevOps / Agile development methodology and tools. The data science team can now develop new models and integrate them in the developed framework.

4.7 Aircraft maintenance and material support analytics

This case is an example for reproducible analytics insights and constantly growing data pipelines to enable end users making data driven decisions in different areas surrounding the operation of military aircrafts. The focus is the aircraft maintenance, where the provided dashboard help support engineers perform root cause analysis of errors and part failures. Another area is material services where the tracking of parts within the repair or maintenance cycle is enabled by dashboards.

Starting with the many source systems on end customer side as well as in house, the data needs to be ingested and further processed on a single platform to leverage the full potential of the collected information. Due to the iterative process and increasing number of included source systems, the challenge of delivering reproducible and consistent results in the required dashboards for the different user groups is a constant challenge. Particularly due to the use of the Hadoop stack, which comes with great processing frameworks like Apache Spark but also the Hadoop Distributed File System (HDFS) which prevents the use of common already established data versioning tools. Other challenges to build a robust and stable data pipeline stem from the different frequencies and volatile quality of the data deliveries from the source systems.

To overcome these issues, a data versioning system was established that connects the version of the code for the processing logic with the used data delivery. Also, a monitoring system for the data quality and consistency of the derived insights was established to be able to spot data issues as soon as possible and provide initial clues for further investigation.

5 Developing Best Practices for a Data Science Architecture

In the following, we distil the best practices from the case studies. This extraction of best practices forms the actual outcome of the performed case study analysis. It is precisely a responsibility of Enterprise Architecture Management to detect these pitfalls and recipes for success and to anchor them in the organizational memory. We map above mentioned pitfalls into following categories:

- *Organizational*: New professions from the field of big data seem to be taking a firm place in organizations, but their tasks in projects or enterprise programs, and at the intersections to other departments, have yet to be sharpened.
- *Procedural*: New tools and ways to create solutions in the data science environment also bring new problems that hinder agility, high level of data quality and integration into the solution building processes at enterprise level.
- *Architectural*: Well working solutions in lab environments often fail to scale and deliver sufficient performance in real world scenarios or an explosion of technical complexity prevents an integration into manageable products.

5.1 Requirements

First and foremost, governance and leadership are needed which will be considered as critical to move machine learning to production and the introduction of data science into operational processes. This innovation requires the transformation and adaptation of software development processes and thus the leadership and vision to support that. Subsequently, we must integrate data science experiment-oriented activity within the software engineering activities, oriented towards developing industrial-grade software systems. This is a clash of cultures, as well as skills and practices. Finally, we need architectures, technical tools and environments to support an end to end processes, encompassing exploration, prototyping, model development and training, then allowing easy transition into industrial software.

The tooling must allow for automation and repeatability. We also need to support serving and monitoring of machine learning models, both from a technical and business standpoint. The architecture patterns must cover different types of machine learning models in different contexts, varying with the business, the deployment constraints, the temporality (real-time vs slow processing) and the volume of data to process.

In order to cover existing and future requirements of developments in the field of Advanced Analytics and Machine Learning, there is also a need to upgrade and expand the analytics platform. It requires an integrated system of coordinated hard- and software in the local data center. The hardware must be designed to meet the data science specific requirements This results in particular

- due to increasing amounts of data as well as significantly increased demand for computing capacity because of the growing complexity of the methods and algorithms used.

- from the increased demands in the development of analytical models with regard to data protection and security, data ethical principles and legal framework conditions (GDPR, EU Whitepaper on AI).
- from the constantly increasing amount of machine learning increments used in productive operation, requiring larger and more flexible functionalities in terms of data and workflow management, interfaces and monitoring of data and models.

Finally, machine learning requires a high degree of connectivity within the organisation. This places completely new demands, but also opportunities on IT security, see [Be19]. A framework that allows flexibility in tooling and methodology while maintaining the integrity of sensitive data within the organization.

5.2 Defining a MLOps process

In order to overcome the above-mentioned problems and satisfy the requirements, we propose viewing developing ML applications as a software engineering activity, thus in need of a software engineering process. We adapt best practices from the Agile as well as the DevOps movement. Main characteristics are:

- Release early, release often
- Short adaption cycles, small safe increments
- Strong version control
- Testing in every stage and automatically
- Automated integration and deployment (CI / CD)
- Reproducible processes and reliable software releases
- End to end responsibility of teams from Dev to Ops

Fig. 1 shows a typical DevOps process which is basis for continuous integration (CI) and continuous delivery (CD), see also [SWW19].

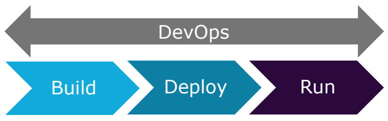


Fig. 1: Typical DevOps process

For ML applications the process for continuous development and deployment gets more complex, because of additional types of artifacts which are subject to change:

(1) Data: A model is chosen, trained and operated upon data, whereas changes in schemes or attribution can occur any time.

(2) Model: Algorithms, parameters and hyperparameters representing the model behavior as outcome from experiments and training over time

(3) Code: Primary artifact in classic software engineering including all kinds of source code, scripts, tests

The idea of Continuous Delivery must adopt these dimensions in order to create a repeatable and reliable software release process including Machine Learning artifacts. We refer to this extension of DevOps as MLOps.

Fig. 2 shows the proposed Software Engineering process which is extended by the steps "Define" and "Train" to deal with above mentioned additional dimensions of data and model. This way we add following feedback loops to the process in order to integrate central ML lifecycle steps:

- Define and build a suitable model and adapt based on demo feedback through experiments
- Include training the model and adapt based on observed model performance
- Retrain an operational model based on new real-life data and performance
- Adapt result of the whole process based on live feedback, preventing model drift

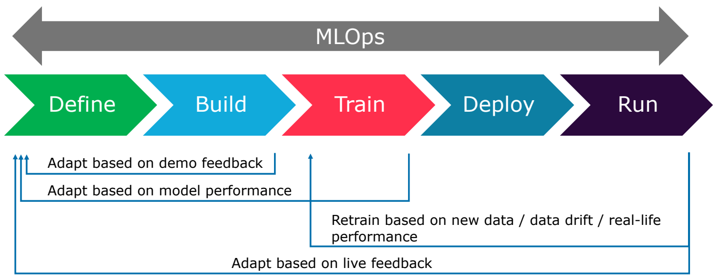
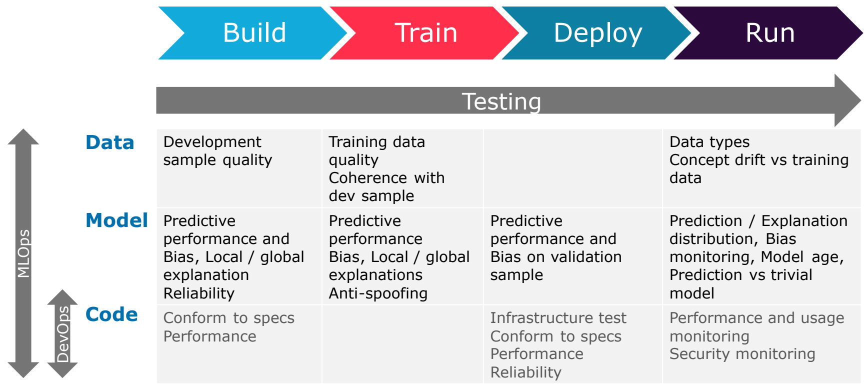

MLOps
Define
Build
Train
Deploy
Run
Adapt based on demo feedback
Adapt based on model performance
Retrain based on new data / data drift / real-life performance
Adapt based on live feedback

Fig. 2: Extension of the DevOps process to proposed MLOps

The resulting process allows reproducible small increments for reliably releasing machine learning applications in short adaption cycles. This enables managing the additional dependencies on changing model behavior and data (see problems mentioned in chapter 1) allowing continuous model training und monitoring model performance in production. In our opinion, this is crucial for preventing model drifts by iteratively checking for performance degradation with the ability to re-train the productive model or implement a new one based on new key assumptions.



	Build	Train	Deploy	Run
Data	Development sample quality	Training data quality Coherence with dev sample		Data types Concept drift vs training data
Model	Predictive performance and Bias, Local / global explanation Reliability	Predictive performance Bias, Local / global explanations Anti-spoofing	Predictive performance and Bias on validation sample	Prediction / Explanation distribution, Bias monitoring, Model age, Prediction vs trivial model
Code	Conform to specs Performance		Infrastructure test Conform to specs Performance Reliability	Performance and usage monitoring Security monitoring

Fig. 3: Tests in every phase of proposed MLOps process

A specific mention concerns testing, as it is not a step somewhere in the process but is to be considered as integral part of the engineering procedure. As so, testing should be carried out in every step on every involved artifact, especially on the model and the data. Our proposed extension of testing in DevOps is shown in Fig. 3: In addition to tests on the code, there are test targets also for data and the model.

5.3 Cross functional teams and mix of competences

From the perspective of a classic ML lifecycle, the role setting of Business Analysts together with Data Scientists and Data Engineers is sufficient for conducting a working ML solution which proves to deliver all required benefits in a simple lab environment. For industrialization purposes in the context of complex enterprise products additional roles must be considered. Again, by applying concepts from the DevOps world these roles may include: Software Developers / ML Engineers, Architects, Project Managers, Designers and DevOps Engineers. These disciplines should form a cross-functional team and act together, else we observe the before mentioned problems of delays and friction, as is common in complex IT projects. This is due to the ownership of different phases of the process (see Fig. 2) to different roles where results are just handed over. We made good experience by creating cross functional teams that has experts from every discipline in order to cover the whole MLOps lifecycle and aim at a common end-result: an application live, serving users and Business.

5.4 Reference Architecture for MLOps

In this chapter we propose a data science reference architecture that works as a blueprint when creating ML solutions using the MLOps approach. This blueprint architecture is part of an overall Data Analytic architecture with three large blocks, see Fig. 4.

- The platform foundations form the common platform elements, providing infrastructure for the other functions; they address the raw storage, raw processing, software engineering DevOps tooling and automation, and security;
- The AI, Analytics & BI Foundations provide tools for performing the processing and analysis of data;

The AI, Analytics & BI Execution block host the custom-made as well as the off-the-shelf algorithms and applications. The AI, Analytics & BI Foundations contains the MLOps foundations, which incorporate common building blocks addressing specific dimensions that form a holistic ML project approach in enterprises. The many commercial or Open Source products and tools address one or more blocks. In this MLOps foundations, we identify the following functional areas:

- An AI Marketplace allows to download and incorporate reusable models and AI libraries, even reusable data sets.
- Analytics Monitoring to supervise models in development and production in order to prevent model drifts or install a trigger on when re-training is needed.
- Analytics Orchestration provides management of data pipelines as well as mechanisms for serving of models and testing them.
- The Data Science Workbench is the development environment for data scientists including management of experiments
- The Model Repository enables the management of the whole lifecycle of machine learning models and stores the various versions together with metadata
- The Data Modelling enables data preparation and features engineering

On the right side of Fig. 4, we located the AI, Analytics & BI Execution which describes platforms solutions that encapsulate and hide many or all the building blocks from the left side in order to deliver ready-to-use AI services for users who don't want to deal with connecting single blocks to the solution themselves. Examples are AI Platform as a Service or API as a Service offerings by Cloud vendors. These products are often referred to as low-code platforms. Following functional areas include:

- Custom exploration providing fixed toolsets for preparation and gaining insights into several kinds of datasets.

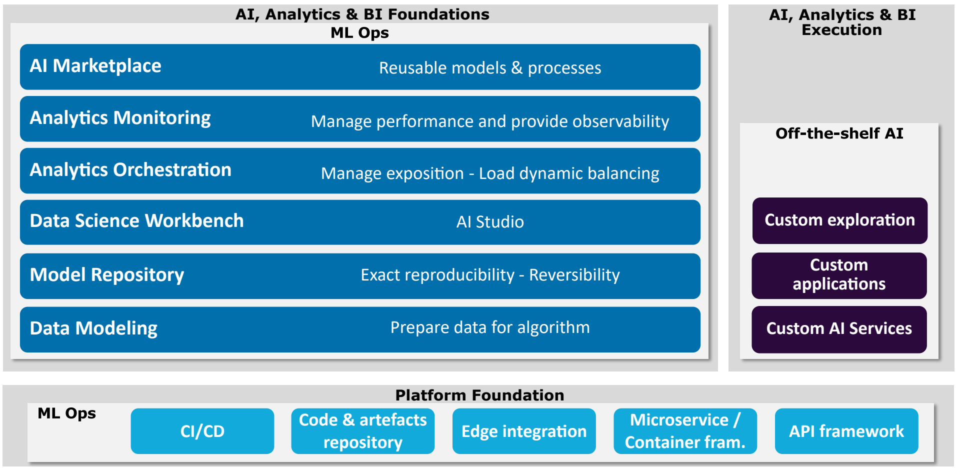

AI, Analytics & BI Foundations
ML Ops
AI Marketplace
Reusable models & processes
Analytics Monitoring
Manage performance and provide observability
Analytics Orchestration
Manage exposition - Load dynamic balancing
Data Science Workbench
AI Studio
Model Repository
Exact reproducibility - Reversibility
Data Modeling
Prepare data for algorithm
AI, Analytics & BI Execution
Off-the-shelf AI
Custom exploration
Custom applications
Custom AI Services
Platform Foundation
ML Ops
CI/CD
Code & artefacts repository
Edge integration
Microservice / Container fram.
API framework

Fig. 4: Reference Architecture containing MLOps-related building blocks

- Custom applications that offering predefined AI solutions for a certain scope of business aspects
- Custom AI Services that can be called to process AI tasks as part of a broader business solution or product.

Platform Foundation forms the bottom part of the proposed architecture defining building blocks that relate to infrastructure and cross-functional aspects:

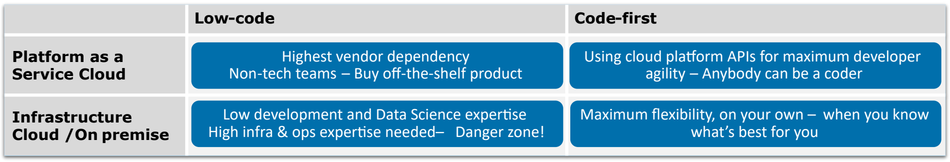
- CI/CD platform allowing to implement the proposed MLOps process chain in order to supply repeatable and reproducible results
- Code and artefacts repository facilitating version control of model data and code
- Edge integration enabling us to include local computing-, sensor- and other IoT-devices in our AI solution
- Microservice / container framework as a standardized execution environment
- API framework that supports the microservice approach in encapsulating and providing access to AI functionality

5.5 Choosing the right Architecture Approach

We discuss different scenarios on how the before mentioned Architecture can be used to choose an approach for creating AI solutions in a concrete environment, see Fig. 5. A first question addresses the ability to create IT-based solutions and understand underlying mathematical and technical mechanisms. Organizations that have teams with such a background can pursue the Code-first approach and take over the responsibility to create and deliver suitable business benefits on their own. A Low-code approach on the other hand offers products and platforms suitable for non-developers such as business-minded citizen data scientists but come with higher vendor dependency and limited solution space.

The second dimension describes to what extent the used building blocks can be customized and their associated flexibility and adaptability. Platform as a Service (PaaS) solutions offer a great variety of ready-made solutions but are limited when they must be adapted to different usage scenarios. They also lock you into a proprietary solution with more or less proprietary interfaces. Going direct over an infrastructure, either on the cloud or on premise provides more flexibility and offer a greater accuracy to fit to the needed solution. Unfortunately, it comes with much higher efforts to form the appropriate product, and require stronger expertise in architecture, development, tuning and operations.

There is no “right choice” here and we hope the simple diagram can help make the first step for making a choice in each case – or choosing to combine approaches for an optimal combination.



	Low-code	Code-first
Platform as a Service Cloud	Highest vendor dependency Non-tech teams – Buy off-the-shelf product	Using cloud platform APIs for maximum developer agility – Anybody can be a coder
Infrastructure Cloud / On premise	Low development and Data Science expertise High infra & ops expertise needed– Danger zone!	Maximum flexibility, on your own – when you know what’s best for you

Fig. 5: Scenarios when adopting AI solutions

5.6 Summary of best practices

The discussed best practices from this chapter can be summarized by again using the before mentioned problem categories: ‘Organizational’, ‘procedural’ and ‘architectural’. Organizational best practices start with governance and leadership: It is necessary to align a cross-functional project team and establish a common vision, but also to solve differences with respect to skills, practices, tools and environments that are used. In addition, data protection and security, ethical principles or legal factors such as GDPR play an important role that is often underestimated.

Second, as the process for continuous development and deployment is getting more complex for ML applications, we advise to combine elements from both, the Agile as well as the DevOps movement. Therefore, we proposed a process that allows reproducible small increments for reliably releasing ML applications in short adaption cycles.

Last, a sound reference architecture as well as architectural approach complete the picture. The building blocks for this consist of the “platform foundations” that provide the infrastructure for other functions, the “AI, Analytics & BI Foundations” that provide the tools for performing the processing and analysis of data, and the “AI, Analytics & BI Execution” block that hosts the custom-made as well as off-the-shelf algorithms and applications.

6 Conclusions and future research

The analysis of cases presented in section 4 and the extraction of best practices by chapter 5 provides a consolidated set of reusable templates for approaching architectural challenges in large AI projects as well as a general set of pitfalls that need to be considered during realizing such projects.

Moving machine learning to production is necessary to avoid the Death Valley of technologies which were the latest hype one day but failed to fulfill their promises. ML has high potential and can transform an organization into a leader in its field, as it has done with Google, Amazon, Netflix, but also Experian and Amex. These organizations did not become leaders because they were the first or because they lacked competition – they did because at the core, they developed the expertise of applying machine learning each to its field. As organizations are struggling to succeed on their markets and overcome competition, machine learning has the potential to improve their chances for success.

As we have shown through the examples, many organizations are experimenting and doing true innovation – that is doing new things and bringing them to serve – with machine learning in production. Tools and processes are not yet mature, but they are improving fast. Software engineering is providing solid foundations and making the Agile organization a reality. The case studies of this report and the experiences of the authors show that combining practice of software engineering and machine learning can realize enterprise grade solutions in a industrialized fashion like we are used to through DevOps.

Many of the above-mentioned case studies where in dedicated environments. You can go a long way with Open Source, but cloud platforms and commercial products are easier paths, especially if you are in a hurry and/or do not have the full technical competencies. Both ways have (dis-) advantages. However, the ability to integrate new and updated Open Source tools is a key requirement in view of the rapid development ongoing in this field. Cloud adoption will probably hugely facilitate the move of machine learning to production by providing easy access to supporting tools with low administration or installation effort. Even then, the dominant factor in the maturation of machine learning in production will be the confirmation and the democratization of best practices, dedicated tools and architecture patterns, based on the first experiences.

In future research, our model could be extended by considering commercial cloud vendors for AI or mixing with Open Source and other cloud vendor tools. First insights suggest that considering wide use of Open Source may increase the control of the overall architecture (and therefore solution fitting accuracy), while the collaborative support of large or distributed teams might be easier on commercial platforms. Especially when it comes to platform foundations and AI execution, commercial platforms seem to add significant value for ML projects compared to a “traditional” Open Source setup, often seen in pure DevOps solutions. Experience has yet to be gained here and should be topic for further research activity.

References

[Be19] Ben Zid, I.; Parekh, M.; Waedt, K.; Lou, X.: The application of Articial Intelligence for Cyber Security in Industry 4.0. In (Draude, C.; Lange, M.; Sick, B., eds.): INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft (Workshop-Beiträge). Gesellschaft für Informatik e.V., Bonn, pp. 255–260, 2019.

[Ch19] Chircu, A.; Sultanow, E.; Baum, D.; Koch, C.; Seßler, M.: Visualization and Machine Learning for Data Center Management. In (Draude, C.; Lange, M.; Sick, B., eds.): INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft (Workshop-Beiträge). Gesellschaft für Informatik e.V., Bonn, pp. 23–35, 2019.

[Ch20] Chircu, A.; Sultanow, E.; Hain, T.; Merscheid, T.; Özcan, O.: Real-Time 3D Visualization of Queues with Embedded ML-Based Prediction of Item Processing for a Product Information Management System. In (Zimmermann, A.; Howlett, R. J.; Jain, L. C., eds.): Human Centred Intelligent Systems. Springer Singapore, Singapore, pp. 347–358, 2020.

[Ha18] Hazelwood, K.; Bird, S.; Brooks, D.; Chintala, S.; Diril, U.; Dzhulgakov, D.; Fawzy, M.; Jia, B.; Jia, Y.; Kalro, A.; Law, J.; Lee, K.; Lu, J.; Noordhuis, P.; Smelyanskiy, M.; Xiong, L.; Wang, X.: Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In: 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). Pp. 620–629, 2018.

[HD17] Hermann, J.; Del Balso, M.: Meet Michelangelo: Uber’s Machine learning platform, Sept. 5, 2017, URL: https://eng.uber.com/michelangelo-machine-learning-platform/.

[Sc15] Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.-F.; Dennison, D.: Hidden Technical Debt in Machine Learning Systems. In (Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; Garnett, R., eds.): Advances in Neural Information Processing Systems 28. Curran Associates, Inc., pp. 2503–2511, 2015, URL: http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf.

[SWW19] Sato, D.; Wider, A.; Windheuser, C.: Continuous Delivery for Machine Learning, martinFowler.com, Sept. 19, 2019, URL: https://martinfowler.com/articles/cd4ml.html.

[Wh19] White, A.: Our Top Data and Analytics Predicts for 2019, Gartner Blog Network, Jan. 3, 2019, URL: https://blogs.gartner.com/andrew_white/2019/01/03/our-top-data-and-analytics-predicts-for-2019.