View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Helsingin yliopiston digitaalinen arkisto

SN Computer Science (2021) 2:342
https://doi.org/10.1007/s42979-021-00726-1

ORIGINAL RESEARCH

Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product

Tuomas Granlund[1] · Vlad Stirbu[2] · Tommi Mikkonen[3]

Received: 24 December 2020 / Accepted: 24 May 2021 / Published online: 19 June 2021
© The Author(s) 2021, corrected publication 2021

Abstract

Agile software development embraces change and manifests working software over comprehensive documentation and responding to change over following a plan. The ability to continuously release software has enabled a development approach where experimental features are put to use, and, if they stand the test of real use, they remain in production. Examples of such features include machine learning (ML) models, which are usually pre-trained, but can still evolve in production. However, many domains require more plan-driven approach to avoid hazard to environment and humans, and to mitigate risks in the process. In this paper, we start by presenting continuous software engineering practices in a regulated context, and then apply the results to the emerging practice of MLOps, or continuous delivery of ML features. Furthermore, as a practical contribution, we present a case study regarding Oravizio, first CE-certified medical software for assessing the risks of joint replacement surgeries. Towards the end of the paper, we also reflect the Oravizio experiences to MLOps in regulatory context.

Keywords Mlops · RegOps · Machine learning · Healthcare · Medical software

Introduction

Agile software development, which has rapidly become mainstream, embraces change [1], as well as manifests working software over comprehensive documentation and responding to change over following a plan [2]. However, there are many fields where a balance between plan-driven and agile development is needed, to ensure that enough attention is paid to practices such as risk management [3]. Such factors are essential for safety–critical, regulated domains, including medical devices.

However, instead of considering balance between plan-driven and agile development in terms of risks [4], it is often perceived that fundamental contradictions exist between agile software development and regulations. To some extent, this is understandable, as regulatory bodies need time to certify things, whereas software development is more and more advancing towards continuous practices [5].

To further complicate the situation, applications that incorporate artificial intelligence (AI) and machine learning (ML) technologies are becoming popular due to their ability to build complex prediction systems. However, the process of continuous improvement is often complex and involves changes in the following areas: the application code, the model used for prediction, and the data used to develop the model. This also introduces further challenges for testing, verification, and validation [6], which in turn is affected by regulations.

This article is part of the topical collection "Artificial Intelligence for HealthCare" guest edited by Lydia Bouzar-Benlabiod, Stuart H. Rubin and Edwige Pissaloux.

Tuomas Granlund
tuomas.granlund@solita.fi

Vlad Stirbu
vlad.stirbu@compliancepal.eu

Tommi Mikkonen
tommi.mikkonen@helsinki.fi

[1] Solita, Tampere, Finland

[2] CompliancePal, Tampere, Finland

[3] University of Helsinki, Helsinki, Finland

Despite the popularity of machine learning medical applications in the public and scientific space, these approaches suffer from limitations when considering their release as medical products. For instance, a recent survey on using ML on medical imaging [7] points out that the majority of the included papers failed to present the necessary data that are needed for reproducing the study itself, which is far less demanding than passing conformity assessment process. Furthermore, regulatory bodies do not, in general, provide guidelines how to fulfill their requirements in the context of ML, but only approve or fail the product, based on data that the applicant provides. As this process is expensive, and to a large degree non-transparent, there are few studies on how to meet regulatory requirements in ML context, apart from reports from regulatory bodies (e.g., [8]) or case studies from device manufacturers.

In this paper, we first introduce continuous software engineering practices, and then extend these practices to the emerging concept of MLOps, or continuous delivery of ML features to operations. Then, we consider MLOps in the light of regulations and the development of regulatory-compliant medical systems. As a hands-on contribution-related to MLOps, we present a case study regarding Oravizio,[1] world's first CE-certified medical software for assessing the risks of joint replacement surgeries. The focus of the case study is on ML features and their inclusion under the umbrella of regulatory requirements and implications. Our paper contribution is twofold: first, we introduce the methodology in which we mapped the state of the art in MLOps pipelines to the development **practices** employed in the Oravizio application, to identify the gaps in regulatory practice, then we presented the corrective actions taken to enhance the maturity of existing pipelines to an appropriate level under the new MDR regulatory framework. Although the investigation and the corrective actions reflect, to a large degree, the particularities of the environment in which Oravizio was developed, the same procedure can be used in similar constrained environments where organizational, legal, or data ownership limitations prevent the use of full-fledged MLOps pipelines.

The paper is organized as follows. In Section "Towards MLOps", we provide the necessary background of the paper, by introducing the path from agile software development to continuous delivery and further to MLOps. Then, in Section "Regulatory Constraints for Medical ML systems", we consider the implications of regulatory constraints on MLOps. In Section "Case Study: Oravizio", we present the technical design of the case system. In Section "Discussion, we list key observations based on the case study and discuss the validity of the research. In Section "Conclusions", we draw the final conclusions.

Towards MLOps

The concept of agile software development has been evolving for over 2 decades. While the origins of the approach manifest the value of people over the value of tools [2], today at the center of agile development is a toolset that allows delivering as soon as new features are available [5]. The goal of continuous deployment is to enable continuous flow of value-adding software artifacts from the development to the actual production use with a quality assurance. A closely related concept, the DevOps approach [9, 10], can be described as a set of practices whose goal is to shorten the commit feedback cycle without compromising quality [11]. A continuous delivery pipeline, required to deploy software, consists of a set of tools that support the process, from code to delivery. These tools ensure that each stakeholder gets a timely access to what they need.

Machine Learning Lifecycle Challenges

Building on the success of continuous software development approaches [5, 12], in particular DevOps [13], it has become desirable to deploy machine learning (ML) components in real time, too. To this end, MLOps refers advocating automation and monitoring at all steps of ML system development and deployment, including integration, testing, releasing, deployment, and infrastructure management.

To understand the challenges related to MLOps, let us first explain the steps necessary to train and deploy ML modules [14]. As the starting point, data must be available for training. There are various somewhat established ways of dividing the data to training, testing, and cross-validation sets. Then, an ML model has to be selected, together with its hyperparameters. Next, the model is trained with the training data. During the training phase, the system is iteratively adjusted so that the output has a good match with the "right answers" in the training material. This trained model can also be validated with different data. If this validation is successful—with any criteria we decide to use—the model is ready for deployment, similarly to any other component. Once deployed, ML-related features need monitoring, like any other feature. However, monitoring in the context of ML must take into account inherent ML-related features, such as biases and drift that may emerge over time. In addition, there are techniques that allow improving the model on the fly, while it is being used. Therefore, the monitoring system must take these needs into account.

Based on the above, continuous deployment of ML features is often a complex procedure that involves changes in

[1] https://oraviz.io/.

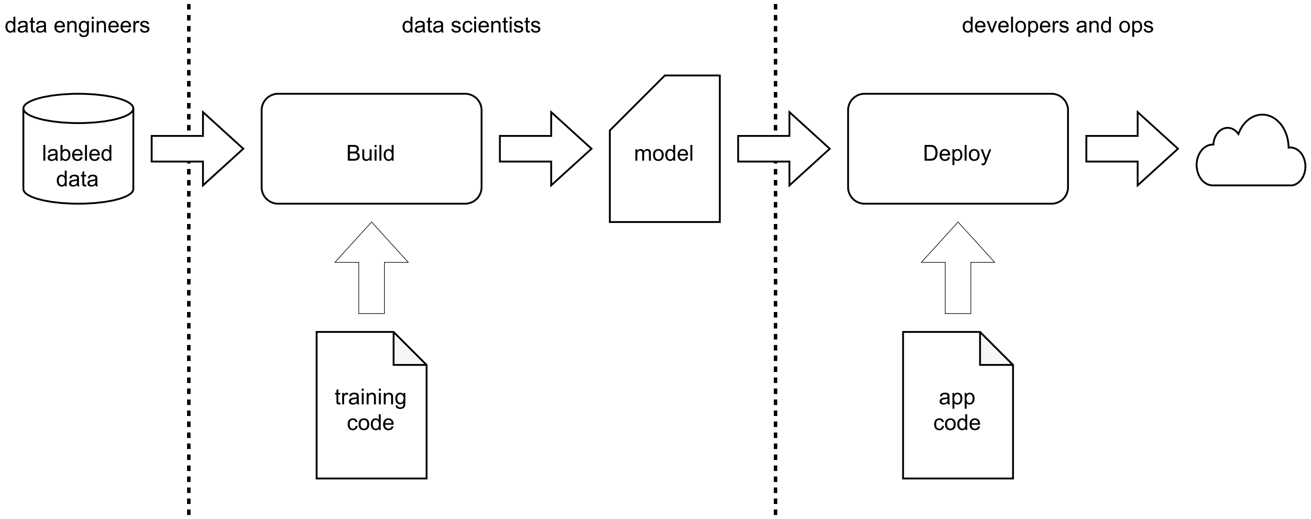

Fig. 1 Functional silos barriers when developing ML applications

the following areas: the application code, the model used for prediction, and the data used to develop the model. Often, these areas are handled separately by software developers, data scientists, and data engineers that rely on different skill sets and tool chains. For example, data engineers are focused on making the data more accessible, data scientists perform experiments for improving the data model, and the developers are worried about integrating the various technologies and releasing them to production (see Fig. 1). The lack of harmonized processes across these domains leads to delays and frictions, such as models never reaching production or deployments that are difficult to update or debug. Due to this variability, machine learning applications are more complex than traditional applications. These characteristics make them harder to test, explain, or improve.

General Purpose MLOps Pipelines

Continuous Delivery for Machine Learning (CD4ML) [15] is an approach formalized by ThoughtWorks for automating in an end-to-end fashion the lifecycle of machine learning applications. In CD4ML, a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles. The approach contains three distinct steps: identify and prepare the data for training, experimenting with different models to find the best performing candidate, and deploying and using the selected model in production. This is illustrated in Fig. 2.

The first step has the goal of making the data *discoverable and accessible*. It consists of collecting relevant data from different internal and external sources, transforming and exposing it in a format that is used by the data scientists to train the model. The data pipeline codifies the directed acyclic graph that contains the sources, the destinations, and the transformations performed on the data. As the source data used in this step can be very large, it is not practical to check it in version control. Instead, metadata that conveys the location, ranges, and other parameters that determine the shape of the data source used for training. Over time, the data can evolve over two axes: data schema or sampling frequency. Storing the pipeline, the source data metadata, and the code that performs the transformations is an effective data provenance mechanism.

The next step is to train model candidates based on the data collected in the previous step. The input data are split into training and validation data. The training data are used to evaluate combinations of algorithms that produce a model. The model is evaluated against the validation set to assess its quality. This process is codified as the *machine learning* pipeline. During development, the pipeline can change frequently and is difficult to reproduce the process outside the local environment without the assistance of specialized tools like Data Science Version Control (DVC)[2] or Pachyderm.[3] These tools provide git-like functionality that keeps track of data and code used in experiments, allowing execution on other environments. As most experiments do not yield good results, it is critical to preserve all data, metadata, metrics, and the code that captures how the experiment was conducted. This record supports the decision process for promoting a particular model to production. Tools like MLFlow Tracking[4] are designed to support this process.

[2] https://dvc.org/.

[3] https://www.pachyderm.com.

[4] https://mlflow.org/docs/latest/tracking.htm.

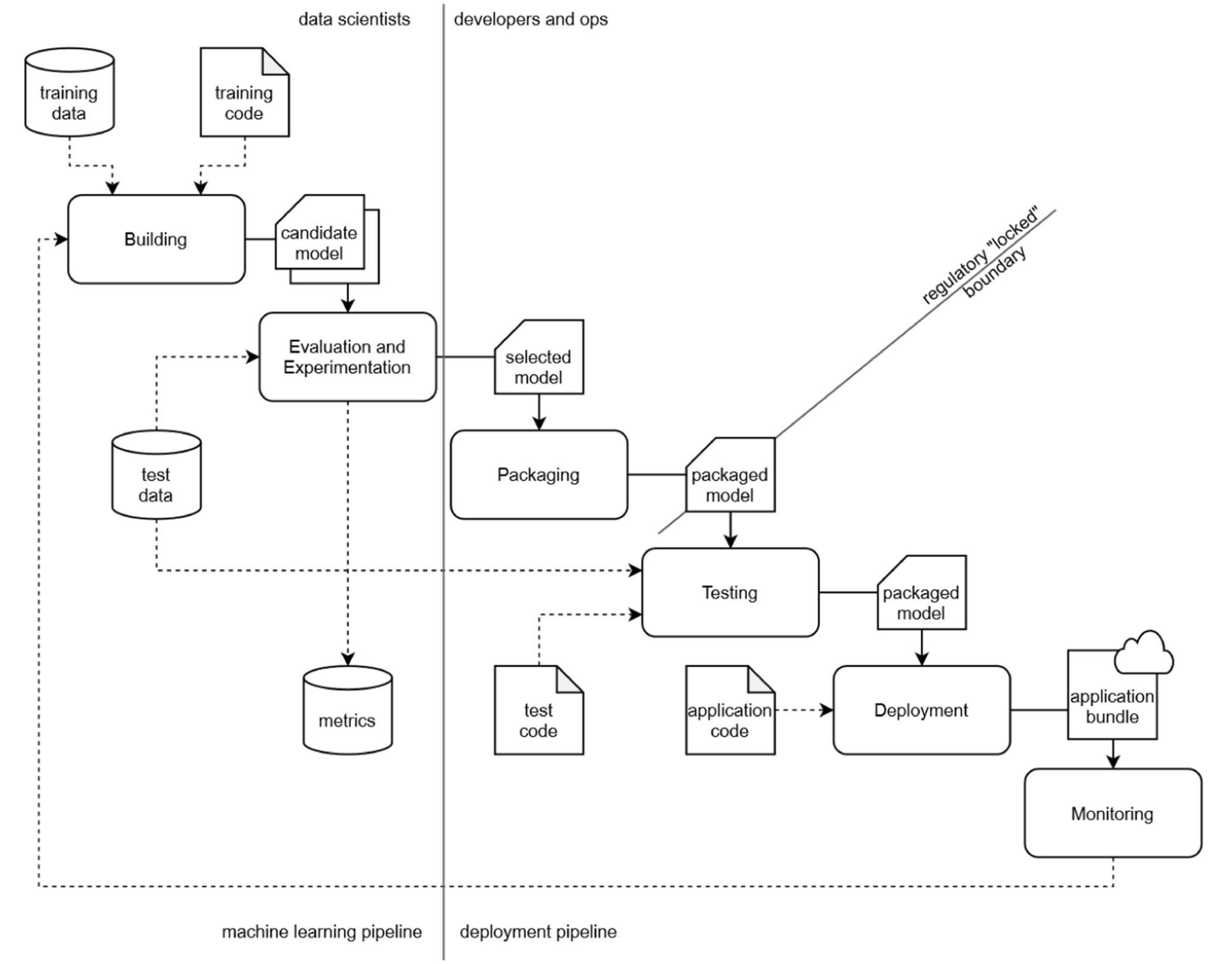

Fig. 2 CD4ML pipelines and artifacts

The final step is to deliver the model into the production environment using a *deployment pipeline*. The process consists of testing the model selected for production, packaging in the format suitable for production, followed by deployment. The deployment can follow one of the following patterns: include in the application code as normal dependency, run it as standalone service, or deploy at runtime as data. A special case of deployment is *online learning*, where the models constantly learn in production. Versioning the model as static artifact will not yield the same results as they are fed different data. Once in production, continuous monitoring ensures that the model behaves as expected, and anomalies are detected and handled properly. The feedback loop allows the model to be improved over time using observations from production environment. The "locked" boundary and the special understanding of monitoring from a regulatory perspective are addressed in more details in Sect. 3.2.

CD4ML is not the only solution that aims to prevent the accumulation of hidden technical dept in machine learning applications [16]. While other solutions are heavily optimized for a particular cloud infrastructure [17, 18], or an ML software stack implementation [19], the CD4ML model has the advantage that it can be used as a reference model in any application domain. Furthermore, it has the necessary phases documented at the appropriate level, and the implementation is based on open source components. These qualities enabled us to effectively identify the following new parts of the delivery pipeline that need to be considered from a regulatory software development lifecycle perspective, to be able to continuously deliver ML components to production:

- model and its versioning,
- different data sets used for training model and their versioning,

- monitoring the output of the model to detect bias and other problems.

In addition, parts relying on ML need to go through the same tests and staging process as any new software features. This part of the process will be largely overlooked in this paper, where we focus on new parts of the pipeline in a regulated context.

Regulatory Constraints for Medical ML systems

Previous work has shown that the relationship between regulatory actions and continuous deployment is not straightforward [20–22]. On one hand, risk management is easily overlooked in the process of continuously changing software that is automatically deployed. On the other hand, the delivery pipeline—a key component in continuous deployment—can act as an element that brings rigor to the development, thus supporting regulatory activities.

Extending regulatory actions of DevOps to include model-related parts of MLOps looks easy on the surface. After all, all that is added in the regulatory process is the ML model and its training and monitoring. However, as Fursin et al. [23] pointed out, finding the relevant code and training models is only the tip of the MLOps iceberg. Therefore, it is not enough to consider models from the regulatory perspective, but instead, considering the whole is necessary. This, in turn, adds the complexity of regulatory actions compared to considering regulative requirements in the DevOps context. The new parts—the models, training data sets, and bias detection—must be added under the regulatory umbrella to create regulatory-compliant medical ML systems.

Regulatory Landscape

The new revision of the EU legislation for medical devices is being implemented with two new regulations, Regulation (EU) 2017/745 on medical devices (MDR) [24] and Regulation (EU) 2017/746 on in vitro diagnostic medical devices (IVDR) [25]. Regulation is a legal instrument in the EU legislation system that will enter into force directly in all Member States, aiming to harmonize the national adoptions' differences. However, it is also possible to supplement EU regulations with national laws.

The regulatory framework can be interpreted to consist of several layers, including the following:

- Union harmonized legislation (i.e., regulations and directives),
- National legislation,
- Harmonized standards,
- Guidelines.

A vital feature of the regulatory framework is to limit legislative harmonization to the essential, product-related requirements [26]. The nature of the essential requirements is that they address the fundamental requirements of public interest, which are generally dealing with health and safety protection. They do not specify precise technical solutions, only required outcomes and the hazards that need to be addressed. It should be noted, however, that MDR and IVDR are relatively detailed documents. In the context of medical devices, safety and state-of-the-art clinical performance are the most critical features. For this reason, they are enforced in the annexes of MDR and IVDR: Annex I is titled as "General safety and performance requirements" and Annex II "Technical documentation."

Medical device manufacturers can use harmonized standards to demonstrate conformity. The use is voluntary, but it is highly recommended as it provides a preassumption of conformity to those essential requirements that the standard aims to cover. Furthermore, international standards are a readymade and effective tool to address the requirements and are generally expected by the regulatory authorities. The suitable set of harmonized standards to be used must be selected based on the identified applicable essential requirements. Currently, there are no harmonized standards under the MDR or IVDR. As a result, there is a unique challenge for manufacturers on how to demonstrate conformity.

The EU Commission provides guidance documents to help manufacturers and stakeholders implement the regulatory requirements. These guidance documents are adopted by the Medical Device Coordination Group (MDCG) that was created by the Commission to ensure a harmonized implementation of MDR and IVDR in the Member States. The MDCG guidance documents are usually developed in active conjunction with the industry stakeholders. While the guidances are not legally binding, they provide advice and support for the manufacturers and, at the same time, contribute to the expectation level of the regulatory authorities.

All of the layers mentioned above provide a general framework for medical device development. They also leave the responsibility to tailor the exact details of the development process for the manufacturers.

Regulatory Considerations for AI/ML

The two new Regulations seek to remedy certain shortcomings of the previous legislation and, as a result, introduce particular new concepts regarding, for example, scientific innovations and emerging manufacturing technologies. Against this background, it is surprising to see that AI/ML is not among them. However, the subject will be addressed in the future by MDCG as there is ongoing guidance development with the title “Artificial Intelligence under MDR/IVDR framework” [27].

As discussed earlier, there are currently no harmonized standards under the new regulatory framework. Nonetheless, the EU commission’s recent standardization request [28] can be used as a source of information to get an insight into the expectations of regulatory authorities related to applicable standards. The use of a standard from the harmonization request list is easily justified. Therefore, the list of applicable standards can be used to define precise regulatory requirements based on the device type. As the standardization request list does not contain specific AI/ML device-related standards that would be generally available, the set of standards commonly used in medical device software development is applicable also in the AI/ML context.

Medical device manufacturers are expected to demonstrate the clinical benefits of their products to a sufficient degree. The device’s technical file must include a description of the technology used and adequate safety and performance evidence. While an ML-based system can be a potent tool to utilize existing healthcare data to create new insights for the patients’ benefit, the ML model can be relatively complex and challenging to understand in full. Therefore, the manufacturers should pay special attention to the level of explicability and, in particular, to transparency of their ML solutions to enable clinical validation. Furthermore, as the clinical evaluation is an ongoing process throughout the lifecycle of a medical device, the post-market surveillance activities must be planned to monitor the device’s clinical performance.

One key feature of a specific ML system is the ability to learn, adapt, and optimize operations in real-time. The ability to improve by learning from data while performing the critical operation may be the most valuable asset of ML technology. However, this raises the question of the ML system’s autonomous operation in relation to the safety and clinical performance in the medical device domain. As a result, the regulatory authorities have traditionally promoted the approach of “locked” algorithms [8, 29], where the system is designed, so that it is being trained during the development phase, and the ability to improve is disabled in real-world use. The study by Feng et al. investigated the problem of “locked” algorithms in the regulated medical space and proposed different policies for regulating modifications to AI/ML-based medical software systems [30]. Although the study was motivated by the FDA’s proposed approach introduced in their discussion paper [8], the regulatory authorities have not yet adopted these proposals.

The limitations listed above can be mapped to the MLOps pipelines as policies. Although the general purpose pipeline has the ability to deliver re-trained models throughout the entire application lifecycle in a continuous fashion, the models are “locked” after the packaging state, while the monitoring phase is limited only for assisting the post-market regulatory activities. As this setup reflects only the current regulatory practice, the policy can be revised when the practice evolves as the “locked” algorithm limitation is not technical by nature. We can envision that, as the practice of using ML technology in medical products becomes more mature, we will be able to change the restrictions encoded by this policy and enable a more dynamic model update behavior in production.

Risk Management for ML in Medical Applications

Algorithms are an essential ingredient of machine learning. The risks inherent to algorithm design propagate to medical machine learning applications due to their increased complexity, lack of transparency, inappropriate use, or weak governance. According to Krishna et al. [31], algorithmic risk can be split into three categories: input data, algorithm design, and output decisions. Flaws in input data can lead to mismatches between the data used for training and the data used during normal use. Output decision flaws relate to incorrect interpretation or use of the output. Algorithm design flaws cover technical flaws, such as the lack of technical rigor or conceptual soundness during development, training, testing, and validation, as well as human biases, such as cognitive biases of model developers and users which can lead to flawed output. Moreover, usage flaws—incorrect implementation or integration in operations can lead to inappropriate decision-making, or security flaws. Inappropriate consideration of these ML-related risks can have a dramatic impact on the effectiveness of medical products, considering their “locked” algorithmic state. For example, slight changes in patient demographics can render a product ineffective when deploying in a different geographic area, as the training data is different than the usage data. Similarly, a product may be considered ineffective, if similar patients are not treated similarly, due to the algorithm instability. The *similar inputs/similar outputs over same inputs/same outputs* approach [32] can be encoded into tests and enforced in an automatic fashion by MLOps pipelines during development.

As new machine learning techniques are so opaque, it is hard to understand how they operate. Industry groups [33] and academia recognized the challenges associated with machine learning trustworthiness. Their proposals to improve trustworthiness include metrics like interpretability (degree to which a human can understand the cause of a decision [34]) and explainability (the degree to which a human understands the behavior of a system [35]), or introduce frameworks that allow to explain machine learning models as black [36, 37] or white boxes [38]. The intended

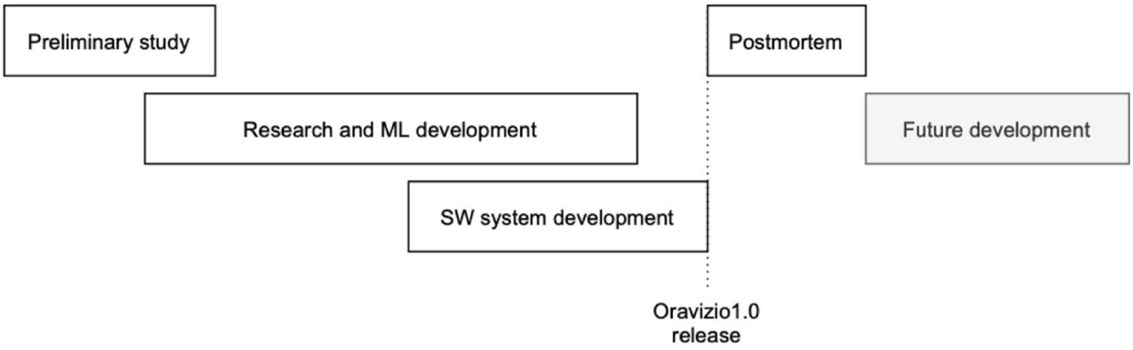

Fig. 3 Oravizio 1.0 development timeline

purpose, the associated Quality Management System, and the automated pipelines following DevOps practices provide a solid foundation for handling in a consistent way the design control and risk management activities related to machine learning. Practical experience is needed to evaluate how medical manufacturers can digest these concepts.

Case Study: Oravizio

Oravizio is a medical device software product that employs machine learning technologies to provide decision support for the orthopedic specialist for patient-specific joint replacement surgery risk evaluation. Started as a preliminary study, the development continued with a research phase for the development of the ML models, and a productization phase that resulted in the release of the first product version. The imminent MDR coming into force triggered an analysis of the development process. The aim was to identify potential opportunities to improve in the current practice to be utilized in the development of next product iterations, see Fig. 3.

In this section, we describe first the methodology we have used in this case study. Then, we provide a brief description of the application domain. We continue with a detailed description of the development processes used during the research and software development phases, and conclude with the regulatory gap analysis.

Methodology Considerations: Postmortem Investigation

One of the DevOps culture's main drivers is that "you should run what you build" [39]. With this model, a development team with access to all aspects of the product development lifecycle has more autonomy. This means making informed local decisions, working with increased transparency, having visibility to tools and processes, and ultimately designing the software with production in mind, since every increment is deployable. However, there are often hard administrative and organizational barriers in regulated environments that make it difficult to always run what is being built. Therefore, the faults along the organizational boundaries separate the teams on either side as they develop, test, and validate different parts of the software system, often using non-harmonized processes.

Our investigation focuses on identifying the rifts in the development process lifecycle from the regulatory perspective. For this purpose, we use *post-mortem*, a DevOps discipline used typically during incident response, to investigate the root cause of a defect in a software system and identify the corrective actions and draw the lessons learned, so that such incidents do not happen again. The investigation draws some similarities with the *agile retrospective*, as we seek to identify areas of improvements. However, it is not performed by the development team, and its focus is on longer term improvements and on establishing best practices.

The post-mortem investigation was performed after the team delivered the first iteration of the software system, e.g., Oravizio 1.0 release. The investigation aims to document how the software system was developed currently by the different teams involved and identify the regulatory gaps in the software development lifecycle that would be blockers when transitioning to the new MDR. The investigation findings are to be used for developing new iterations of Oravizio, and serve as input when planning new regulated products.

Oravizio: An overview

Oravizio[5] is a medical device software product that provides data-driven information about patient-level risks related to hip (total hip arthroplasty, THA) and knee joint (total knee arthroplasty, TKA) replacement surgery. Even though THA and TKA are highly effective surgical procedures with a high success rate, they involve a certain risk of adverse events.

[5] https://oraviz.io/

The most common adverse events after THA and TKA surgery include re-operations, prosthetic joint infection, and other serious medical complications that can lead to death. However, a successful operation can dramatically improve the patient's quality of life and reduce society's costs.

In its current form, Oravizio provides three different dedicated prediction models:

- Risk of infection within 1 year from surgery.
- Risk of revision within 2 years from surgery.
- Risk of death within 2 years from surgery.

Oravizio aims to help the surgeon collaborate and negotiate with a patient to make informed decisions when assessing the patient's eligibility for surgery. Such a shared decision-making process is recognized to be a vital part of modern medicine. Oravizio provides both the patient-specific risks and the surgery's expected outcomes in a fashion that is understandable to the patient. As people are generally not good at understanding risks and probabilities, it is essential to present the surgery's risks and expected outcomes understandably to help the layman understand their practical meaning. Furthermore, the amount of patient history data behind Oravizio's risk calculation model is so vast that it would be impossible for a surgeon to process it manually during a patient appointment. The risk calculation algorithms combine manually submitted patient-specific data with comprehensive patient history data.

Oravizio 1.0 Development

Oravizio 1.0 was developed in close collaboration with a hospital that is specialized in joint replacement surgery and Solita, a European software company headquartered in Finland. The clinical partner hospital had accumulated a large volume of data from surgeries for more than 10 years. The development of Oravizio was based on the vision that this data asset might include factors that indicate risks for joint replacement surgeries down to an individual patient's level. The preliminary study was carried out to validate the preconceptions.

Data consolidation and pre-processing During the development of Oravizio 1.0, the data included over 30,000 patient records. However, the data were not well organized—it was stored in various formats and computer systems, some of which had already been retired. As a result, consolidation and pre-processing were a considerable task. The first task was to consolidate the relevant data into a data lake. Second, the data needed to be pre-processed to determine its quality and ensure its uniformity and future utility.

Selection of variables The prepared data included over 750 pre-operation variables from 2008 onwards. The variables included, for example, general information (age, gender, BMI), medication, laboratory values, diagnoses, patient-reported information, and derivate variables. The clinical partner's clinical know-how was used during the research, combined with research literature and computational methods, to decide the variables to be further analyzed by data scientists. The well-documented linking between particular variables and risks in joint replacement surgery was used as a baseline for the selection.

The goal was to select a relatively limited set of relevant variables with a significant impact on the calculation. The modest set of input variables ensures that the finished product is practical to use, and the impact of individual risk factors can be illustrated in an easily understandable way. In the analysis, computational methods LASSO, Ridge regression, and Elastic net were used to select the most suitable parameters for the models.

Building and testing the machine learning algorithms Several mathematical methods were considered during the research. The aim was to create an explanatory machine learning model for each risk to enable validation and ensure regulatory compliance. It should be noted that the relations between the risk in joint replacement surgery and selected explanatory values were mostly known from clinical literature. The following methods were tested for building the risk models:

- Logistic regression.
- Decision tree-based methods (Random forest).
- Gradient boosting methods (XGBoost [40]).
- Weibull/Cox survival mode.

During the development, data from years 2008–2015 were used for training, and data from years 2016 to 2018 were used for testing. The performance was estimated with AUC values and ROC curves [41]. Based on the results, gradient boosting with XGBoost produced the best performance. As a result, XGBoost was selected, and the final model for the product was built accordingly. The model is deterministic by nature. Therefore, it can be validated with test data in a test environment without the need to do the validation in the final production environment. To address the earlier discussed medical device regulatory constraint related to self-adaptation, Oravizio was designed to be deployed in a production environment with its ML model in a "locked" state. In practice, Oravizio's models are trained during the development phase, and their ability to improve the outcome on the fly is disabled in production use for regulatory reasons.

Software development process Solita's process for designing and developing medical device software is well established and supported by an ISO 13,485 certified quality management system. The process is an implementation of an agile development process, utilizing ideas, tools, and best

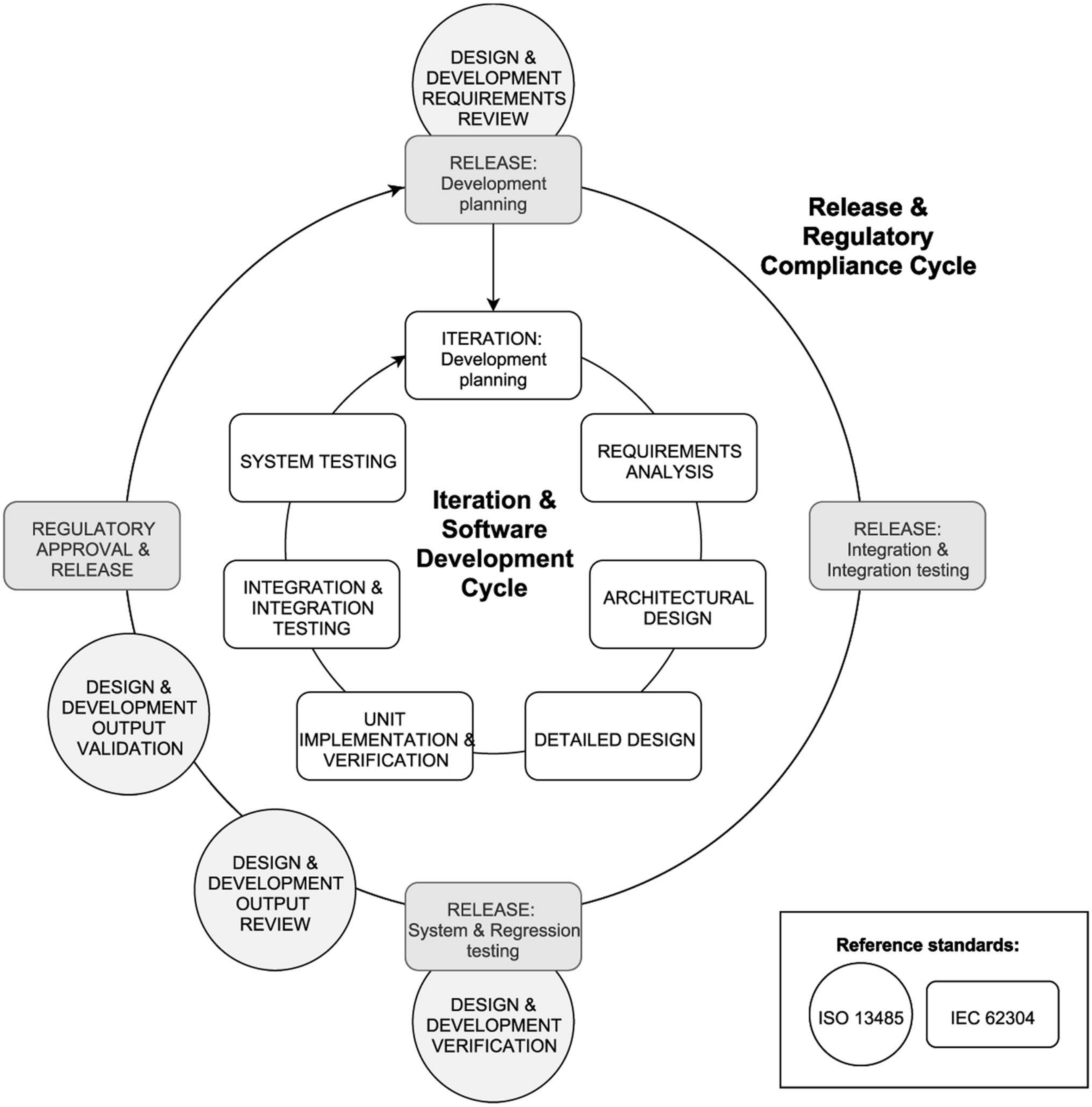
DESIGN & DEVELOPMENT REQUIREMENTS REVIEW
RELEASE: Development planning
Release & Regulatory Compliance Cycle
ITERATION: Development planning
SYSTEM TESTING
REQUIREMENTS ANALYSIS
Iteration & Software Development Cycle
REGULATORY APPROVAL & RELEASE
RELEASE: Integration & Integration testing
INTEGRATION & INTEGRATION TESTING
ARCHITECTURAL DESIGN
DESIGN & DEVELOPMENT OUTPUT VALIDATION
UNIT IMPLEMENTATION & VERIFICATION
DETAILED DESIGN
DESIGN & DEVELOPMENT OUTPUT REVIEW
RELEASE: System & Regression testing
DESIGN & DEVELOPMENT VERIFICATION
Reference standards:
ISO 13485
IEC 62304

Fig. 4 Solita regulatory-compliant development process. For simplicity, risk management and usability engineering concepts are left out from the figure

practices from several different agile methods. The process complies with the applicable regulation and international standards, namely IEC 62304, ISO 14971, and IEC 62366-1. The process is presented in Fig. 4.

In the process, the development tasks are divided into two nested cycles: the inner cycle for daily development tasks carried out in short iterations and the outer cycle for a formal execution of regulatory tasks needed for the final approval of the software release. With this approach, different level tasks can be assigned to persons according to their role and competence requirements, and they can be completed asynchronously.

Design and development requirements, i.e., design inputs, are reviewed at the release level. Once accepted, design inputs are transferred to the development cycle for further analysis and architectural design. Architecture design is verified against the requirements and the detailed unit design. After the implementation is done based on the accepted design, it is verified with the review and automated integration and system testing.

When all requirements have been implemented within the needed number of development iterations, the work can be transferred to the release cycle. In the release cycle, the software release is authorized through formal final testing, verification, review, and validation stages. Simultaneously, as the release tasks are performed, development work can continue in the development cycle with the requirements of the next release.

As a result of the regulatory requirements, the medical product, consisting of the ML models and application code, can only be accepted for deployment by authorized persons within the controlled design and development output review and validation processes. A release decision implies completeness of risk management activities, and therefore, risk managers and compliance officers are typically involved.

Development Process Gap Analysis

While Solita's medical device software development process can be considered state of the art, the same assessment does not apply to the ML development process used during Oravizio 1.0 development. ML development was based on a manual process: the steps of data consolidation, data preprocessing, datalake creation, variable selection, ML method selection, model training, and validation were all done manually, at least to some degree. There were no automatic CI or CD pipelines. Some process steps that were executed repeatedly were automated with shell scripts; however, the scripts were executed in local development machines.

The main reason for the level of manual work in the ML development process is that the ML development was done within a scientific research project, and the decision to proceed to commercial software product development was made only during the project. Furthermore, the research project was performed on the premises of the clinical partner hospital. As the use of patient records is strictly regulated, the hospital's computational environment is tightly restricted and isolated. In practice, only a few selected and accredited data scientists had access to a restricted environment. In contrast, the rest of the development team, including software developers and risk managers, worked in Solita's environment. The fact that the product development work was divided into two different organizations' environments can be seen as a challenge for utilizing routine DevOps and MLOps practices.

To mitigate the challenge, all development artifacts, including the scripts that were used to create ML models, were stored in a mutually shared version control system (VCS). However, as the trained model exists as a binary file, the model handover from data scientists to software developers was done through a network drive. Even though the process included an element of disconnection between data scientists and software developers to a certain extent, the development team did not consider this as a problem due to the small team size and close cooperation.

By design, the production use of Oravizio does not generate data that could be used for re-training. Instead, the needed data are generated in other clinical processes of a hospital. Due to the ML development environment's restrictive nature, frequent changes or re-trainings to the ML models were not anticipated.

However, the second iteration re-training with 45,000 patient records improved models' performance to a certain extent. As a result, the benefits of model re-training became evident.

Although active quality monitoring of the "locked" model performance in production might not be essential in all sets of conditions, it is a vital activity in the medical device domain due to the regulatory framework's post-market surveillance requirements. Therefore, this capability was already implemented in the system in the first iteration to monitor systems' clinical performance specifically.

Discussion

Designing Oravizio has given us much insight into the development of medical software where ML is involved. Below, we summarize the key lessons learned in the process, followed by a discussion regarding the limitations of this work.

Lessons Learned

From experimentation to production Numerous AI projects start with a feasibility study and data analysis to understand if it is possible to solve the problems with ML [42]. It is vital to be able to be agile during the data science and ML research phase—after all, there is always a certain level of uncertainty involved in the research, and the end result is usually achieved through trial and error. However, it is also reasonable to be prepared for the situation where the experiment succeeds. In practice, it is recommended that there is a clear understanding of arranging and automating the re-learning, re-building, revalidation, and re-deployment of the model. Also, experimentation with new and improved methods should be considered. Ideally, these aspects are implemented already in the research phase using practices and tools familiar from MLOps. Immediate benefits are possible as automation potentially reduces the amount of manual work instantly.

Regulatory compliance In comparison to ML projects from non-regulated settings, there were some particular features and challenges in Oravizio development. When

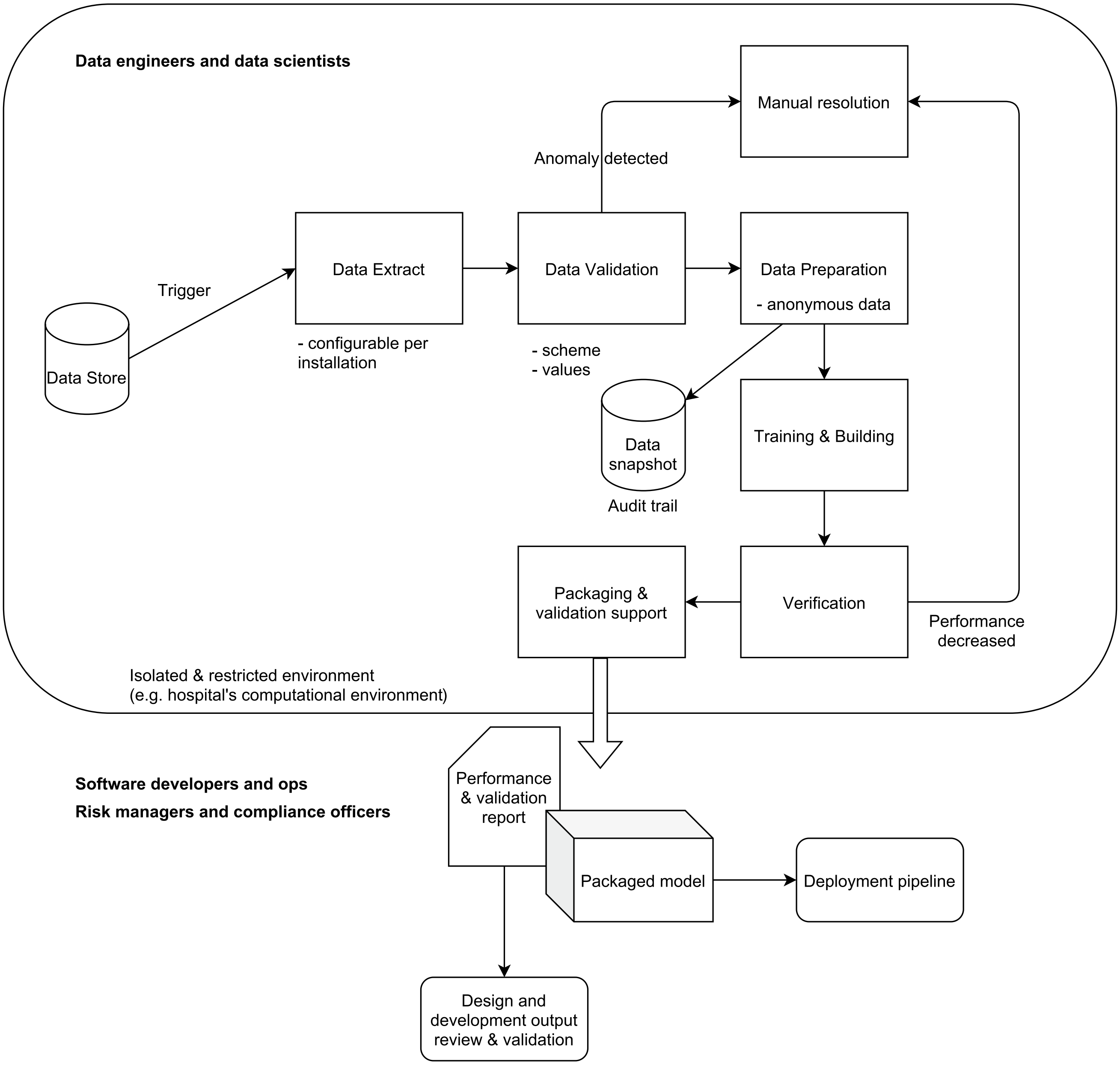
Data engineers and data scientists
Manual resolution
Anomaly detected
Data Store
Trigger
Data Extract
- configurable per installation
Data Validation
- scheme
- values
Data Preparation
- anonymous data
Data snapshot
Audit trail
Training & Building
Packaging & validation support
Verification
Performance decreased
Isolated & restricted environment
(e.g. hospital's computational environment)
Software developers and ops
Risk managers and compliance officers
Performance & validation report
Packaged model
Deployment pipeline
Design and development output review & validation

Fig. 5 Continuous training pipeline, with arrows indicating data flows

developing a medical device product, it is essential to clearly understand applicable regulatory requirements and determine a regulatory strategy accordingly from the beginning of the project. In practice, the regulatory strategy was implemented by choosing an explainable and deterministic ML model that is deployed to production in a "locked" state. This strategy enables easily verifiable regulatory compliance and validation of the end product. The unique challenge in the Oravizio project was the severely restricted access to the data used in model development.

Continuous training Based on our practical example, MLOps automation goals are achievable, and the addressed regulatory challenges solvable by implementing the *continuous training* pipeline presented in Fig. 5. The pipeline's design builds on the general solution illustrated in Fig. 2 and addresses specific regulatory constraints identified in the case study. As discussed previously, the data that can be used to re-train the model are not collected in Oravizio's computational environment—only an anonymized audit trail is generated. However, it is possible to design the continuous training pipeline to operate inside the clinical partner's controlled environment and fetch new data from the data lake based on pre-defined triggers. The automated continuous training pipeline design includes specific new steps that were not present in the first development iteration.

The data validation step ensures that the input data are as expected by checking data validity against the pre-defined data schema and value limits. In the unlikely event that data anomalies are found during the data validation, the pipeline's execution is terminated for manual resolution. It is worth noticing that the manual resolution is the only process step where data engineers' and data scientists' involvement is needed once the continuous training pipeline operates automatically.

After re-training and building, the new model is verified with a subset of the data reserved only for testing. The performance metrics are compared to the previously accepted model's values, and the validation report is generated automatically with relevant metrics and performance graphs. This practice of continuous re-verification is well aligned with the approach of continuous risk-monitoring, as suggested by Babic et al. [32].

Once the model is packaged, and the validation report generated, these artifacts can be automatically delivered from the isolated and restricted environment to the development team in another organization. The development team integrates the model into the software system.

With the continuous training pipeline approach presented above, the development team does not need access to the restricted environment beyond the continuous training pipeline's installation and maintenance. The validation report, generated by the pipeline, can be used in validating the re-trained model to address the regulatory requirements related to clinical performance validation. Furthermore, the automated pipeline enables fairly effortless initial ML model development in new clinical setups and different health care environments.

Limitations

Oravizio was designed and implemented before the concept of MLOps had emerged. However, its design goals were mainly similar to MLOps characteristics in a regulatory context, simply because this seemed the most straightforward way to realize the system. Hence, considering Oravizio in the light of regulated MLOps can be considered justifiable, although it is admitted that there are also some apparent mismatches. These mismatches, however, are not related to technical implementation but the organizational setup and data ownership. Improved understanding of such practicalities forms a direction for future work.

There are some validity-related aspects that are associated with the case study. A clear threat to the validity of any single case study is that it is questionable to what extent empirical evidence can be relied on or generalized. In this particular case, the data are from an ongoing project that has been running in production for several years now. The involved organizations have invested much attention in ensuring the regulatory compliance of the implementation and related processes. The contents have been checked by the team working on Oravizio for accuracy to mitigate risks related to misconceptions with respect to the process and implementation. Finally, it can be questioned if the case study setup is typical, as the operation is run in collaboration with a hospital and a private company. However, since the body of knowledge in the literature on how to handle possible biases is limited, we have little means to mitigate these threats.

Conclusions

Continuous software engineering has become commonplace in numerous contexts. The emerging practice of MLOps takes this one step further, also enabling continuous delivery of ML features. However, such continuous practices are not immediately compatible with regulatory requirements that may need authority involvement.

In this paper, we have presented a case study on deploying ML components to production in the medical context. Based on the case study, we pinpointed what parts of MLOps were not immediately compatible with the domain. In addition, we identified additional challenges that do not originate from technology but the organizational setup, which also affects on how MLOps can be implemented. In particular, this is related to the availability of training and testing data. In the future, we are studying opportunities to streamline further the deployment pipeline in the case study, based on the results of the paper.

Acknowledgements The authors would like to thank Business Finland and the members of the AHMED (Agile and Holistic MEdical software Development) consortium for their contribution in preparing this paper.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Beck K. Embracing change with extreme programming. Computer. 1999;32(10):70–7.
2. Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt, A Jeffries R, et al. Manifesto for agile software development. 2001. https://agilemanifesto.org/. Accessed 21 Dec 21 2020.
3. Mc Hugh M, Cawley O, McCaffcry F, Richardson I, Wang Z. An agile v-model for medical device software development to overcome the challenges with plan-driven software development lifecycles. In: 2013 5th International Workshop on software engineering in health care (SEHC), pp. 12–19. IEEE, 2013.
4. Boehm B, Turner R. Using risk to balance agile and plan-driven methods. Computer. 2003;36(6):57–66.
5. Fitzgerald B, Stol K-J. Continuous software engineering: a roadmap and agenda. J Syst Softw. 2017;123:176–89.
6. Braiek HB, Khomh F. On testing machine learning programs. J Syst Softw. 2020;164:110542.
7. Roberts M, Driggs D, Thorpe M, Gilbey J, Yeung M, Ursprung S, Aviles-Rivero AI, Etmann C, McCague C, Beer L, et al. Common pitfalls and recommendations for using machine learning to detect and prognosticate for covid-19 using chest radiographs and ct scans. Nat Mach Intell. 2021;3(3):199–217.
8. FDA. Proposed regulatory framework for modifications to artificial intelligence/machine learning (ai/ml)-based software as a medical device (samd) - discussion paper and request for feedback., 2019. https://www.fda.gov/media/122535/download. Accessed 14 Mar 2021.
9. Debois P. Devops: a software revolution in the making. Cutter IT J. 2011;24(8):3–9.
10. Rajkumar M, Pole AK, Adige VS, Mahanta P. Devops culture and its impact on cloud delivery and software development. In: 2016 International Conference on Advances in computing, communication, & automation (ICACCA)(Spring), pages 1–6. IEEE, 2016.
11. Bass L, Weber I, Zhu L. DevOps: A software architect's perspective. Addison-Wesley Professional; 2015.
12. Fitzgerald B, Stol K-J. Continuous software engineering and beyond: trends and challenges. In: Proceedings of the 1st International Workshop on rapid continuous software engineering, pp. 1–9, 2014.
13. Lwakatare LE, Kuvaja P, Oivo M. Dimensions of devops. In: International Conference on agile software development, pp. 212–217. Springer, 2015.
14. Mikkonen T, Nurminen JK, Raatikainen M, Fronza I, Mäkitalo N, Männistö T. Is machine learning software just software: a maintainability view. In: Software quality days. Springer, 2021.
15. Sato D, Wilder A, Windheuser C. Continuous delivery for machine learning, Sept 2019. https://martinfowler.com/articles/cd4ml.html. Accessed 21 Dec 2020.
16. 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: Proceedings of the 28th International Conference on neural information processing systems—Volume 2, NIPS'15, page 2503–2511, Cambridge, MA, USA, 2015. MIT Press.
17. AWS Solutions. AWS MLOps Framework. https://docs.aws.amazon.com/solutions/latest/aws-mlops-framework/welcome.html. Accessed 14 Mar 2021.
18. Google Cloud Solutions. Mlops: Continuous delivery and automation pipelines in machine learning. https://cloud.google.com/solutions/machine-learning/mlopscontinuous-delivery-and-automation-pipelines-in-machine-learning. Accessed 14 Mar 2021.
19. Baylor D, Breck E, Cheng H-T, Fiedel N, Foo CY, Haque Z, Haykal S, Ispir M, Jain V, Koc L, Koo CY, Lew L, Mewald C, Modi AN, Polyzotis N, Ramesh S, Roy S, Whang SE, Wicke M, Wilkiewicz J, Zhang X, Zinkevich M. Tfx: a tensorflow-based production-scale machine learning platform. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, pp. 1387–1395, New York, NY, USA, 2017. Association for Computing Machinery.
20. Laukkarinen T, Kuusinen K, Mikkonen T. Devops in regulated software development: case medical devices. In: 2017 IEEE/ACM 39th International Conference on software engineering: new ideas and emerging technologies results track (ICSE-NIER), pp. 15–18. IEEE, 2017.
21. Laukkarinen T, Kuusinen K, Mikkonen T. Regulated software meets devops. Inf Softw Technol. 2018;97:176–8.
22. Lie MF, Sánchez-Gordón M, Colomo-Palacios R. Devops in an iso 13485 regulated environment: a multivocal literature review. In: Proceedings of the 14th ACM/IEEE International Symposium on empirical software engineering and measurement (ESEM), pp. 1–11, 2020.
23. Fursi G, Guillou H, Essaya Nn. Codereef: an open platform for portable mlops, reusable automation actions and reproducible benchmarking. arXiv preprint arXiv:2001.07935, 2020, 2020.
24. European Parliament and the Council. Regulation (EU) 2017/745 on medical devices, 2017. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02017R074520200424#tocId168. Accessed 21 Dec 2020.
25. European Parliament and the Council. Regulation (EU) 2017/746 on in vitro diagnostic medical devices, 2017. https://eur-lex.europa.eu/legalcontent/EN/TXT/?uri=CELEX:02017R0746-20170505#tocId157. Accessed 21 Dec 2020.
26. European Commission. Commission notice – the 'blue guide' on the implementation of eu products rules 2016. In: Official Journal of the European Union, volume 59, 2016.
27. Medical Device Coordination Group (MDCG). Ongoing guidance development within MDCG Subgroups – October 2020. https://ec.europa.eu/health/sites/health/files/md sector/docs/mdcongoing guidancedocs en.pdf. Accessed 21 Dec 2020.
28. European Parliament and the Council. M/575 commission implementing decision of 14.4.2021 on a standardisation request to the european committee for standardization and the european committee for electrotechnical standardization as regards medical devices in support of regulation (eu) 2017/745 of the european parliament and of the council and in vitro diagnostic medical devices in support of regulation (eu) 2017/746 of the european parliament and of the council, 2021. Retrieved: May 2021.
29. Granlund T, Mikkonen T, Stirbu V. On medical device software ce compliance and conformity assessment. In: 2020 IEEE International Conference on software architecture companion (ICSA-C), pp. 185–191, 2020.
30. Feng J, Emerson S, Simon N. Approval policies for modifications to machine learning-based software as a medical device: a study of bio-creep. Biometrics. 2020;77(1):31–44.
31. Krishna D, Albison N, Chu Y. Managing algorithmic risks—Safeguarding the use of complex algorithms and machine learning. In: Deloitte, 2017.
32. Babic B, Gerke S, Evgeniou T, Cohen IG. Algorithms on regulatory lockdown in medicine. Science. 2019;366(6470):1202–4.
33. High-Level Expert Group on Artificial Intelligence European Commission. Ethics guidelines for trustworthy ai, 2019.
34. Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning, 2017.
35. Bhatt U, Xiang, A Sharma S, Weller A, Taly A, Jia Y, Ghosh J, Puri R, Moura JMF, Eckersley P. Explainable machine learning in deployment. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, FAT* '20, page 648–657, New York, NY, USA, 2020. Association for Computing Machinery.

36. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D. A survey of methods for explaining black box models. ACM Comput Surv. 2018;51(5):1–42.
37. Molnar C. Interpretable Machine Learning. 2019. https://christophm.github.io/interpretable-ml-book/. Accessed 21 Dec 2020..
38. Freitas AA. Comprehensible classification models: a position paper. SIGKDD Explor Newsl. 2014;15(1):1–10.
39. Orban S. Enterprise devops: Why you should run what you build, Aug 2015. https://aws.amazon.com/blogs/enterprise-strategy/enterprise-devops-why-youshould-run-what-you-build/. Accessed 21 Dec 2020.
40. XGBoost project. Xgboost extreme gradient boosting. https://github.com/dmlc/xgboost. Accessed 21 Dec 2020.
41. Huang J, Ling CX. Using auc and accuracy in evaluating learning algorithms. IEEE Trans Knowl Data Eng. 2005;17(3):299–310.
42. Aho T, Sievi-Korte O, Kilamo T, Yaman S, Mikkonen T. Demystifying data science projects: A look on the people and process of data science today. In: International Conference on product-focused software process improvement, pp. 153–167. Springer, 2020.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.