

VTT Technical Research Centre of Finland

## Agile and Holistic Medical Software Development

Lähteenmäki, Jaakko; Ahola, Pasi; Baraian, Andrei; Förger, Klaus; Granlund, Tuomas; Hopia, Jani; Kaikkonen, Risto; Mikkonen, Tommi ; Niemirepo, Timo; Pajula, Juha; Partanen, Jari; Pellinen, Timo; Stirbu, Vlad; Torhola, Mika

Published: 13/02/2023

*Document Version*  
Publisher's final version

*License*  
Unspecified

[Link to publication](#)

*Please cite the original version:*

Lähteenmäki, J. (Ed.), Ahola, P., Baraian, A., Förger, K., Granlund, T., Hopia, J., Kaikkonen, R., Mikkonen, T., Niemirepo, T., Pajula, J., Partanen, J., Pellinen, T., Stirbu, V., & Torhola, M. (2023). *Agile and Holistic Medical Software Development: Final report of AHMED project.*



VTT  
<http://www.vtt.fi>  
P.O. box 1000FI-02044 VTT  
Finland

By using VTT's Research Information Portal you are bound by the following Terms & Conditions.

I have read and I understand the following statement:

This document is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of this document is not permitted, except duplication for research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered for sale.

# Agile and Holistic Medical Software Development

Final report of AHMED project

Authors: Jaakko Lähteenmäki (editor), Pasi Ahola, Andrei Baraian, Klaus Förger, Tuomas Granlund, Jani Hopia, Risto Kaikkonen, Tommi Mikkonen, Timo Niemirepo, Juha Pajula, Jari Partanen, Timo Pellinen, Vlad Stirbu, Mika Torhola

Confidentiality: Public

Date: 13.02.2023



|  |  |
|--|--|
| <b>Report's title</b><br>Agile and Holistic Medical Software Development – Final report of AHMED project   |  |
| <b>Customer, contact person, address</b><br>Pekka Ollikainen, Business Finland   | <b>Order reference</b><br>27048/31/2020 (BF)             |
| <b>Project name</b><br>Agile and Holistic Medical Software Development (AHMED)   | <b>Project number/Short name</b><br>126 VTT-R-01079-22   |
| <b>Author(s)</b><br>Jaakko Lähteenmäki (editor), Andrei Baraian, Timo Niemirepo, Juha Pajula (VTT)<br>Pasi Ahola (Taipuva Consulting)<br>Klaus Förger, Mika Torhola (Atostek)<br>Tuomas Granlund, Risto Kaikkonen (Solita),<br>Jani Hopia (Terveystalo)<br>Jari Partanen (Bittium)<br>Timo Pellinen (Mylab)<br>Vlad Stirbu (University of Helsinki)<br>Tommi Mikkonen (University of Helsinki / University of Jyväskylä)   | <b>Pages</b><br>53/                                      |
| <b>Keywords</b><br>MDR, IVDR, artificial intelligence, machine learning, MLOps, RegOps   | <b>Report identification code</b><br>VTT-R-01079-22      |
| <b>Summary</b><br>AHMED is a co-innovation project ( <a href="https://www.regops.fi/">https://www.regops.fi/</a> ) with six companies (Atostek, Bittium, Mylab, Solita, Taipuva Consulting and Terveystalo) and two research organisations (University of Helsinki and VTT). This report provides summarized results of the AHMED project focused on research, development and adoption of solutions for efficient production of data-driven applications for healthcare. The project particularly addressed challenges related to the usage of artificial intelligence, machine learning and personal data in regulated medical software. We present experiences and best practices for transition to the new medical device regulation (MDR), AI/ML compliant RegOps models, tools and toolchains for medical software development, healthcare register data usage for innovative applications and traceability approaches for research and AI/ML model development. |  |
| <b>Confidentiality</b>   | Public   |
| Espoo 13.02.2023<br><b>Written by</b><br>Jaakko Lähteenmäki<br>Principal Scientist   | <b>Reviewed by</b><br>Mika Hilvo<br>Research Team Leader |
| <b>VTT's contact address</b><br>Jaakko Lähteenmäki, Teknologian tutkimuskeskus VTT Oy, jaakko.lahteenmaki@vtt.fi   |  |
| <b>Distribution (customer and VTT)</b><br>Business Finland, AHMED project partners and external steering group members.<br>Open distribution to all interest parties.  |  |
| <i>The use of the name of "VTT" in advertising or publishing of a part of this report is only permissible with written authorisation from VTT Technical Research Centre of Finland Ltd.</i>  |  |



## Preface

---

The AHMED co-innovation project responds to several major trends affecting the development of software based innovations for healthcare. Artificial intelligence and machine learning are increasingly used in healthcare applications and information systems. Sensitive personal data is in many cases needed already in the software development phase. At the same time, there is a need for improving productivity by employing agile software development methods. These trends are challenging from the regulated medical software perspective, which assumes waterfall type development processes and rigorous acceptance procedures for software changes. For many companies these challenges are acute as the deadlines concerning the transition to the new medical device regulation are rapidly approaching.

The AHMED project started in July 2020 with six industrial partners (Atostek, Bittium, Mylab, Solita, Taipuva Consulting and Terveystalo) and two research partners (University of Helsinki and VTT). This report provides a summary of project results achieved by the end of 2022 when most partners completed their part of the project. Atostek, Taipuva, University of Helsinki and VTT continue their project activities still during 2023.

We are grateful for the support of Business Finland which was essential for the realization of the co-operative project and for the dissemination of the results to a wider audience. We also extend our warm thanks to all AHMED steering group members for their excellent co-operation and guidance throughout the project.

Espoo 13.02.2023,

Authors

## Contents

---

|   |    |
|---|----|
| Preface.....  | 3  |
| Contents.....   | 4  |
| 1. Introduction.....  | 6  |
| 2. Experiences on MDR transitions.....  | 8  |
| 2.1 MDR transition periods .....  | 8  |
| 2.2 Bittium experience on MDR certification process.....                                | 8  |
| 2.3 Early stage reflections of AI/ML themes.....  | 9  |
| 2.4 MDR from Terveystalo perspective.....   | 9  |
| 2.5 IVDR from Mylab perspective .....   | 10 |
| 3. RegOps Lifecycle for AI/ML compliance .....  | 12 |
| 3.1 Overview .....  | 12 |
| 3.2 Objectives.....   | 12 |
| 3.3 Key outcomes.....   | 13 |
| 3.3.1 Regulatory landscape .....  | 13 |
| 3.3.2 Continuous training of AI/ML in a regulated environment .....                     | 14 |
| 3.3.3 RegOps Lifecycle for AI/ML.....   | 16 |
| 3.4 Business opportunities.....   | 17 |
| 4. Methodologies and tools for developing medical device software .....                 | 18 |
| 4.1 Overview of activities.....   | 18 |
| 4.2 Medical software development processes and regulatory Background .....              | 18 |
| 4.3 True RegOps cycle.....  | 19 |
| 4.4 Tools .....   | 20 |
| 4.4.1 GitHub and GitLab – software development, version control and task management ... | 21 |
| 4.4.2 Polarion – for planning, management and documentation.....                        | 21 |
| 4.4.3 JIRA, Confluence – agile project management and documentation.....                | 22 |
| 4.5 Out-of-the-box process for software development .....                               | 22 |
| 4.6 Traceability.....   | 23 |
| 4.7 Reusable process for tool chain validation.....                                     | 25 |
| 4.8 Challenges in building multitool RegOps tool chains.....                            | 26 |
| 4.9 Considerations on using ML technology in medical devices .....                      | 28 |
| 4.9.1 Model cards as ML development ledger.....   | 28 |
| 4.9.2 Multi-organization ML development.....  | 29 |
| 5. Data exploitation challenges.....  | 30 |
| 5.1 Overview .....  | 30 |
| 5.2 Goals.....  | 30 |
| 5.3 Methods .....   | 30 |
| 5.4 Challenges .....  | 30 |
| 5.5 Outcome.....  | 31 |
| 6. Research and ML model development for medical device software.....                   | 32 |

- 6.1 Overview .....32
- 6.2 Machine learning model development using sensitive personal data .....32
  - 6.2.1 General .....32
  - 6.2.2 Use case: health and social services decision support .....33
  - 6.2.3 Secure processing environment .....33
  - 6.2.4 ETL and analysis scripts .....34
  - 6.2.5 Ensuring traceability .....35
  - 6.2.6 Data version control tools .....38
- 6.3 DevOps tools supporting ML based health monitoring solution .....39
  - 6.3.1 General .....39
  - 6.3.2 Tools .....39
  - 6.3.3 Reference system .....40
  - 6.3.4 Integrated system .....41
  - 6.3.5 Requirements and issues management .....42
  - 6.3.6 Continuous integration and deployment .....43
- 6.4 Development of ML based machine vision for medical purposes .....43
  - 6.4.1 General .....43
  - 6.4.2 Image enhancement using AI for endoscopic imaging .....43
  - 6.4.3 DVC and Gitlab in supporting image data control .....44
- 7. Conclusions .....47
- References .....49
- Annex 1: Terms and Acronyms .....52

## 1. Introduction

---

Agile software development has become the dominant approach in modern software production. The agile development model emphasizes incremental delivery, team collaboration and continuous planning, instead of waterfall type planning and static documentation. Productivity is enhanced further by applying DevOps practices, targeted to improve collaboration between development and IT operations teams [1].

Agile software and DevOps approaches aim at faster software development cycles and continuous deployment. This contradicts the regulatory requirements in safety critical application areas, such as healthcare, where regulatory compliance of a product is based on rigorous risk management and release acceptance processes. Typically, a new regulatory acceptance is needed whenever the product has been changed.

The legal and regulatory requirements of medical devices are set out in the Medical Device Regulation 745/2017 (MDR) and in Vitro Diagnostic Device Regulation 746/2017 (IVDR) [2], [3]. Medical device software is software that is intended to be used, alone or in combination, for a purpose as specified in the definition of a “medical device” in the MDR or IVDR. Food and Drug Administration (FDA) is responsible for the corresponding regulation in the United States, where the term *Software as a Medical Device* (SaMD) is used for regulated medical software [4].

There is a growing interest towards the use artificial intelligence and machine learning (AI/ML) to improve healthcare [5]. These technologies are extensively used for automatic medical image analysis for supporting and improving human interpretation. AI/ML is increasingly also used to support precision medicine by predicting patient outcomes, identifying patients with elevated risk and suggesting most favourable care pathways and services for patients. AI/ML models empower decision support applications providing guidance to healthcare professionals and patients [6]–[8].

The increasing use of AI/ML in healthcare is especially challenging from regulatory perspective as such applications would need to be frequently updated when new data becomes available<sup>1</sup> [9]–[11]. The specific practices for trustworthy and efficient deployment and maintenance of AI/ML based solutions in production are referred as MLOps [12].

Traceability of software product versions is an important regulatory requirement, but difficult to achieve in AI/ML based application development. This is especially the case when personal data is needed for AI/ML model development due to the limitations for data access implied by the General Data Protection Regulation (GDPR)<sup>2</sup> and the specific legislation on secondary use of data<sup>3</sup> [13]. Additional requirements for AI/ML based applications are set out in the Artificial Intelligence Act, a new regulatory proposal by EU [14]. After approved, the Artificial Intelligence Act will considerably affect the development, marketing and use of AI/ML based applications.

In addition to the international regulation, also national regulation may apply to the use of medical software in the national healthcare information system infrastructure<sup>4</sup>. For example, all information systems and applications to be connected with the Finnish national Kanta architecture, need to pass a specific process to demonstrate compliance with the related essential requirements [15].

For efficient production of high-quality medical software it is essential to fully understand the regulatory requirements and to integrate them closely with the software development pipeline. Such integration, also

---

<sup>1</sup> <https://futurium.ec.europa.eu/en/european-ai-alliance/document/artificial-intelligence-medical-device-legislation>

<sup>2</sup> [EUR-Lex - 32016R0679 - EN - EUR-Lex \(europa.eu\)](#)

<sup>3</sup> [Laki sosiaali- ja terveystietojen toissijaisesta... 552/2019 - Säädökset alkuperäisinä - FINLEX®](#)

<sup>4</sup> [https://health.ec.europa.eu/other-pages/basic-page/overview-national-laws-electronic-health-records-eu-member-states-2016\\_en](https://health.ec.europa.eu/other-pages/basic-page/overview-national-laws-electronic-health-records-eu-member-states-2016_en)



referred as RegOps<sup>1</sup>, requires new processes, tools and skills to be developed by the organizations involved in the development of medical software. The objective of the AHMED project was to research, identify, share and adopt solutions and best practices for efficient production of data-driven applications for healthcare. The AHMED project creates continuous software engineering approaches enabling efficient development of regulation-compliant applications for healthcare. In particular, the project seeks best practices for data-driven application development under medical device regulation.

The project partners include six companies (Atostek, Bittium, Mylab, Solita, Taipuva Consulting and Terveystalo). The project responds directly to the business objectives of the participating companies to develop their MDR/IVDR compatible infrastructures. The AHMED project enabled such capacity to be co-created with other companies and the two participating research organisations (VTT Technical Research Centre of Finland and University of Helsinki). AHMED is a co-innovation project financed by Business Finland and the participating organisations.

The activities carried out in AHMED roughly divide into the following categories:

- Transitioning from earlier regulation on medical devices<sup>2</sup> and in vitro medical devices<sup>3</sup> to the new MDR and IVDR regulation (Chapter 2 by Bittium, Terveystalo and Mylab)
- AI/ML compliant RegOps lifecycle development (Chapter 3 by Solita)
- Tools supporting the RegOps cycle (Chapter 4 by Taipuva and University of Helsinki)
- Large scale personal data exploitation from company perspective (Chapter 5 by Atostek)
- Research and AI/ML model development for medical software (Chapter 6 by VTT)

---

<sup>1</sup> [https://link.springer.com/chapter/10.1007/978-3-030-91452-3\\_20](https://link.springer.com/chapter/10.1007/978-3-030-91452-3_20)

<sup>2</sup> [EUR-Lex - 31993L0042 - EN - EUR-Lex \(europa.eu\)](#)

<sup>3</sup> [EUR-Lex - 31998L0079 - EN - EUR-Lex \(europa.eu\)](#)

## 2. Experiences on MDR transitions

### 2.1 MDR transition periods

The MDR and IVDR regulation include transition periods targeted to ensure that manufacturers have enough time to update their quality systems to comply with the new regulation<sup>1</sup>. At the moment a new transition scheme into MDR is under negotiation in EU and first guidelines were introduced in early December 2022<sup>2,3</sup>. Basically, the transition period is now shifting until May 2027 / May 2028 depending on medical device classification varying from the original May 2024 target schedule. However, as the basic requirements introduced for MDR still remain, the RegOps approaches developed during AHMED project are even increasingly important as they help to adapt the operations in the ever changing requirements environment.

### 2.2 Bittium's experience on MDR certification process

Bittium's medical technologies business provides healthcare technology products and services for biosignal measurements and analysis in the areas of cardiology, neurology, rehabilitation, occupational health, and sports medicine<sup>4</sup>.

During the project Bittium implemented the transition of its quality system from MDD to MDR directive. The preparation phase included the discovery of MDR requirements and extensive gap analysis during roughly one year. Based on the gap analysis preparations for the quality system improvements were addressed. All the quality management system parts were thoroughly analysed, and the original changes conducted through a comprehensive review during the first half of 2021 (Figure 1).

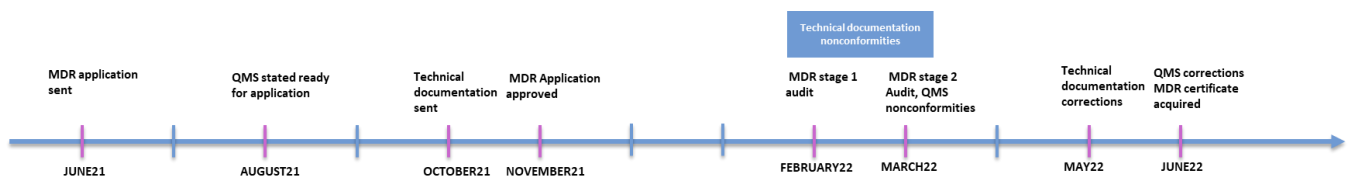


Figure 1. Timeline for Bittium MDR certification June 2021 – June 2022.

Technical documentation according to MDR was first submitted to the notified body during October 2021. The MDR application was approved based on the maturity of the documentation in November 2021. During the first quarter of 2022 the MDR stage 1 and stage 2 audits were conducted. As a result some QMS non-conformities were indicated. Based on the audits corrections for the technical and QMS documentation were addressed during quarter 4 of 2022.

In summary, the timeline from the MDR application into the point where the QMS certificate based on MDR requirements was granted was about 12 months in total. According to EU Eudamed<sup>5</sup> Bittium was the 14<sup>th</sup> company to receive the certification in June 2022. However, it has to be noted that based on information

<sup>1</sup> [https://health.ec.europa.eu/system/files/2020-09/importersdistributors\\_factsheet\\_en\\_0.pdf](https://health.ec.europa.eu/system/files/2020-09/importersdistributors_factsheet_en_0.pdf)

<sup>2</sup> <https://www.fimea.fi/-/komissio-kertoi-epsco-ministerikokouksessa-md-asetuksen-suunnitelluistalisasiirtymaajoista>

<sup>3</sup> <https://data.consilium.europa.eu/doc/document/ST-15520-2022-INIT/en/pdf>

<sup>4</sup> <https://www.bittium.com/medical>.

<sup>5</sup> <https://ec.europa.eu/tools/eudamed/#/screen/hom>

received from notified body (NB) representatives the Eudamed register information is updated non-periodically based on NB own policies so information might not be totally up-to-date.

## 2.3 Early stage reflections of AI/ML themes

The AHMED project addressed many of the challenging parts of RegOps development including the use of machine learning and artificial intelligence. Learnings and identified opportunities in relation to AI/ML approaches were the following:

- “Good data” is required to be able to make good AI/ML models (e.g. various annotations of similar phenomena)
- Regulated development of medical software requires continuous documentation, which is challenging in AI/ML context
- Use of ML/AI to support doctors in their documentary challenges and reduce the barrier to accept ML/AI
- Regulation and interaction with the notified body – especially in AI/ML context
- Applying high level of automation to regulatory compliance validation

The comments of the notified body on the use of ML were mostly related to understanding the design. For example, the NB wanted to know about use of on-line learning, transparency of the datasets and how the performance was evaluated.

In summary, a lot of data is needed for comprehensive use of AI/ML, as one needs good validation dataset and data for training the model. It is also easy to lose the focus from solving actual problems and instead juggle with the regulative issues and spend time discussing what is required by MDR and what is not.

## 2.4 MDR from Terveystalo perspective

As Medical Device Regulation (MDR) came into force in 2017 replacing Medical Device Directive (MDD) after three-year transition period, many healthcare service providers were still fast asleep as whether the changes in the regulation environment would indeed have an impact on them. Terveystalo, even though being a private health care service provider, was paying attention and quickly realized that MDR would have an impact that would have implications on how Terveystalo’s software products would be viewed and classified in the future from the medical device regulation perspective.

Timeline was still very critical for many operators in this domain even though due to Covid-19 the three-year transition period became a four-year transition period as MDD was allowed to co-exist with MDR until May 2021.

The AHMED project was very well-timed to provide its participants with relevant information and experience sharing especially concerning the very tight timelines and critical nature of the MDR. The project proved to be very useful when creating and building up a suitable medical device software development framework that takes into account the requirements of regulated medical software and agile software development.

Under the framework of the AHMED project software development processes were further developed in order to make them medical device software development compliant. This included activities such as harnessing and embedding software development tools into more integral part of the software development, including software development tool validation. Terveystalo received ISO 13485:2016 certification during the AHMED project in August 2021 which proved that software development process

along with interfacing support processes were assessed to be compliant from medical device manufacturer quality management standard viewpoint.

The project received extensions allowing all partners to carry on work until the end of 2022 or longer. The extensions were well utilized by the AHMED project participants, and in fact most face-to-face meetings with all the collaborators took place during the extensions. This is naturally mostly due to Covid-19 which prevented such meetings from taking place earlier. These face-to-face meetings, however, were extremely useful and will bear fruit in the future.

At the end of the year 2022 work in Terveystalo related to AHMED project is work in progress. This is largely due to in-house medical device software development timeline. All the information and sparring received in the AHMED project has had, and will have, an imperative role when finalizing Terveystalo's first ever medical device software product. The work started in AHMED will naturally continue in the future as medical device software development activities will be continuously improved.

## 2.5 IVDR from Mylab perspective

Mylab Oy is the only pure IVD (in vitro medical device) software manufacturer in the AHMED consortium. The MD and IVD regulations were just getting new guidance for classifications and clearer interpretations of rules, when the AHMED project started. We carefully made regulatory assessment and risk analysis for the products and found that the IVD regulation was still valid for us. It was clear that although our medical devices already had CE certifications under IVD directive the product documentation needed an upgrade to achieve new CE mark under the regulation.

Mylab's primary research question as documented in the project plan was: *"We aim to recognize opportunities for improvement and automatization in Medical Device Design and Development design processes for accelerated market entry, thus fulfil the regulatory requirements. We aim to call into question how IEC 62304 (Software Development Lifecycle) fits to current IVD/MD-Regulation and what are the possible bottlenecks when the Agile development paradigm is introduced."*

For finding answers to the research question we started comprehensive gap analysis by comparing the existing Quality Management System (QMS) (ISO 13485:2016), the software lifecycle management (ISO 62304: 2006/Amd 1/2015) and Risk management (ISO 14971:2019) to IVDR and relevant MDCG guidance requirements. We decided to refine the QMS according to gap analysis results.

The QMS Processes and instructions were written and structured against on IVDR and Standards so that the audition is clear and fast. The process artefacts were bind into instructions and Jira workflows so that they cannot be bypassed. Furthermore, we found out that it could be beneficial to have more holistic way to manage product and service lifecycle than the current system. We implemented SAFe (Scaled Agile Framework)<sup>1</sup> set of organization and workflow patterns to scale better the agile practices.

DevOps is obviously the key ingredient - the oil - in the well-aligned medical device software development. During the AHMED project we focused on making prototypes in test automation, automatic system test environment building processes and artifact as well as on version control. We are exploring the possibilities to implement continuous integration, delivery and deployment (CI, CD) pipelines. It was equally important to research container technology for the microservices. Microservice architecture increases uptime for the services and makes faster release cycles possible.

The key outcome from the AHMED project was that it is crucial to have comprehensive gap-analysis against current regulations, guidelines and standards. Furthermore, well-defined regulatory strategy for the products helps to focus on key issues and artefacts. We also found that change control could be a

---

<sup>1</sup> <https://www.scaledagileframework.com/>

restriction in implementing the DevOps in medical device development processes. The manufacturer must have a clear plan for isolating medical device modules and for controlling risks and changes. Finally, quality and DevOps are philosophies that need strong organizational culture to support them.

### 3. RegOps Lifecycle for AI/ML compliance

---

#### 3.1 Overview

Before the start of the AHMED project, Solita had developed an ISO 13485-based quality management system to address the QMS-related EU regulatory requirements for medical device manufacturers. Solita's medical device, Oravizio<sup>1</sup>, is the world's first CE-marked surgical risk assessment tool for orthopedic surgeons.

Although Solita had already made the necessary initial investments to enable the development of medical device software within the company in Finland, there were identified further development needs to answer the interest in Solita's expertise in international business. To meet the requirements of large global companies, there was a need to create a service model to enable product development partnerships in compliance with the regulatory requirements. Furthermore, to ensure that the service model meets customer expectations as closely as possible, there was an identified need to standardize working practices of product development to improve efficiency and ensure compliance with relevant requirements. The concrete result of the AHMED project was the establishment of the Solita RegOps framework, which is Solita's standardized practice for medical device software development in compliance with the EU regulatory framework.

One of the core components of Solita RegOps is related to AI/ML development practices in relation to applicable regulatory requirements. This aspect of the medical device software product realization process was particularly interesting to Solita because of the presence of AI/ML components in Oravizio. In addition, as AI technology has already become a reality in healthcare, the related development capabilities can be seen as a vital enabler for development partnerships with global companies in the field.

#### 3.2 Objectives

Solita's initial objectives related to AI/ML development capabilities in the AHMED project were:

- Identification and research of relevant regulatory requirements related to data-driven AI/ML software products to understand how they are assessed by the regulatory authorities
- Identification of different approaches for AI/ML product real-world monitoring and evidence collection in relation to regulatory requirements for post-market surveillance
- ML model re-training in a real operating environment
- Scientific publishing of relevant research outcomes and, in this way, participating in the scientific discourse and validating conclusions

During the AHMED project, the following new objectives were set:

- Understanding specific characteristics of an AI system that are of specific interest from a regulatory perspective
- Extending existing RegOps development practices to include necessary AI-specific lifecycle aspects according to the identified needs

---

<sup>1</sup> <https://oraviz.io/>

### 3.3 Key outcomes

This section describes Solita's key outcomes of the AHMED project related to AI/ML-specific aspects.

#### 3.3.1 Regulatory landscape

The EU regulatory framework for medical devices can be interpreted to consist of several layers, including the following [12]:

- Union harmonized legislation (such as MDR and IVDR)
- National legislation
- Harmonized standards (such as ISO 13485:2016)
- Guidelines (such as MDCG Guidance documents)

Even though the new EU medical device regulations seek to resolve specific problems of previous legislation by introducing new concepts regarding, for example, scientific innovations and manufacturing technologies, the aspects of AI/ML are not explicitly addressed at the legislation level. At the beginning of the AHMED project, it was expected that the Medical Device Coordination Group (MDCG) would publish a guidance document titled "Artificial Intelligence under MDR/IVDR framework," as expressed in their work program. Still, the document title was later withdrawn from their public plans. In addition, the currently known list of standards already harmonized<sup>1</sup> or planned to be harmonized<sup>2</sup> [3] against medical device regulations does not include specific AI/ML device related standards. The missing guidelines and harmonized standards lead to a lack of shared understanding of how regulatory compliance is achieved in practice in the context of AI systems.

As a result of this shortcoming, the set of commonly used standards in general medical device software development, as presented in Figure 2, must be applied also when developing AI/ML systems. In addition, all general requirements, such as MDR/IVDR general safety and performance requirements, must be interpreted in the context of AI/ML development. To support this work in practice, it is highly recommended to consult other non-binding regulatory sources for reference, such as "Good Machine Learning Practice for Medical Device Development: Guiding Principles"<sup>3</sup>.

---

<sup>1</sup> European Commission. Summary of references of harmonised standards published in the Official Journal, 2022. <https://ec.europa.eu/docsroom/documents/50115>

<sup>2</sup> 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.

<sup>3</sup> The U.S. Food and Drug Administration (FDA), Health Canada, and the United Kingdom's Medicines and Healthcare products Regulatory Agency (MHRA). Good Machine Learning Practice for Medical Device Development: Guiding Principles, 2021



|                     |   |
|---------------------|---|
| <b>Organization</b> | <ul style="list-style-type: none"> <li>• ISO 13485 – Quality management system</li> </ul>   |
| <b>Product</b>      | <ul style="list-style-type: none"> <li>• IEC 82304-1 – Health software product</li> <li>• ISO 14971 – Risk management</li> <li>• IEC 62366-1 – Usability Engineering</li> </ul> |
| <b>Software</b>     | <ul style="list-style-type: none"> <li>• IEC 62304 – Software life cycle</li> <li>• IEC 81001-5-1 - Security - Activities in the product life cycle</li> </ul>                  |

Figure 2. Relevant standards for general medical device software development

### 3.3.2 Continuous training of AI/ML in a regulated environment

One of the key technological advantages of AI/ML systems is their ability to learn, adapt and optimize operations in real time. In fact, the change capacity may be, in general, the most valuable asset of AI/ML technology. However, the concepts of change management and change control are essential medical device software development paradigms. To elaborate, the traditional paradigm for performing verification and validation activities rely on the fact that the system remains static between the releases and, when the change occurs, is re-validated prior to the deployment to the operational environment<sup>1</sup>[5]. Therefore, the real-time adaptation breaks the paradigm of being able to maintain a validated state. As a result, the regulatory authorities have traditionally promoted the approach of “locked” AI/ML algorithms<sup>2</sup>, where systems are trained during the development phase, and the ability to improve is disabled in the actual operational environment.

<sup>1</sup> AAMI. AAMI Consensus report, Appropriate Use of Public Cloud Computing for Quality Systems and Medical Devices. AAMI/CR510:2021. 2021.

<sup>2</sup> 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>.



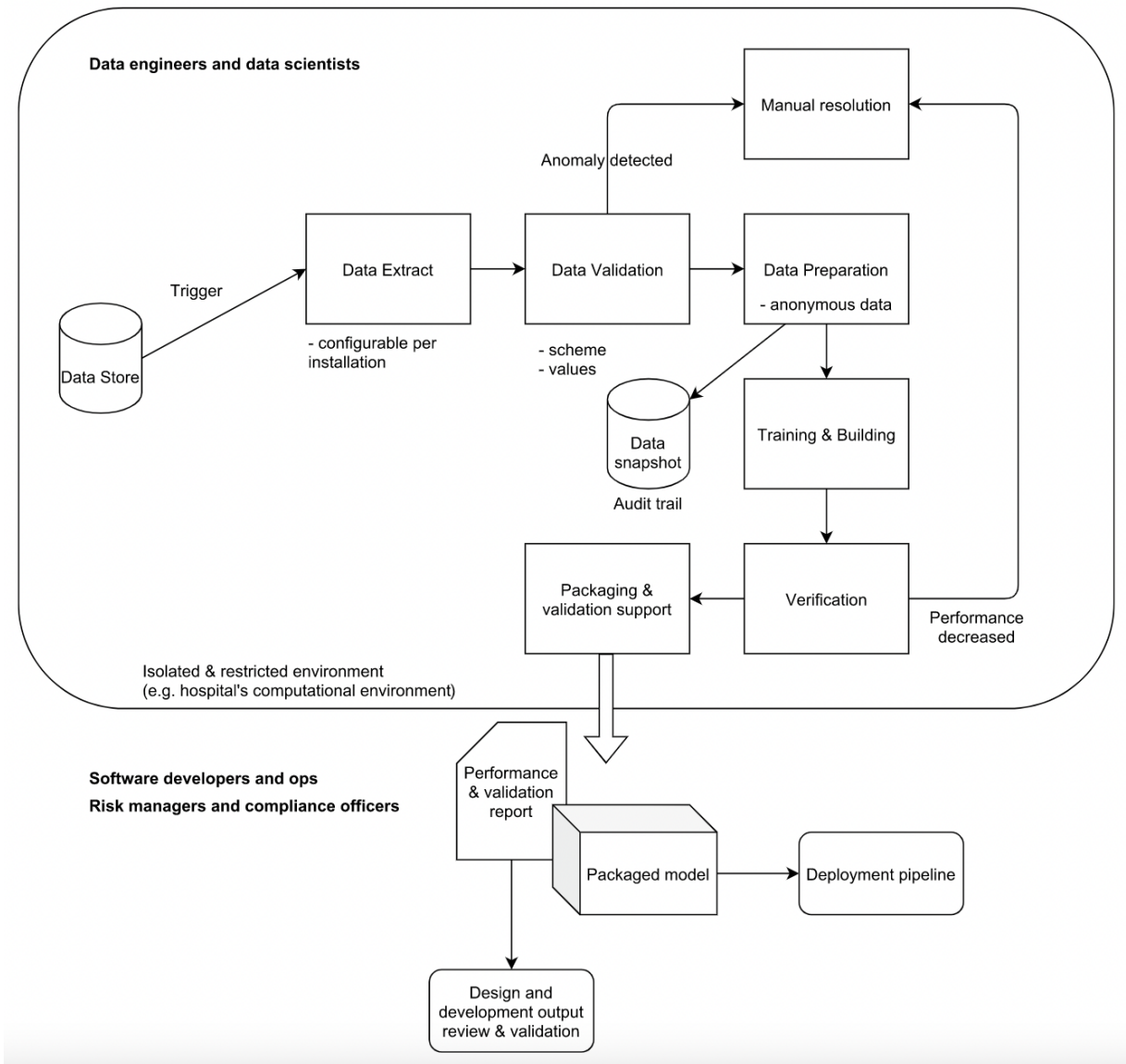


Figure 3. Continuous training pipeline [12]

During the AHMED project, we revisited the development project of Oravizio to enhance the maturity of the MLOps practices used in the development to ensure compliance against new MDR requirements in the future and to improve quality assurance activities further. Our journal article “Towards Regulatory-Compliant MLOps: Oravizio’s Journey from a Machine Learning Experiment to a Deployed Certified Medical Product” reports the complete research [12].

The primary outcome of the research, a continuous training pipeline, is presented in Figure 3. For regulatory reasons, Oravizio was designed to be deployed to the operational environment in a “locked” state. However, there was an identified need to be able to re-train the ML model based on the new data. The specific challenge was related to the fact that the data was privacy-sensitive patient information and, therefore, could not be accessed and used without special arrangements. Working in an isolated and restricted computational environment, the automated training pipeline minimizes the need for human interaction and access to restricted data. The pipeline produces the re-trained ML model and related automatically generated performance reports in an anonymized form. As a result, the deliverables can be distributed to the development team. The main difference to the general MLOps practices is that deploying the model to the operational environment requires manual actions and approval.

### 3.3.3 RegOps Lifecycle for AI/ML

During the AHMED project, we identified the need for new objectives to be achieved within the project context. First, as there is a lack of regulatory requirements that specifically address AI-related aspects, we needed to identify specific characteristics of AI systems that are of interest from a regulatory perspective. This task was carried out using an informal literature review of various guidance documents on the subject from different stakeholders and regulatory authorities. In addition, relevant AI/ML-related standards from other industries were examined. However, it is worth noting that if relying heavily on a document outside the EU regulatory framework to demonstrate compliance, the use of the document may need to be justified in the device’s technical documentation. Finally, the already established RegOps development model was extended to include the identified AI-lifecycle aspects.

There are certain advantages to using a general medical device development model as a baseline for AI/ML extension. The AI/ML systems are, by nature, a combination of “traditional” software items and AI/ML-based software items. Therefore, general processes are needed in any case to develop the system. In addition, as discussed before, also the AI/ML items are a subject of the general regulatory design controls. Furthermore, re-using the already battle-proven existing practices is beneficial when aiming for effective process implementation.

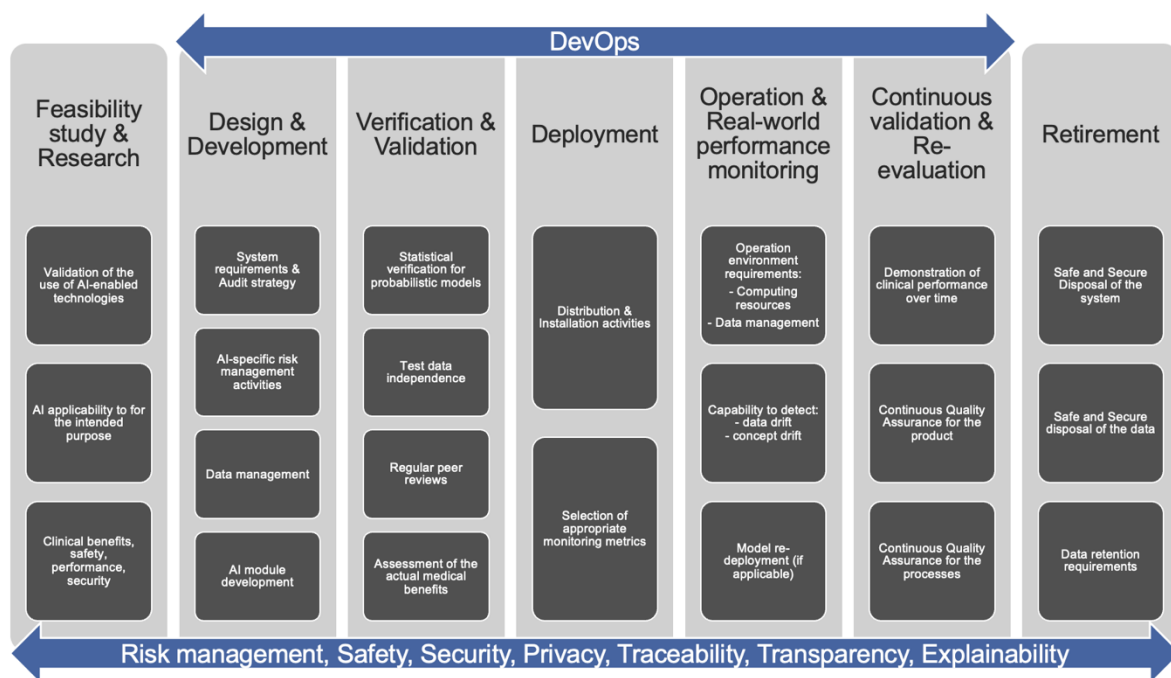


Figure 4. AI-specific addition to general medical device software development lifecycle process

The AI-specific additions to Solita’s RegOps development model are presented in Figure 4. The model is loosely based on the general idea of the AI system lifecycle model presented in the draft version of standard ISO/IEC DIS 5338:2022<sup>1</sup>. It is worth noting that Solita’s RegOps software development model is currently under active development, especially as regards AI-specific aspects presented in Figure 4.

The model addresses specific AI-related aspects that are especially relevant from the regulatory perspective but are currently not explicitly covered within the requirements. From the point of view of the

<sup>1</sup> ISO/IEC DIS 5338 - Information technology — Artificial intelligence — AI system life cycle processes. Draft version, 2022.

intended clinical purpose of the system, the use of AI-enabled technology needs to be justified and proven safe, clinically effective, and secure.

As AI systems may be more complex than traditional software systems, there needs to be an audit strategy that allows the conformity of the product to be assessed. As a result of the increased complexity, regular peer reviews need to be implemented within the development workflow to minimize the probability of mistakes made by a single person. Furthermore, especially ML-enabled applications may be data-intensive, and the data can be interpreted as being a raw material for the manufacture of the product. Therefore, data management practices need special consideration and alignment with the regulatory requirements. Finally, as the change dynamics of an AI-based product can differ from a traditional software system, the system's risk profile may be more dynamic by nature, and risk management activities need to be tailored to address the system's specific characteristics.

The EU regulatory framework contains comprehensive requirements for post-market surveillance activities; hence, the real-world performance monitoring of an AI system requires particular attention. The most important factors are related to the capability to detect data and concept drifts of the system during its use. Naturally, these capabilities are based on selecting appropriate and relevant indicators from the viewpoint of clinical performance, safety, and security. Due to the dynamic nature of the AI/ML system, continuous quality assurance is critical.

### 3.4 Business opportunities

The AHMED project has allowed Solita to develop our RegOps framework and competencies further, differentiating us from our competitors and has helped us gain new customers.

With this new knowledge, we have, for example, helped our customers to significantly improve their medical device software processes. We are expecting much growth in this business area. However, we have also realized the size of the problem, e.g., the digitalization maturity in some sectors of the health Industry like big pharma.

## 4. Methodologies and tools for developing medical device software

---

### 4.1 Overview of activities

Agile methodologies have become the way to develop software intensive products in a wide range of industries. Relying on fast feedback, responding quickly to changes originating from various stakeholders, and employing continuous software engineering practices allowed agile teams to deliver at increasing pace. The result is high quality software products that fulfils the user's needs. Despite of these advantages, many companies developing medical device software have been slow in adopting agile methodologies and practices. The lean manner in which agile is often used lends to the perception that it lacks the discipline of the traditional plan-driven methodologies, such as waterfall or V-model. As a result, agile is perceived unsuitable to handle key regulatory requirements such as risk management, or validation, verification and traceability.

We conducted research activities in two areas of interest. First, we investigated current mainstream continuous engineering ways of working to identify best practices that can serve as candidates for developing processes for medical device software development. Secondly, we looked at the working practices of data engineers and data scientists to identify how these specialties should be integrated into the strict processes required for medical software. As a supporting activity, we developed tools that assist the implementations of these processes using mainstream off-the-shelf tools. Both activities have been conducted together with relevant industrial partners, resulting in scientific publications and open-source technology demonstrators.

Tool chains of software engineering were investigated and further developed. Polarion tool was equipped with a process implementation that facilitates agile documentation and management which is essential in integrated RegOps cycle. The total process needed to be simple and fast to use, also combining the actual agile work management, in order to be really useful for software developers. The resulting project outcome meets these expectations very well.

In a joint work between Taipuva and University of Helsinki workflow automation was taken all the way to unite DevOps in GitHub and documentation of changes in Polarion. There were encouraging comments from the consortium members and other medical device manufacturers that such an automation is good for improving efficiency and making sure that documentation keeps in sync with actual code changes without forgetting something or laborious manual checking afterwards.

What shall not be overlooked is that the tools and processes used in the medical software development (i.e., design work) need to be validated. As part of the project a clear and reusable approach for the tool validation was created.

### 4.2 Medical software development processes and regulatory Background

EU's Medical Devices Regulation (MDR) and In-Vitro Diagnostics Regulation (IVDR) define some key requirements for design and development as well as set requirements for risk-based approach and post-market surveillance, among other things. They require a quality management system according to ISO 13485 (the US counterpart being the very similar content in FDA CFR Title 21 Part 820) and risk management according to ISO 14791:2019. Specific for software development, one needs to consider usability according to IEC-62366-1. For health applications it is also advisable to follow IEC 82304-1.

In practice, the unharmonized standard for software life cycle processes (IEC-62304:2006) is the state-of-the-art reference. It defines the different process areas of software development in more detail and sets requirements for different risk class software.

For software development tool chain validation the requirements come from ISO 13485, more specifically from clauses 4.1.6, 7.5.6 and 7.6. Additionally, guidelines are set forth in ISO/TR 80002-2. In this sense the US requirements of FDA CFR Title 21 Part 820 and AAMI TIR36:2007 (Validation of software for regulated processes) are practically identical. For electronic signatures and electronic archiving the state of the art is according to US FDA CFR Title 21 Part 11.

Following process areas for medical software development are identified according to IEC 62304:

- SW Risk Management
- SW Development Planning
- SW Requirements Analysis
- SW Architectural Design
- SW Detailed Design
- SW Unit Implementation & Verification
- SW Integration & Integration Testing
- SW System Testing
- SW Release
- SW Configuration Management
- SW Problem Resolution

There are few very fundamental requirements that govern the design and development, as well as documentation – also called as technical file or device history file:

- Traceability between design artefacts, for example tests to requirements and risk-mitigating risk control measures to hazards  
(for some risk classes the traceability needs to extend between code and its design artefacts)
- Change control
- Documentation control, so that there are defined design reviews and all documentation is reviewed, approved and signed off by electronic or paper signatures

Besides the mentioned product safety and effectiveness concerns [16], MDR and IVDR mandates manufacturers to handle cybersecurity concerns during product development lifecycle [17].

### 4.3 True RegOps cycle

The usual DevOps cycle does not mention documentation or process control. And why would it? These areas are something that are not emphasized or even needed in agile development, and that's one of the contradictions to regulated industries. Namely, in medical devices development one needs to present and maintain a lot of documentation as described in Section 4.2. Process control comes into the picture, as one needs to be absolutely sure that organization's defined quality processes (standard operating procedures, SOPs) are followed, and written evidence is left behind (which makes up the audit trail). Some medical device manufacturers estimate that the documentation activity takes more than 70% of the total work effort of a usual medical software development project. Therefore, it is not in vain to think how this part can be done more efficiently than today.

For the above-mentioned reasons, an agile RegOps cycle as depicted below was driven forward within this project. The basic idea is that there is no development activity that would not be appropriately documented, and there is enough process control to ensure it.

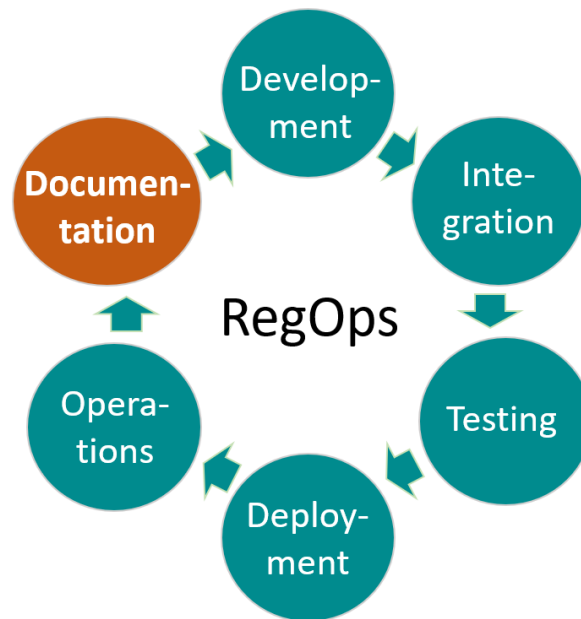


Figure 5. RegOps cycle where the needed documentation activity is an integral part of agile changes.

Process control is something that can be performed in many ways. The traditional way is to apply SOPs and working instructions, and combine them with, e.g., check lists. These typically involve phasing series of activities with so-called quality gates, manual activities and remembering things. The following section describes how tools come into the play in this context.

#### 4.4 Tools

Tools in the RegOps space can be categorized according to their principal “position” in the RegOps cycle, which includes at least the following areas (Figure 5):

- A. Planning, management & documentation
- B. Software development
- C. Integration
- D. Testing
- E. Deployment
- F. Operations & monitoring

The preceding section introduced the concept of process control. Tools can help in this by employing workflows, which ensure that certain tasks are done and which guide or even force activities to take place in a definitive order. Tools can provide automation to workflows. Examples of automation are conditional checks that some required piece of information exists, or there can be automatic workflow actions that take care of multiple things with a press of a button. More complex workflow automation combines several simple workflows together, for example approving a specification document makes all the individual requirement objects in the document approved in their own workflow. At its best RegOps can involve workflow automation that connects different domains and areas together. Typically, this requires that tools of different areas are integrated. In many cases, the gaps requiring manual labor and checking are at the interfaces of *different* tools. These interfaces are also the place where traceability is most often compromised.

Workflow automation is something that can considerably speed things up, but sometimes it is also a two-bladed sword, when it forces a one-size-fits-all solution to all cases. If substantial amount of human intervention is needed to “fix” automated things afterwards, then it is of no use. Therefore, automation must be carefully designed, and often applied to certain process points which are seen feasible to be standardized.

#### 4.4.1 GitHub and GitLab – software development, version control and task management

GitHub is an internet hosting service for software development. Besides the basic source code version control that leverages Git, the service provides capabilities for change control, task management, bug tracking, continuous integration and wiki. A recent capability serves as a basic enabler for project management using continuous development methodologies, like Scrum or Kanban, custom reports categorizations using labels.

Besides the capabilities available via the web user interface, GitHub provides two extension mechanisms that enable 3<sup>rd</sup> parties to augment the native functionality. The first consist of APIs (REST/GraphQL for accessing resources and Webhooks for notifications) which allows the creation of standalone services. The second consist of actions, which allows custom container to be run inside GitHub’s infrastructure. Both mechanisms can be used to augment the GitHub’s capabilities with functionality provided by other services, allowing medical software manufacturers to develop holistic RegOps toolchains. The built-in capabilities and the extension capabilities can be used successfully to develop requirements to implementation traceability [18].

GitHub provides its functionality via the cloud, using a SaaS model, or on-premises (e.g. GitHub Enterprise). Historically, new features are released to the cloud before being available to the enterprise customers. GitLab provides similar functionality. Gitlab offers cloud and on-premises installation options (see also Section 6.3.2).

#### 4.4.2 Polarion – for planning, management and documentation

Polarion<sup>®</sup> is an Application Lifecycle Management (ALM) tool originally developed by Polarion Software, and after an acquisition in 2016 being a part of Siemens Digital Industries Software’s portfolio. This tool was originally designed to meet the requirements of regulated industries, and some medical device manufacturers gave feedback early on to influence its core principles and key features. Software and product development teams use it for requirements management, architecture specification, risk management, test management, change management, configuration management, project management and agile work management. Siemens Polarion is a scaled agile partner platform and it supports Scaled Agile Framework<sup>®</sup> (SAFe) 5.0 – as well as smaller scale agile methodologies with its highly configurable information model, supported by Kanban boards and Scrum burndown charts.

Polarion is a commercial tool, which can be run on cloud (SaaS), hosted anywhere and even on-premise in a closed network. It’s guiding principles are openness and a non-proprietary information model in the backend. Various interface (e.g., REST API) options make it easy to integrate Polarion as part of the RegOps tool chain. Connectors to most popular tools, such as Jenkins, exist out of the box. Connectors to Jira and MS Azure DevOps are there to support co-existence and migrations.

The commercial model for obtaining Polarion is based on user licensing only, i.e., there are no separate fees for server or database. The cost of a user license varies roughly from 5 to 100 € per month, depending on the user’s functionality set. Named and concurrent (floating) licenses are available, and bigger organizations tend to have a lower unit cost due to exploiting concurrent licensing effectively.

Polarion is often relied to for all planning, management and documentation aspects because of its very wide process coverage. There are no plugins or additional maintenance needed to fulfill the regulatory requirements. Key capabilities that make Polarion suitable for regulated industries:

- All processes on one platform enable seamless traceability and transparency
- LiveDoc™ – online documents with trackable objects, for integrated document management
- Full history and versioning of documents, with the contained objects consistently
- Audit trail
- Electronic signature (FDA 21 CFR Part 11 compliant)
- Configurable workflows with automation for actions & conditions and freezing content
- Workflows allow interactions with 3<sup>rd</sup> party tools
- Versatile reuse features that allow reusing the latest or historic information, even with links preserved between object structures

#### 4.4.3 JIRA, Confluence – agile project management and documentation

JIRA enables agile project management features (see also Section 6.3.2). Confluence is an enterprise knowledge management system that enables maintaining complex documentation using the simplicity of the wikis. Both services are provided by Atlassian, and can be integrated with GitHub, GitLab and Polarion.

## 4.5 Out-of-the-box process for software development

Each organization developing software classified as medical device, or possibly to be classified as such in the future, faces the problem of knowing how to be compliant. The next phase is to implement the capability for doing so and getting it rolled out to the organization. The third phase is – if the organization ever gets there – is to analyze if their ways of working are efficient and how to improve. Doing this all from scratch might take years and several hundreds of thousand euros, and even it may not be fully digital or as efficient as it could be.

Before AHMED project Taipuva Consulting had been working a long time with Siemens Polarion tool. Polarion, as described above, is an ideal platform to support RegOps in its areas for planning, documentation and agile work management. Polarion is the market leader in ALM tools in the Nordic Countries: more than one third of big and medium-size medical device manufacturers use Polarion.

No matter how capable a tool is – it is just an empty shell without knowing how to use it. There needs to a smart configuration on it to be able to deal with *all* process areas required by IEC 62304, and to be able to keep everything simple – as well as agile. That was the ultimate goal within AHMED project: to perfect the process concept and make Polarion realize it in an optimum way.

The key result was a modern and all-digital process package (Figure 6). On the background there was a profound understanding of the latest regulatory requirements (MDR), as supported by the project. Solution's highlights include:

- Simple total approach, with years of experience how to fulfill regulations and make it agile
- Efficient workflows for carrying out needed documentation reviews and approvals, with integrated electronic signatures.
- Software architecture and detailed design incorporated into the process so that the traceability to requirements, risks and V&V is intuitive.
- Risk management being able to deal with both safety and cybersecurity risks and incorporate their complex traceability relationships to other design artefacts.



- Agile work management realized with a simple approach that is suitable to team-level, but which is very easy to adapt to the preferences of organizations enterprise agile.
- Verification test management suitable for fast testing results reporting, with a custom widget extension that enables agile test planning.
- Flexible and versatile traceability reporting incorporated (see Section 4.6).

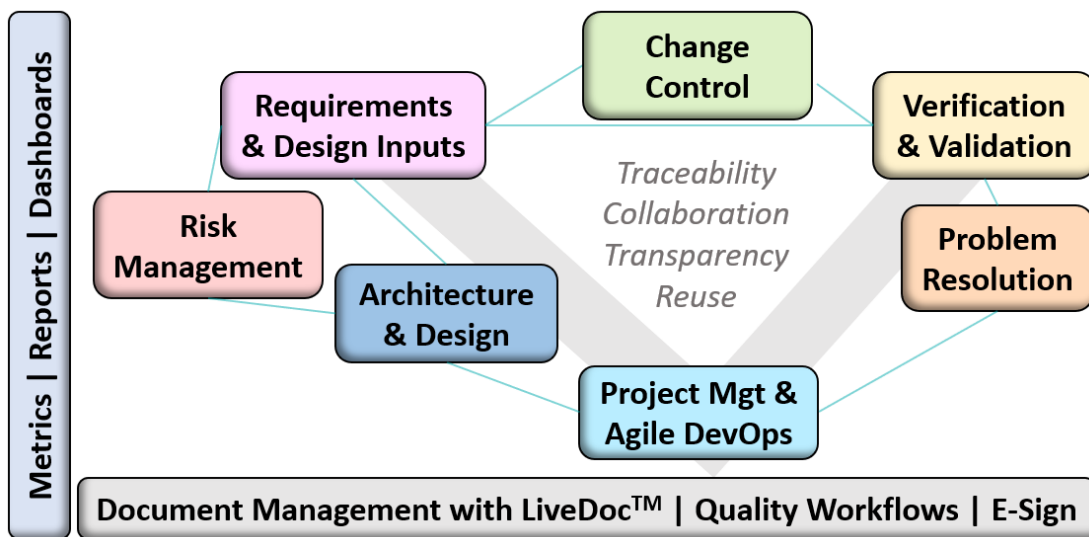


Figure 6. RegOps full-coverage process package developed on Polarion tool by Taipuva.

The process package is now commercially available from Taipuva, thanks to AHMED project. A brand-new concept, Design Controls as a Service, was introduced. It means that organizations are able to use it as a secure service, just via a browser and internet connection. It can be launched immediately and with a short training any organization can use it. Another alternative is to engage in a service relationship with Taipuva. It allows a customer to adapt the process configuration with quick changes to match it to the organization’s preferences. There are also designed services available for data migration from legacy sources, in order to get fast into speed.

The benefits of using this kind of all-digital and efficient process on a capable tool platform are unparalleled. Companies needing traceability in their design work have reported saved effort of 80 % in maintaining traceability and more than 20 % in total. In MDR and other quality audits there have been no non-conformities regarding design processes or created technical file. When writing this final report, two new customer relationships, one in Denmark and one in Finland, are starting that make use of the newly created product.

## 4.6 Traceability

Traceability between design documentation artefacts is a key regulatory requirement and one of the central pain points, because it usually takes so much effort to maintain. Companies have estimated that working with traceability information can take 20-25 % of project effort. Traceability matrices are also becoming bigger due to having more software features in products. With the interrelations between safety and cybersecurity features (and risks at the same time), the structure of design traceability is becoming more

complex – not only having a classical tree structure but a grid of objects, dependencies going back and forth.

Originally the regulatory requirement for traceability was meant to ease tracking of dependencies and the impact of changes in the design. FDA records reveal, year after year, that changes to software after the initial product launch are a cause of many issues in product safety. This is understandable if traceability is followed and updated manually. Even a “few” manual links among thousands automated, leave too much room for human errors. Therefore, 100 % of links need to be covered *seamlessly* – otherwise automation is of little use.

The above-described complete design process on Polarion tool puts all information on one platform, which handles 100 % automatic coverage for traceability. At the same time, Polarion takes care of audit trail and historic traceability, thanks to its full and consistent version history. Taipuva’s process package has a very carefully designed information model that uses directional links of different types. Each link has a semantic meaning. Polarion’s basic feature is filtering information based on link types. This makes it possible to see selected relationships of the information grid at one time, which offers designers to focus on what they need to see at each time.

As explained above, the demands for traceability information are becoming too difficult for even the best out-of-the-box tool features. When there are thousands of objects and easily more than 10,000 links between them, designers need even better views to control all of it. Imagine an extensive software system consisting of multiple layers, applications, modules and components. In time there will be various versions of each of them, and perhaps also variants for different customer deliveries. For one thing, the information needs to be organized and categorized well. Once this has been done, the used tool shall serve designers easily in the daily work. These needs Taipuva Consulting has seen during more than 10 years of working with medical device manufacturers. AHMED project offered a perfect framework to develop a solution that will help organizations to take benefit of design traceability information – today and in the future. A new, flexible and user-friendly traceability matrix view was developed on top of Polarion (Figure 7).

| User Requirement                                    | Status        | System Requirement<br>Technical Requirement                        | Status        | Test Case   | Test Run                              | Test Record  |
|---|---------------|--|---------------|---|---------------------------------------|--|
| METE-907 - Insulin storing                          | ✔<br>Approved |  |               | Missing Test Case!                                      |                                       |  |
|   |               | METE-1235 - Container needs to store 70ml for 1 months normal use. | Draft         | Missing Test Case!                                      |                                       |  |
| METE-908 - Gauge for amount of insulin in reservoir | ✔<br>Approved |  |               | Missing Test Case!                                      |                                       |  |
|   |               | METE-644 - Gauge for amount of insulin in reservoir                | ✔<br>Approved | METE-753 - Gauge for amount of insulin in reservoir tc4 | 20210518-1513 - Tapio Acceptance test | Failed (2021-05-18 15:16)<br>Comment: Insulin measurement does not work if the reservoir is completely empty |
|   |               | METE-868 - Diagnostics logic for insulin gauge sensor              | ✔<br>Approved | Missing Test Case!                                      |                                       |  |
| METE-910 - Basal dose                               | ✔<br>Approved |  |               | Missing Test Case!                                      |                                       |  |
|   |               | METE-636 - Set basal dose  | ✔<br>Approved | METE-686 - Set basal dose                               | 20221201-1226 - From Document         | ✔ Passed (2022-12-01 12:35)<br>Comment: All worked according to specs  |

Figure 7. Traceability matrix view on Polarion

Taipuva’s Traceability Matrix has the following advanced features:

- Clear and logical representation, end-to-end to verification/validation results.
- Generic approach that is not tied to certain process configuration.

- Flexible matrix structure to select displayed objects and their information fields.
- Quick filters for all layers of information (which can be added easily if there are new needs)
- Smart quality indicators that reveal:
  - Missing test coverage
  - Unhandled changes
  - Obsolete test results
  - Incomplete information because of user's limited access
  - Process violations
- Ability to span across different "projects" (which typically are subsystems and components that make up the total product).
- Exporting to PDF and Excel formats – to support organizations that do not want to rely on one solution as an electronic archive

Even though filtered and precise information is useful for designers and testers, for regulatory compliance it must be possible to represent the full product traceability matrix, in order to prove that there are no gaps. For complete products, the full matrix consists of thousands of rows. The solution generates such a report on demand in real-time which is difficult to achieve and cannot be not taken for granted.

It is worth mentioning that the matrix is able to run for a given time point in history. This means that exported snapshots of traceability do not need to be stored separately for regulatory compliance purposes. Everything is available electronically, should the auditors desire to see it.

As a conclusion, traceability information is so essential that it is the first thing auditors usually want to see. As discussed, it needs to be represented very differently in different cases. It is the only way to make real use of the traceability information – actually help designers in their work, instead of being a burden. Taipuva's solution is available to enhance productivity and to help designers ensure safety and cybersecurity, as well as prove compliance with a press of a button.

## 4.7 Reusable process for tool chain validation

It is required by the ISO 13485 quality management standard that design tools must be validated, so that patient-safe and cybersecure design as well as the correctness of product documentation can be ensured. The validation itself for an unexperienced organization can take roughly one year or longer and cost more than 100,000 euros in work effort.

As part of the project Taipuva Consulting created a reusable approach for the tool validation. The validation process was designed according to the guidelines of ISO/TR 80002-2. It was realized on Polarion tool as a project template. As an example of the process feasibility the design controls package described in Section 4.5 was validated.

The validation process makes use of Polarion's work item object configuration, workflows and test management capabilities. It also contains document templates and base documentation. Therefore, the validation package can be used as a basis and a trusted process for *any* tool validation – just by completing documents defining intended use, specific requirements, risks and validation test cases. The general documentation structure is represented in Figure 8. It is designed to minimize the effort in the case of re-validations. Documents can be reviewed and approved electronically. Polarion's test management allows to execute and report the defined tests, and the automatically generated report becomes a part of the documentation, as well as automated reports show the needed traceability between risks, requirements

and tests. In the very end, conclusions and the final result are stated in the validation report. In case the validated software is changed the revalidation can be done with minor work effort.

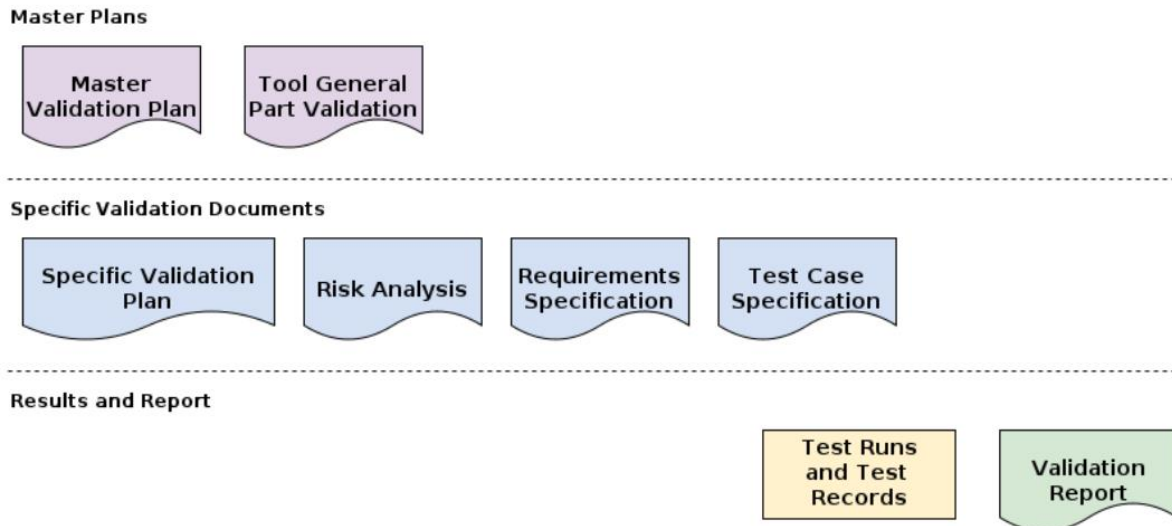


Figure 8. General documentation structure for medical software.

Taipuva is now able to offer a ready-made validation process as a service – and apply it to any part of the tool chain in cooperation with the customer. Cooperation is always needed, because tools and processes have a strong link to organizational quality system definitions. The first commercial tool validation for a globally operating medical device company has been delivered using the results of this project.

The biggest benefits of such a validation package and service come for organizations that have not done tool validations before, or who have a lot of them or if they need to repeatedly re-validate. Instead of months or even years spent in tool validation we are talking about weeks. In the case of Polarion or very similar tool validation, it can just take just a few days.

#### 4.8 Challenges in building multitool RegOps tool chains

Building effective RegOps toolchains requires different people with specialties to work together. Traditionally, following a waterfall process, the regulatory specialists and the software developers have performed their activities at different stages of product development, using different tools. As innovation in medical devices is driven more and more through software, the appeal of agile software development methodologies increases. The iterative and incremental nature of these methodologies brings the regulatory specialists more and more in contact with software developers, as a result they need to perform regulatory activities efficiently using DevOps style practices [19]. To achieve this goal, the tools used by each specialty need to integrate together and the processes need to be aligned [20].

To achieve this goal, Taipuva and University of Helsinki performed a series of proof-of-concept experiments and gathered feedback from medical software manufacturers on using RegOps practices. The experiments involved an imaginary example of developing a medical software that had a web interface to access the medical data stored in an SQL database. High-level product planning and documentation was managed in Polarion, while the software development was implemented in GitHub. The integration of these two domains made it possible to couple the workflows and tie together the regulatory documentation and development activities.

We considered two change scenarios, which are depicted below.

1. Top-down (Figure 9): A change is initiated from the specifications point of view by creating a change request in Polarion, approving it and then proceeding to both development (in GitHub) and updating documentation (in Polarion) as synchronized activities.
2. Bottom-up (Figure 10): A change is initiated from the code point of view by creating an issue in GitHub, connecting it to a change request in Polarion. Development (in GitHub) and documentation (in Polarion) can be started after the approval of the request.

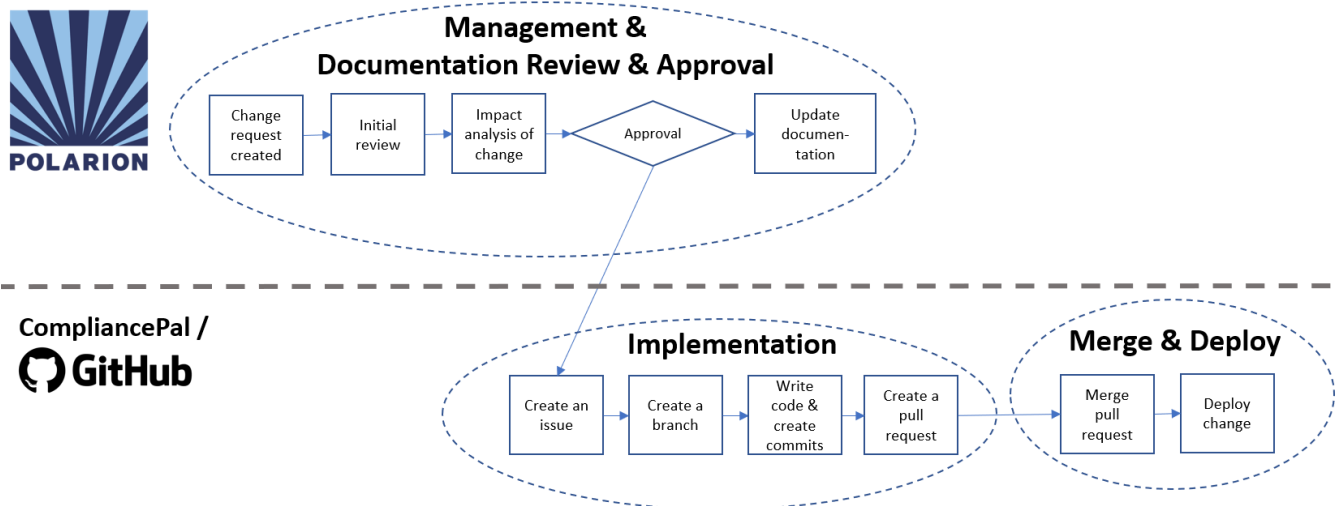


Figure 9. Scenario 1 of a top-down agile change.

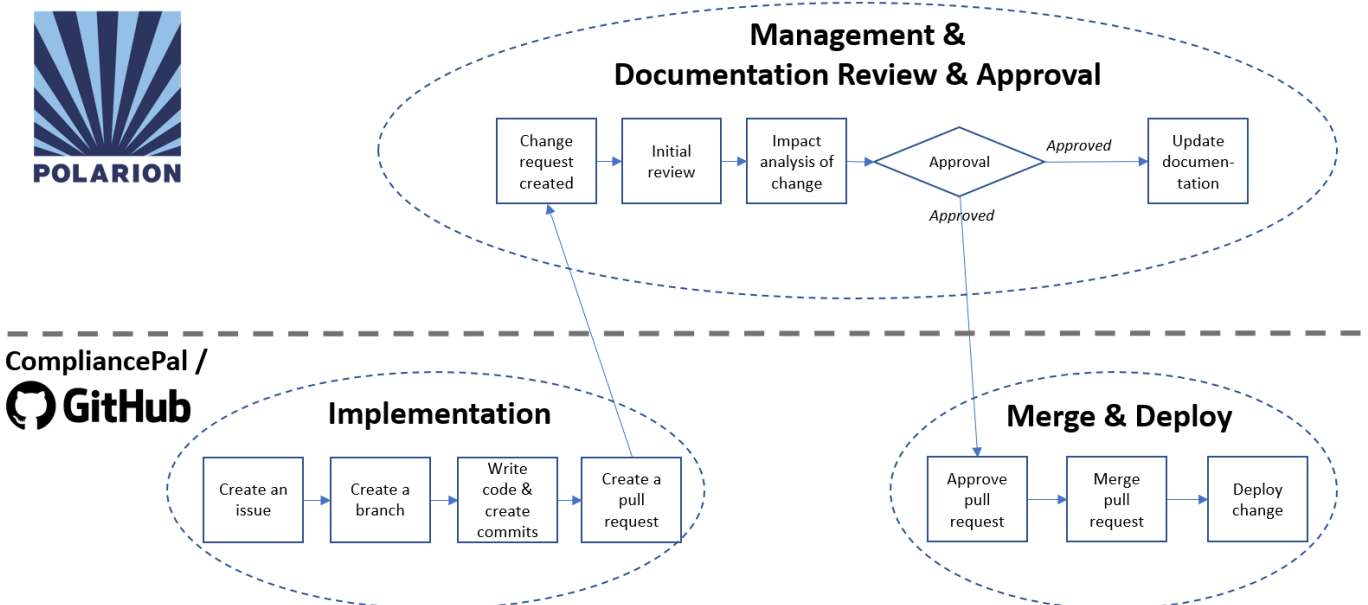


Figure 10. Scenario 2 of a bottom-up agile change.

The two systems were connected via a backend service that mapped the Polarion’s *work items* to GitHub’s *issues* and *pull requests*. The backend service monitored both Polarion and GitHub, created the corresponding mapping and transferred the updates done by the regulatory specialist or the software developer. Additionally, the workflow control mechanism specific to Polarion (work item status) and GitHub

(e.g. pull request review state) were synchronized, preventing the workflow to proceed unless both tools were aligned.

The results revealed that although it is technically possible to realize an API level multitool RegOps tool chains (Polarion-GitHub in this case), the integration and concept mapping between the tools is not straightforward. Further study is needed to determine how to better transform the dynamics from one tool to the other, in a meaningful way for all roles involved.

## 4.9 Considerations on using ML technology in medical devices

AI/ML technology introduces additional challenges when used in a regulated environment, compared with traditional software. Besides software engineers, AI/ML technology requires two additional specialities: data scientists for developing the models and data engineers for preparing the data used for training. As the end result of these development activities are included in medical software as software components, the IEC 62304 needs to be adapted and applied to data engineering and data science development activities. Manufacturers need to ensure they have enough provisions to ensure lineage for the datasets used for training and model development experiments are tracked accordingly.

### 4.9.1 Model cards as ML development ledger

Model cards are a proposal from Google Research [21] that aims to collect in a single place the relevant information about a model development .

The model card approach consists of a metadata document that captures:

- Administrative information: owner, version, or license
- Model parameters: inputs, outputs, and datasets used for training
- Quantitative information: performance metrics that can be used to monitor drifts
- Considerations: comprehensive documentation about the intended users and use-cases, limitations and tradeoffs, as well as ethical considerations.

The metadata document can be used to create specific documentation intended for developers, end users or regulatory purposes. The evaluation of the suitability of the model cards approach was implemented in the context of continuous development for machine learning (CD4ML), a reference implementation of automating the ML system life cycle in an end-to-end fashion formalized by Thoughtworks<sup>1</sup>. The model card is able to capture the versioned training, validation and test datasets, the selected model out of the experiments tracked in MLflow, as well as monitoring the model performance based on the expected quantitative parameters [22]. The model card is an effective development ledger that supports both engineering and regulatory needs.

---

<sup>1</sup> <https://martinfowler.com/articles/cd4ml.html>

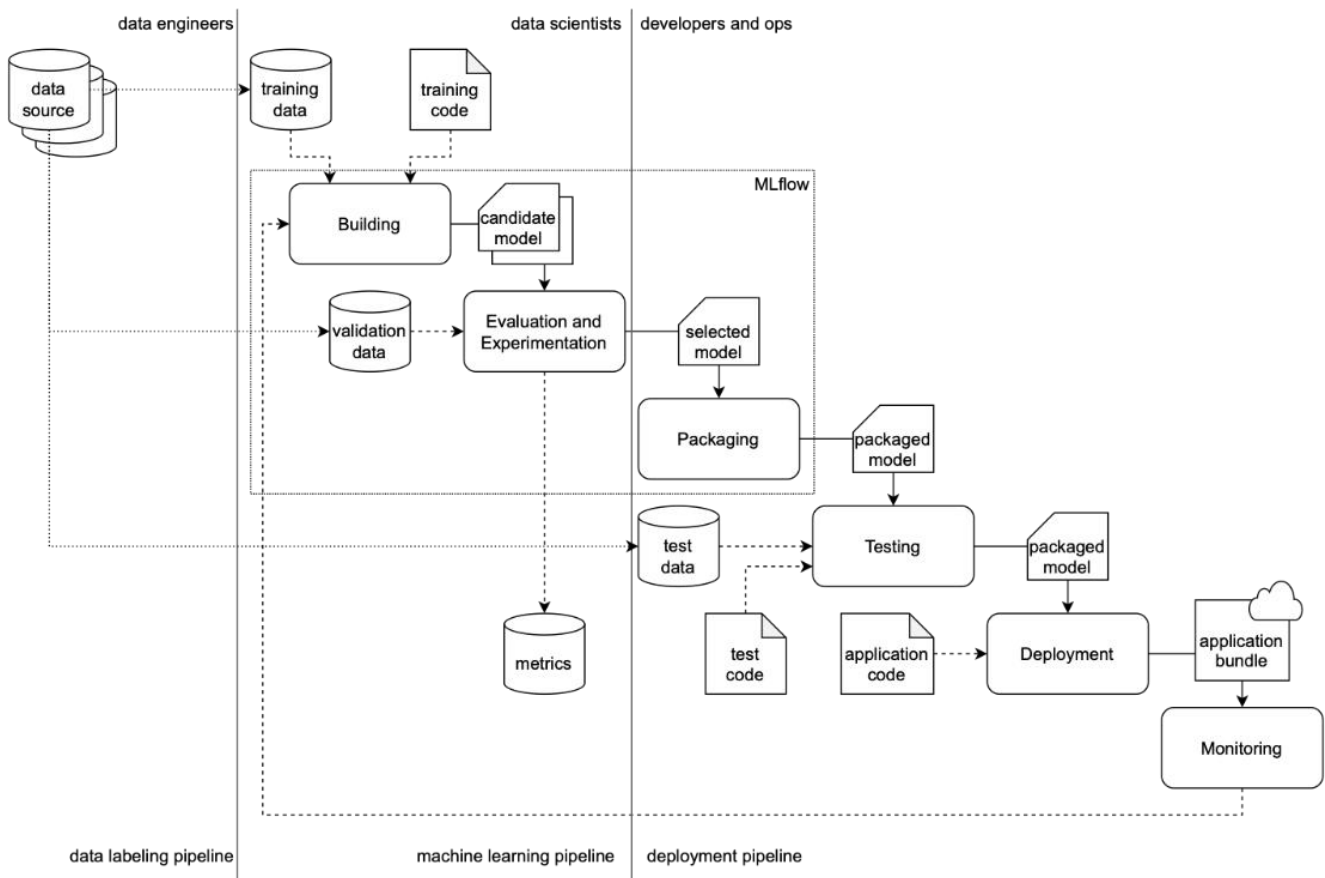


Figure 11. CD4ML: Continuous development for machine learning

#### 4.9.2 Multi-organization ML development

Developing ML technology in regulated environments may introduce specific challenges that complicates the development process. For example, in the Oravizio case (see also Chapter 3), due to privacy concerns related to the data used for training the model, the model is developed in an isolated environment then transferred to the software developers for integration. In practice, the Oravizio's development boundaries resemble a multi organization setup [23]. From a regulatory perspective, the development processes and practices across these boundaries needs to be properly documented using model cards or similar techniques.

For the cases where the ML model development practices cannot be assessed, the models should be considered SOUP ML [24], in the same way as any other software components not developed according to MDR/IVDR compliant practices.

## 5. Data exploitation challenges

---

### 5.1 Overview

As part of the project AHMED, Atostek has worked on developing new ways to use big health data to create innovative software solutions for healthcare [6]. The related sub-project is named Jasmine. The project was inspired by the Kanta archive<sup>1</sup> which is the national electronic health record system of Finland, and the Act on the Secondary Use of Health and Social Data (552/2019), which allows researcher access to health and social data. The Kanta archive contains a wealth of data that could be potentially used in development of new health related software solutions. The data application process from Findata (Finnish Social and Health Data Permit Authority) turned out to be much longer than initially anticipated. Thus, the project Jasmine is still a work in progress and extended until 31.12.2023.

### 5.2 Goals

The initial goals set for the project Jasmine were:

1. Identify health data driven use cases that are considered valuable by medical professionals
2. Gain research access to health data in Kanta
3. Use artificial intelligence and machine learning methods to realize the use cases
4. Prepare a demonstration of the data driven use cases outside the research environment

An additional consideration in the project Jasmine is that it should be possible to continue after research project with R&D activities, that aim to create actual software products. Thus, in addition to the research activity also working technical solutions, discovering and solving possible obstacles related to regulations, privacy, and data availability must also be considered.

### 5.3 Methods

Methods used for identifying valuable health data driven use cases were brainstorming, interviews of medical experts, and analysis of data available in the XML format used by the Kanta archive. The found use cases were used as basis of the research questions required by the data application to Findata. The key issue for the data application was to get the data in original XML format as that is the format available to operational health care software. Once the data is available, we plan to make an analysis of the actual data resources in Kanta, for example the amount of structured data vs. non-structured data. Methods planned for data analysis include natural language Processing (NLP) and deep learning for automated analysis of textual data, and automatic classification methods for structured health data. The practical demonstration is planned to be a simple replication of the found solutions outside of the secured research environment. The idea is to show that taking advantage of results from machine learning on sensitive health data is possible without privacy violations.

### 5.4 Challenges

During the work on project Jasmine, we have identified the following challenges which should be considered when planning similar projects in the future:

---

<sup>1</sup> <https://www.kanta.fi/en/system-developers/what-are-kanta-services>



1. Data application process from Findata and collecting and pseudonymization of the data can take more than a year. In our case, it took 14 months from submission of the application to acceptance of the application. The current estimate for time required for collection and pseudonymization of the data is 8 months. This creates uncertainty for the allocation of personnel for the work.
2. Data in Kanta on the level of individuals is not available for product development. Individual-level data can only be used for scientific research and the results must be published. For companies, this limits the competitive advantage gained from the research, thus there is a need to carefully consider what type of research is good use of resources.
3. All requested data needs to be justified by the research questions in data application, thus enough time needs to be reserved for defining the proposed research.
4. Data Protection Impact Assessment (DPIA) may be needed in large data application. Thus, expertise and time for making the assessment is required.
5. Use of machine learning based solutions for example in automatic identification of people with medical risks may lead to inequality in offered health services, which could hinder practical use of such solutions
6. Kapseli is a secured research environment, a virtual machine, where the data will be made available. The environment is accessible only to named researchers and does not have an internet connection which sets limitations on practical research work.
7. No individual level health data is allowed to be exported from the Kapseli, thus machine learned results must be checked carefully to prevent accidental inclusion of private data.

## 5.5 Outcome

Results this far from the project Jasmine include:

- Two promising health data driven use cases were identified:
  - Automated medical risk calculation
  - Summarisation of data for medical professionals
- Data application for Kanta data in original XML format has been accepted by Findata, and data pseudonymization is in progress
- Master's thesis work done with an alternative health data set showed that natural language processing methods are promising for example finding out if a person is a smoker based on textual data
- Comparison of existing medical risk calculator against data that should be available in Kanta was done, and resulted in a publication that is in pre-print [25]
  - We found that by improving the availability of a few risk calculator parameters, we can calculate multiple different risk calculators
    - Data identified to be required by part of examined risk calculators, but that is completely missing from Kanta include: data about living habits, habitation, and physical activity and family related data such as family medical history

## 6. Research and ML model development for medical device software

---

### 6.1 Overview

Data and AI driven health applications, such as those used to assist diagnosis or care path selection are categorized as regulated medical software. Companies developing medical software are responsible for MDR/IVDR compliance of the final product. Typically, the development of data-driven medical applications is a long process contributed by several partner organisations. In particular, the development process may involve a scientific research and model development phase carried out by a university or a research institute. The research performer should then take MDR/IVDR requirements into account in order to ensure efficient transition from research to product development. For example, it is important that the process of tuning, validating and testing an ML model is properly documented and traceable, if the model is to be used as part of medical software at a later stage.

In the following subsections, regulatory challenges of healthcare innovation development are analysed through three use cases:

- Machine learning model development using sensitive personal data
- DevOps tools supporting ML based health monitoring solution
- Development of ML based machine vision applications

### 6.2 Machine learning model development using sensitive personal data

#### 6.2.1 General

Machine learning models are extensively used for automatic medical image analysis to support and improve human interpretation. They are increasingly also used to support precision medicine by predicting patient outcomes, identifying patients with elevated risk and suggesting most favourable care pathways and services for the patients. Machine learning models empower decision support applications providing guidance to healthcare professionals and patients [2, 3].

Availability of data is a critical precondition for ML model development [26]. Concerning applications supporting social and health services, sensitive personal data is typically needed. Such use of data may take place in Finland under the Act of Secondary Use of Health and Social Data<sup>1</sup> as described in Section 5. The legislation enables the use of data from health and social services data for register-based studies. Alternatively, the model development may take place in the context of a prospective trial setting where data is collected based on the consent of the data subject.

In both cases, the security requirements for data processing are high. In the case of register-based study settings, data is only available for a fixed time period and in an isolated processing environment<sup>2</sup> without connections to external systems. Such processing environment cannot be directly integrated with standard software development environments, which causes challenges to maintain the link between ML model development and the final software product. The medical software product documentation should include accurate information about the used data sets for ML model training, validation and testing.

---

<sup>1</sup> <https://www.finlex.fi/fi/laki/ajantasa/2019/20190552> (in Finnish)

<sup>2</sup> <https://findata.fi/en/kapseli/regulation-on-secure-operating-environments/>

We developed a concept to support traceability for ML model development in a secure processing environment. We tested the concept in the case of MAITE project [13], which develops AI based models for predicting elderly population's needs for health and social services.

### 6.2.2 Use case: health and social services decision support

The MAITE project emerges from the observation that health and social services expenditure is dominated by service consumption by a small fraction of the population<sup>1</sup>. It is expected that future heavy users of services could be identified based on their current health and social status and service usage history. The objective of the MAITE project is to develop an ML based model and proof-of-concept (PoC) application for predicting future service usage of elderly individuals. In particular, the project aims to identify individuals which use a large number of different services and which would benefit from integrated services. Identification of such target groups would help in service planning and in executing personalized and group-level preventive interventions.

VTT is responsible for the ML model development based on register data of Päijät-Häme Joint Authority for Health and Wellbeing (PHHYKY), a public health and social services provider with catchment area of 200 000 inhabitants in Southern Finland. The Finnish institute of health and welfare (THL) is responsible for coordinating the co-operation of stakeholders and ensuring continuous interaction between developers and end-users.

### 6.2.3 Secure processing environment

The data resources needed by the MAITE project include health and social services encounter data (demographic data, diagnoses, medication, laboratory, operations), social services decisions and assessments of service need and physical functioning. The cohort includes data from 37 000 individuals. Data permit to use the data was issued by PHHYKY. For processing, the data was transferred to the secure processing environment (Kapseli) operated by Findata<sup>2</sup>. The setting for data processing in Kapseli is depicted in Figure 12.

---

<sup>1</sup> <https://www.julkari.fi/handle/10024/80171>

<sup>2</sup> <https://findata.fi/>

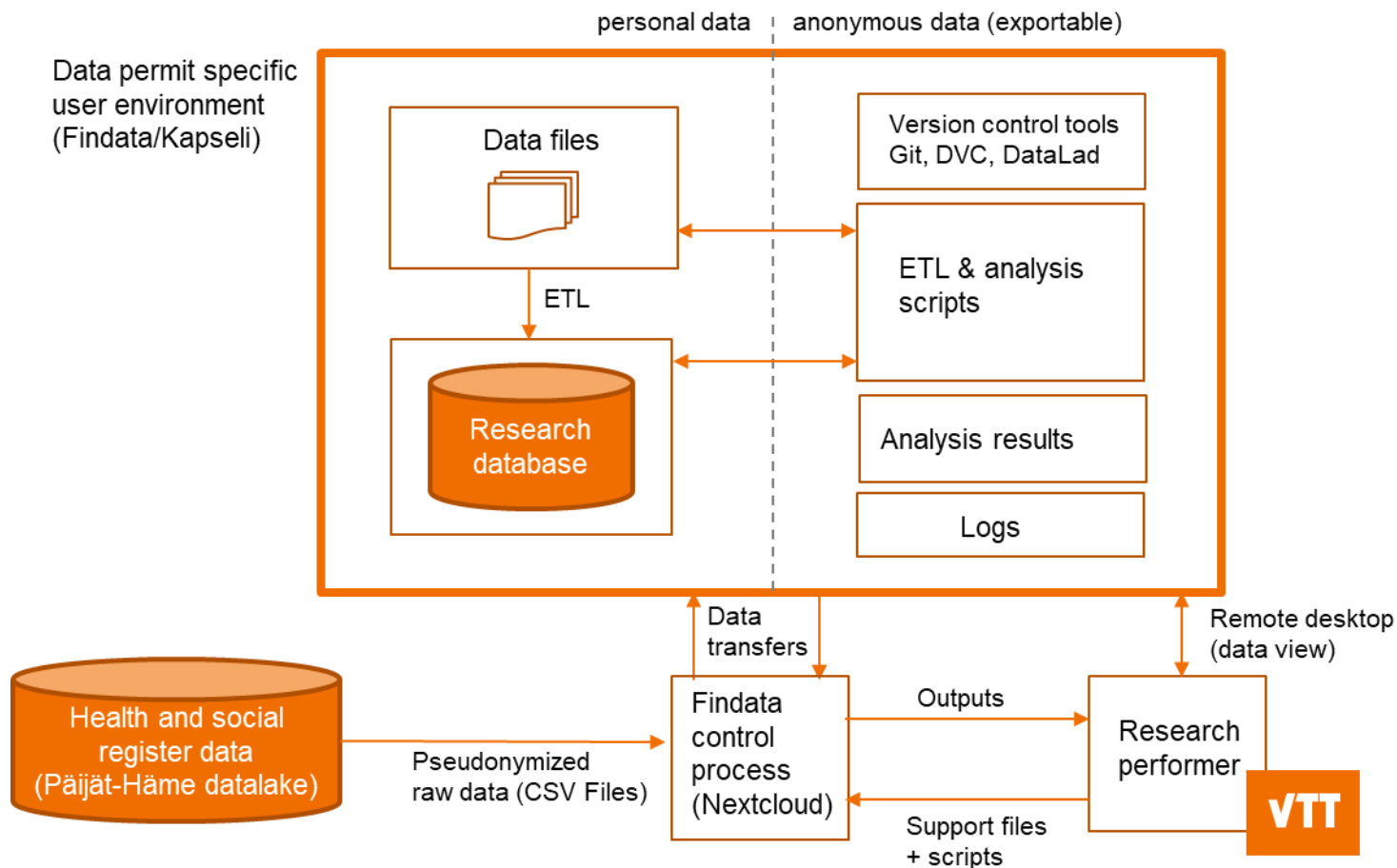


Figure 12. Data processing set up for MAITE project.

The data processing in Kapseli takes place in data permit specific user environments also referred as secure processing environments. The data user (research performer) has a remote desktop access to environment. Most typical data analytics tools are preinstalled in the environment and available for the data user. Also the Git, DVC and DataLad version control tools are available. The remote desktop access provides a data view and keyboard entries, but does not enable data downloads or uploads. All data transfers to and from the environment go through the Findata's control process. In the beginning of the project, pseudonymized data (raw data files) and any needed support files (e.g. clinical vocabularies) are transferred to the environment. In the end and during the project analysis results and user developed scripts need to be transferred out from the environment. Findata's control process aims to ensure that all materials exported from Kapseli are anonymous. Additionally, the data user is responsible for ensuring that materials requested to be exported are anonymous.

#### 6.2.4 ETL and analysis scripts

The ML model development workflow involves two kinds of scripts: *ETL scripts* and *analysis scripts*. Extract transform load (ETL) refers to the process of extracting data from its original source, transforming it to the required form and loading it to a database where it is ready to be used for the data analytics task in question. In the case of MAITE, the first part of data extraction was carried out in the data lake environment of Päijät-Häme. The resulting raw CSV-files were transferred to Kapseli using the NextCloud services of Findata. In Kapseli a number of ETL scripts were executed in sequence. First, the data was extracted from the raw CSV files and transferred to the staging area in the research database (MySQL). Then the data was transformed into suitable form for data analysis and loaded into the main analysis table of the research database. The main analysis table has one row per participant with >300 variables describing the individual's health history and usage of social and health services.

The analysis scripts support the ML model development involving feature extraction, training, validation and testing. Feature extraction includes the operations needed to derive informative and non-redundant variables (“features”) from the original variables as relevant for the ML model. Training refers to the phase where machine learning model versions are iteratively evaluated based on validation data not used for model training. Testing refers to the phase where a final, unbiased evaluation of the model performance is carried out based on data not yet used in training and validation phases. The analysis scripts get input data from the main table of the research database generated in the ETL process.

### 6.2.5 Ensuring traceability

Workflow of ML model development is depicted in Figure 13.

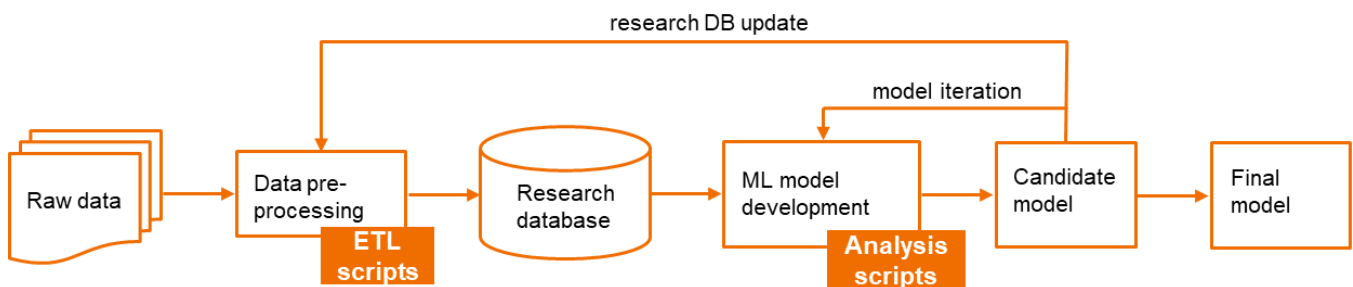


Figure 13. ML model development workflow.

Traceability of the actions carried out during the product development is an important requirement for medical software as discussed in Section 4.6. In the case of ML based applications, traceability requirements do not apply only to software code, but also to the used data sets. The objective is to ensure that all analysis results (e.g. tuned model parameters, weights, coefficients and performance measures) are well-documented and reproducible. The traceability solution shall ensure that for any candidate model, current versions of the executed analysis scripts are saved and can be restored later if needed. Furthermore, the traceability solution shall ensure that complete information is available for recreating the used input data sets.

Traditional version control software (e.g. Git, CVS, Mercurial) are best suited for source code control, but not optimized for data version control. Especially, large data file size causes problems for traditional source control systems. There are existing solutions, such as Data Version Control (DVC)<sup>1</sup> and DataLad<sup>2</sup>, particularly designed for data version management (see sections 6.2.6 and 6.4.3). A common approach in such systems is to use a traditional version control (e.g. Git-based) to manage metadata information (e.g. file location, hash) of data files stored in the local or remote file system.

Existing data version control software can be used to manage different versions of the raw data files used in the workflow of Figure 13. However, there is need for additional methods to ensure traceability of the research database building process as well as the usage of the database data in ML model development. In order to fulfil this need we developed and tested a new Git-based practice.

The practice is based on logging detailed information about the ETL and analysis processes. In the typical workflow (Figure 13) the ETL scripts are executed once in the beginning to create the research database. After this, the research database is used iteratively during a longer period by different versions of the

<sup>1</sup> <https://dvc.org/>

<sup>2</sup> <https://www.datalad.org/>

analysis scripts. In reality the research database may need to be modified or re-created several times during the ML model development activity. Consequently, also the ETL scripts need to be modified and re-executed. This is the case for example, when the research database is updated by new data. Therefore, for each analysis run and candidate model, information for restoring the research database state shall be saved along with the analysis results. An important feature of the solution is that no personal information is logged so that the logs can be downloaded from the secure environment to be archived without privacy risks. Furthermore, structured logging is used to enable easy automatic processing of log data. It is important to note that the research database is not needed to be stored permanently for traceability. It can always be recreated based on the ETL scripts in Git, the logged information and the original data files.

The specific actions carried out by the ETL and analysis scripts to ensure traceability are listed in Table 1. Each execution of the ETL scripts produces a specific read-only directory (identified by a unique ETL\_ID) where the structured log of the ETL execution is stored. Similarly, each execution of the analysis scripts produces a specific read-only directory (identified by a unique ANA\_ID) where structured log of the analysis is stored. Additionally, this directory contains the analysis results.

*Table 1. Actions automatically performed by ETL and analysis scripts to ensure traceability*

| Phase    | Action   | Objective   |
|----------|--|---|
| ETL      | Create unique ETL execution ID (ETL_ID) and store it in the research database.   | Identifies the current execution of the ETL scripts. Storage of the ETL_ID in the database enables the analysis scripts to access it and store it along with the analysis results for traceability.               |
| ETL      | Commit ETL script files to Git. Get CommitID_1. Store CommitID_1 in log.   | Version control of ETL scripts. CommitID_1 enables the current version of ETL scripts to be retrieved afterwards.   |
| ETL      | Create a new database and populate it with pre-processed data. Log information about the ETL process (selected options, raw data files, software environment etc.) | Raw data is converted to a form appropriate for later analysis. Exact information of the ETL process is documented and can be accessed later.   |
| ETL      | Create triggers  | Database triggers are created to insert information about any changes made to the database after the ETL process was terminated.  |
| ETL      | Store log in a read-only directory tagged with the ETL_ID.   | Detailed information about the ETL execution can be accessed afterwards based on the ETL_ID.  |
| Analysis | Create unique analysis execution ID (ANA_ID).  | Identifies the execution of the analysis scripts.   |
| Analysis | Commit analysis script files to Git. Get CommitID_2. Store CommitID_2 in log.  | Version control of analysis scripts. CommitID_2 enables the current version of analysis scripts to be retrieved afterwards.   |
| Analysis | Get ETL_ID from the database and store it in log.  | Based on the stored ETL_ID all information of the ETL process relevant to the current database version can be accessed. The ETL_ID enables this version of the research database to be recreated later if needed. |
| Analysis | Read the history table from database and store information if the database has been changed in log.  | Enables to reveal if changes have been made to the database after ETL scripts were executed.  |
| Analysis | Log information about model development (selected algorithms and used data sets for training/validation/testing etc.)  | Collect detailed documentation of model development and testing process.  |
| Analysis | Store log data and model performance results in read-only directory tagged with ANA_ID.  | Detailed information about the analysis execution can be accessed afterwards based on the ANA_ID.   |

## 6.2.6 Data version control tools

There exists few other solutions for the data version control challenge. Two interesting open source solutions are DVC<sup>1</sup> and DataLad<sup>2</sup>. Both software are solving the same problem, but with different angle. Table 2 underlines the principal similarities and differences of them:

Table 2 Comparison of DataLad and DVC

| Property  | DataLad   | DVC   |
|---|---|---|
| License   | MIT   | Apache 2.0  |
| Target  | All-purpose version control solution for data and code  | Version control system for machine learning   |
| Backend   | Git, Git-annex  | Git   |
| Developed and maintained by                                       | DataLad community   | Company (Iterative <sup>3</sup> ) with supporting community   |
| User interface  | Command line  | Command line, VS code plugin, python library, Studio by Iterative.ai (proprietary software)                                   |
| Collects user statistics  | No  | Yes, possible to opt out  |
| Workflow management   | Special run and rerun commands to record and (re)run commands. Option to define analyses on containers for reproducibility. | Has system for ML code pipeline management with <i>experiments</i> functionality for evaluation and retraining of the models. |
| Supports external version control services like Github or Gitlab. | Yes   | Yes   |
| Requires Git commands in addition to its own commands             | No, Git is run under the hood by DataLad commands.  | Yes, DVC prepares files for Git but basic Git commands are required to manage version control.                                |

Both tools are using hashes to manage large data files with Git. The basic principle is that data is stored in separated location and the associated Git repository contains only information about data but not data itself. DVC has its own implementation for the large data management and DataLad uses Git-annex<sup>4</sup> - a Git extension to manage large files.

DataLad in principle tracks every action ran with the data, especially when analytics are applied within "datalad run" commands. Each action gets its own record at Git history and are thus possible to trace back afterwards. With DVC user must run the Git commands separately to maintain the version control of the task. DVC manages the data related action on its own side when the code and metadata is version controlled at Git by user activity.

<sup>1</sup> <https://dvc.org/>

<sup>2</sup> <https://www.datalad.org/>

<sup>3</sup> <https://iterative.ai/>

<sup>4</sup> <https://git-annex.branchable.com/>



When comparing DataLad to the ETL process described in 6.2.4 it has very similar functionality, but it works with files instead of ETL scripts and database tables. Similarly to the ETL based approach DataLad stores unique identifier for every action under the DataLad managed project. The main difference is that all of the actions are stored in Git history. This makes possible to follow the project activity through Git log for any files processed at the project.

There is a good comparison of DataLad and DVC available at DataLad handbook<sup>1</sup>. DataLad and DVC are fully focused on data stored in files and do not support version control of data stored in a database. Therefore, they cannot provide a direct alternative for managing the research database through the ETL process as described above. They could be used in parallel or in conjunction with the ETL process to manage the raw data files and machine learning models. DVC has some special functionality which may make it more useful for machine learning project but DataLad is simpler to use after the run processes are defined as separated Git commands are not needed.

## 6.3 DevOps tools supporting ML based health monitoring solution

### 6.3.1 General

Remote health monitoring refers to the follow-up of various types of health conditions outside hospital settings, e.g. at home. Remote health monitoring is typically based on a medical device including a sensor, associated control electronics and embedded software responsible for sending the measurement data to a back-end server. Continuous integration and deployment of the embedded software require special arrangements compared to software applications. In particular, connection between the development environment and the device needs to be provided. The software may also include AI/ML models which need to be trained and tested. This adds complexity to the SW development environment and process.

### 6.3.2 Tools

Our main focus has been in tools for requirements and issues management, application building with the CI/CD, and testing. We have chosen the widely used GitLab, Jenkins and Jira systems as reference systems.

#### *GitLab*

Git is an open-source distributed version control management system. It is used by a broad range of different types of projects. The change tracking feature enables monitoring of the progress also for the non-technical project members. GitLab<sup>2</sup> is an extension for Git and it adds tools and features to make it a complete DevOps platform. GitLab is available as a self-managed or as a cloud based version.

In our use case, GitLab was used as a code and data repository for a health monitoring application and the associated ML model. We used the self-managed installation which is distributed under MIT license. When using self-managed version we can freely and without cost make different configurations with unrestricted access. For production use the GitLab SaaS version may be more suitable.

We run GitLab on a Docker container. Instructions for downloading and starting the self-managed GitLab Docker image can be found from the GitLab web site<sup>3</sup>.

---

<sup>1</sup> [https://handbook.datalad.org/en/latest/beyond\\_basics/101-168-dvc.html](https://handbook.datalad.org/en/latest/beyond_basics/101-168-dvc.html)

<sup>2</sup> <https://about.gitlab.com/>

<sup>3</sup> <https://docs.gitlab.com/ee/install/docker.html>

### *Jenkins*

Jenkins<sup>1</sup> is an open-source (MIT license) environment that has a large set of CI/CD tools and features to automate tasks related to building, test, deployment, integration, and release of the software. One of the Jenkins' strengths is that its automation pipeline script is considered to be well structured and easily understandable and hence it is easy to set up. One powerful feature of Jenkins is its large plugin library which enables the use of 3<sup>rd</sup> party tools if needed, e.g., in the testing phase. The use of 3<sup>rd</sup> party plugins is also considered to be a weakness in Jenkins ecosystem, because of the vulnerabilities in plugin's maintenance and further development.

Jenkins is implemented in Java and it can be installed as a standalone Java application on any machine that includes Java Runtime Environment (JRE). Besides Java WAR package, Jenkins is also distributed as native packages, installers, and Docker images. In our experiment we use Java application version of Jenkins run in a Windows 10 PC.

### *Jira*

Jira Software<sup>2</sup> is an issue tracking and project management tool developed by Atlassian. Software teams can use this tool for requirements, issues, and tasks management, as well as bug tracking and workflow approvals. This tool is widely used also among non-technical teams for operations such as change requests and general task management. Jira has also support for Kanban and Scrum agile frameworks.

Jira is not an open-source application. For the cloud based version the licences are available from free basic version (up to 10 users) to large scale premium version which costs USD 1525 per year (up to 10 users). The Jira Software Data Center is also available as a self-managed enterprise version. In this document we use the free trial version of the self-managed Jira Software Data Center. The Docker image of the Jira is freely available<sup>3</sup>. The self-managed version is currently a discontinued product, but our setup can be directly extended to the cloud-based versions.

### 6.3.3 Reference system

We took a remote heart rate monitoring system as a reference. The reference system depicted in Figure 14 encompasses a Windows application (SensorData gateway) connected by Bluetooth to an Arduino Nano 33 BLE Sense microcontroller device to collect heart rate and photoplethysmography (PPG) data. The collected data is used to create a user-specific ML model to determine heart rate from the PPG data. The model is based on convolutional neural network (CNN) algorithm implemented in Python using Tensorflow machine learning tools and libraries. The created ML model is integrated into the software, that is uploaded to the microcontroller based device and used to calculate heart rate from the PPG data in real-time.

---

<sup>1</sup> <https://www.jenkins.io/>

<sup>2</sup> <https://www.atlassian.com/software/jira>

<sup>3</sup> <https://hub.docker.com/r/atlassian/jira-software>

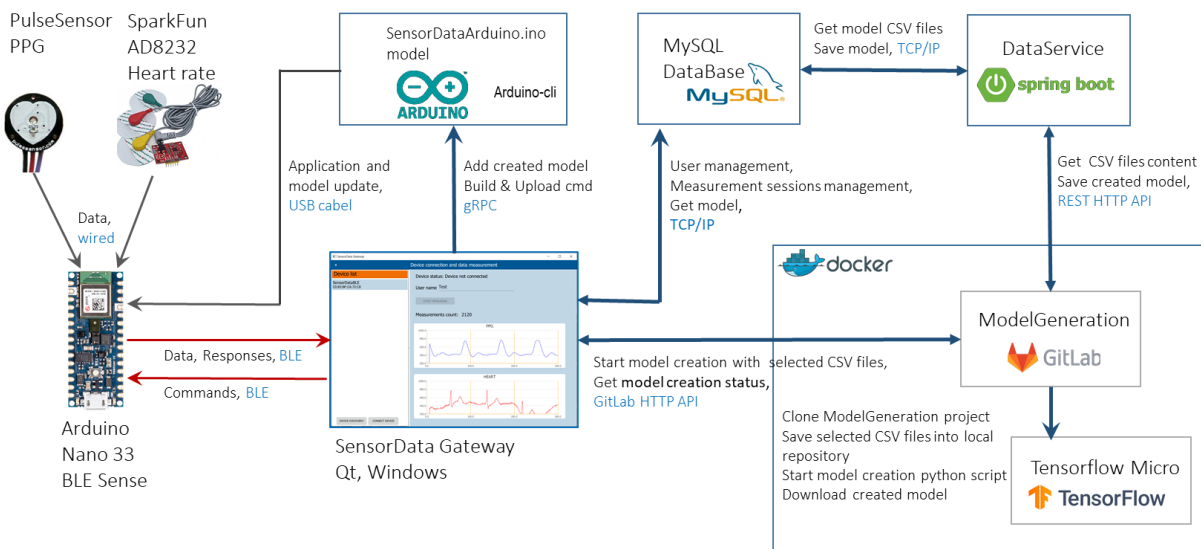


Figure 14. Architecture of the reference system

In this reference system the focus is in collecting sensor data and sending that data to the ML model generation process. The SensorData Gateway uses GitLab HTTP API to start a ML model creation pipeline in GitLab. The ML model creation pipeline script downloads sensor data (csv-files) from MySQL database, executes model creation Python script, and uploads the created ML model into the database. When the ML model is ready, the SensorData Gateway downloads the ML model from the database and saves it locally to be used in the Arduino application development process. The Arduino application development is started by the SensorData Gateway by using Arduino command line interface (Arduino-cli). The Arduino-cli is also used for the deployment of the Arduino application with USB cable from the PC to Arduino device.

The reference development system depends on the SensorData Gateway to start the development and deployment processes. The device application code resides in the local computer, so that changes to the software code are not available to other project members. In addition, changes to the device application code are not automatically compiled, tested and deployed.

### 6.3.4 Integrated system

In order to address the shortcomings of the reference system we have set up an experimental system for continuous integration and deployment (Figure 15). The functionalities are shortly described in the following subsections.

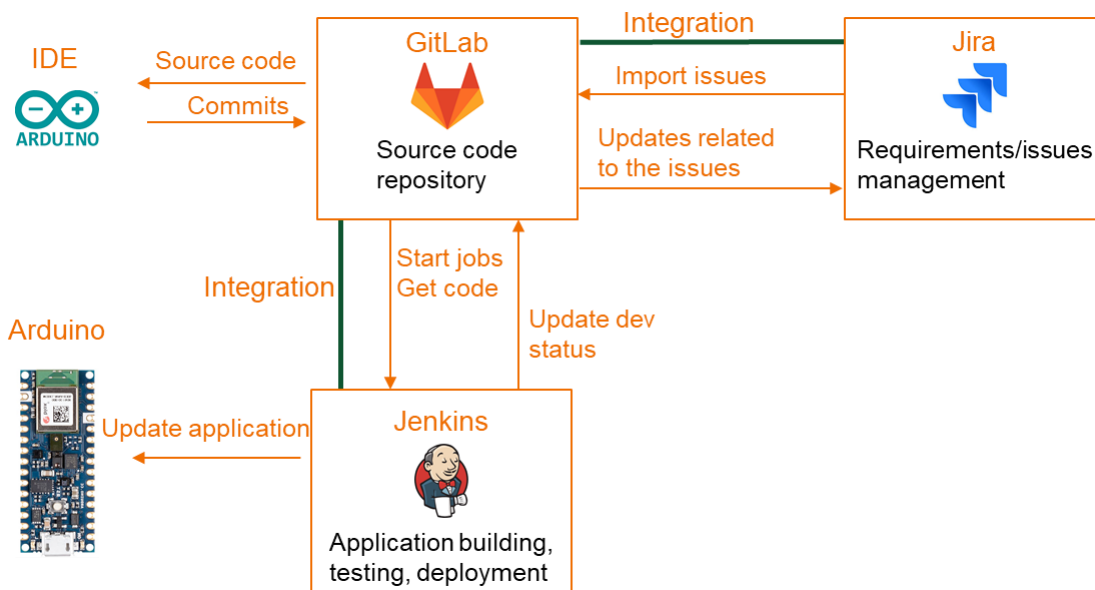


Figure 15. Architecture for continuous integration and deployment.

### 6.3.5 Requirements and issues management

Jira is used for requirements and issues management. In Jira we create an own project for each component of the application. Machine Learning model creation project defines the requirements for the ML model needed by the application. Business logic project takes care of data processing needed to fulfil the application purpose and also functionalities such as database access, user management and external communications. We also keep the compliance requirements from regulatory bodies in a separate project.

Integration between GitLab and Jira enables importing of issues and requirements managed in Jira to be available for developers in GitLab. Integration can take place in two ways: *Basic Jira Integration* or *Jira Development Panel*. These integrations can be used separately, but it is recommended, that both of them are enabled.

Basic Jira Integration can be defined in the GitLab's side either at instance-level<sup>1</sup> (only self-managed GitLab version) by administrator or by a project owner at a specific project level. If the instance-level integration is used, it provides default values for configuring the integrations between GitLab and Jira and it is applied to each of the projects in the GitLab's instance that don't yet have integration configured. The default values can be overwritten in the project's settings. If the integration takes place at the project level, then all the values are set separately for each project. Each GitLab project can also be configured to integrate with all the projects defined in the Jira instance. In that case a GitLab project is not needed to be associated with any particular Jira project.

Jira Development panel integration enables referencing Jira issues in GitLab and it presents commits, branches, and merge requests with links towards GitLab. This integration makes it possible to use *Smart Commits*<sup>2</sup>, i.e., when in the GitLab the Jira issue's ID is put in the commit message or comment, then these are made visible also in Jira's side in the *Development Panel*. Jira's issue transition can also set with a workflow state name with the Jira issue ID.

<sup>1</sup> [https://docs.gitlab.com/ee/user/admin\\_area/settings/project\\_integration\\_management.html#manage-instance-level-default-settings-for-a-project-integration](https://docs.gitlab.com/ee/user/admin_area/settings/project_integration_management.html#manage-instance-level-default-settings-for-a-project-integration)

<sup>2</sup> <https://support.atlassian.com/jira-software-cloud/docs/process-issues-with-smart-commits/>

### 6.3.6 Continuous integration and deployment

Continuous integration and deployment (CI/CD) is based on integration between GitLab and Jenkins. Jenkins uses the GitLab API for retrieving source code files from GitLab to be compiled, built and deployed to the target device (Arduino). After uploading, unit tests are executed. GitLab's push mechanism is used to trigger automatic execution of this process, for example after a Git commit operation. The process also includes the necessary conversions of the test outputs into JUnit format for graphical output in Jenkins UI. In the end of the process the generated build files and test results are stored in GitLab.

## 6.4 Development of ML based machine vision for medical purposes

### 6.4.1 General

Machine vision applications are widely used in the healthcare sector and have a plethora of uses, ranging from improving the patient experience up to guiding or driving clinical procedures. These applications are increasingly relying on latest advancements in machine learning and deep learning algorithms because of the unarguably better performance in terms of accuracy, robustness, processing time, etc [27]–[29]. However, a pre-requisite of these modern techniques is to have large enough datasets that allow researchers and developers to train, validate and test the developed models. This shifts the paradigm from a code-oriented development to a data-driven development, which has a significant impact on the tools used for MDR and IVDR compliance.

The FDA whitepaper on AI/ML in Medical Devices<sup>1</sup> has a clear emphasis on data. Data is part of various steps in the total product lifecycle (TPLC) and has multiple instances (training/validation/testing, live data, re-training data, etc.). It becomes clear that we need specific tools for efficient data handling since the quality of the data can significantly influence an AI/ML system. Based on the QMS standard ISO 13485 (Article 4.2.5) control of all records is required, which implies the versioning and documentation of training, validation, and testing datasets.

In the case of machine vision applications, data is usually recorded in the form of images, and it is organized in folders. Images occupy a large amount of storage space and considering the needs of ML models for large amounts of data, the storage space can easily become a problem. Therefore, efficient organization and tracking of image data is a complex task. Traditional version control tools, such as Git cannot directly handle these huge amounts of data, since they were originally developed for lightweight files that store software source-code.

Therefore, a holistic system capable of capturing the version history of data, source code and analysis outputs is a highly desired feature for any medical device software. By providing a snapshot of all dependencies and a documented history of all changes, an AI/ML system is closer to full compliance with the ISO 13485.

### 6.4.2 Image enhancement using AI for endoscopic imaging

To showcase the benefits and potential challenges of having a fully tracked system, we will present one use case from the FOSDIGUM project<sup>2</sup>, where we research AI-based image enhancement techniques for

---

<sup>1</sup> <https://www.fda.gov/files/medical%20devices/published/US-FDA-Artificial-Intelligence-and-Machine-Learning-Discussion-Paper.pdf>

<sup>2</sup> <https://www.ipt.fraunhofer.de/en/projects/fosdigum.html>

endoscopic imaging. The main goal of this use case is to deliver the best possible image quality to surgeons, helping them in endoscopic procedures. We start by analysing still images, which can be captured by surgeons for documenting purposes. The processing time requirement in this case is not strict and the use-case allow for offline computation. However, the end-goal would be to optimize the algorithm and use it as a live video enhancement, which require real-time computation capability. The research work for this use case has been carried out during VTT's researcher visit to endoscope manufacturer Karl Storz in Germany and Switzerland.

There are various aspects that affect the quality of an image, such as sensor noise, compression artefacts, illumination conditions, motion blur and many others. Because there are multiple algorithms that are tackling these problems independently (or even jointly in some cases), experimenting with all the blocks that form the imaging pipeline can soon become a problem from the data management and tracking point of view. Figure 16 depicts a generic pipeline for image enhancement. In the experimentation phase, the building blocks can also change their position in the pipeline and exhaustive testing is needed in order to find the winning combination.

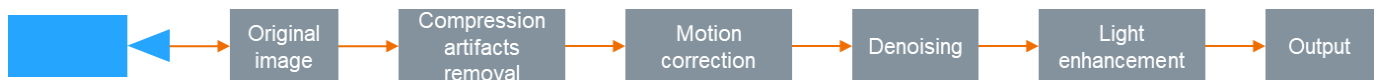


Figure 16 General pipeline for image enhancement.

These blocks can be internally implemented using different ML models; hence it is necessary to keep track of not only the original data, but also of the training/validation/testing data needed for each model. According to FDA's risk classification of medical devices, we have classified endoscopic imaging as being part of the treat or diagnose class, meaning that there is a high level of risk that we need to address. This further encourages us to adopt a robust and complete tracking system for our image enhancement pipeline. While adding more and more intelligence to the imaging pipeline, it needs to be ensured that the output image that the surgeon analyses is identical to the real-world situation.

To address data management and tracking, we have turned our attention to data version control software like DVC and DataLad. We considered the DVC to be most relevant for this use case.

#### 6.4.3 DVC and Gitlab in supporting image data control

DVC is an open-source version control system for ML projects which enables data versioning, ML workflow automation and experiment management through a seamless integration with Git. An intuitive representation for DVC is "Git for data". While DVC is technically not a version control system, it creates the needed artefacts to enable data version control via Git (see also Section 6.2.6).

In Figure 17, we present the architecture we used to demonstrate the advantages of DVC. We use two computers to simulate a team of researchers/developers working on the same project. The goal of the system is to have the computers synchronized with regard to the source code and data, and have a data version control mechanism.

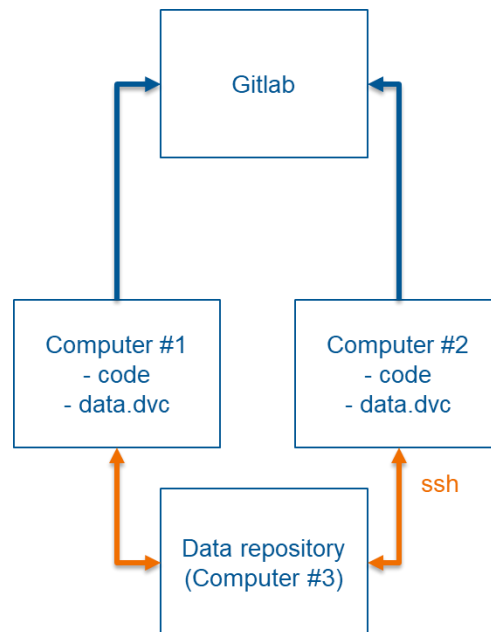


Figure 17 System architecture for image data control with DVC.

DVC offers the option for maintaining data in a safe remote data repository (Google Drive, Amazon S3, etc.). In our case, we have set up a third computer as a data repository, which we can access via SSH. A Gitlab instance server works as the synchronizer between the two main workstations. DVC versions the data by generating a `.dvc` file, containing the corresponding hashes for all data files that are tracked. When the `.dvc` file is created, the actual data folder will be added to the `.gitignore` file, and only the `.dvc` file will be pushed on the remote Gitlab repository. Therefore, the Gitlab repository will remain light, but data will be correctly tracked.

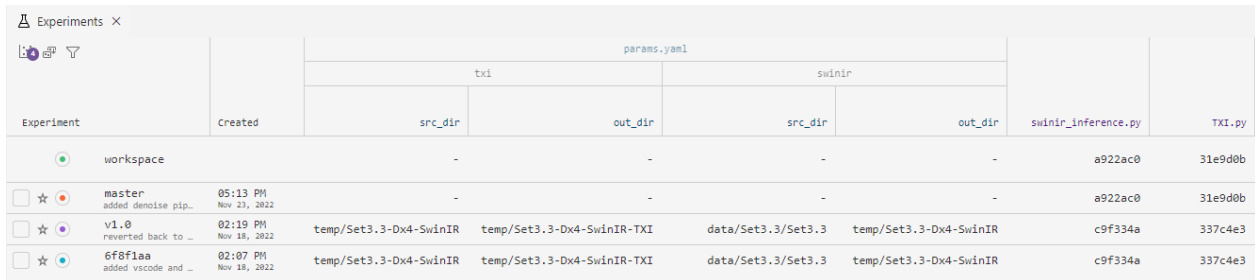
Let us analyse the following scenario: On computer #1, new data is added. The following protocol will be followed:

- Run `dvc status` to check what has changed.
- Run `dvc add` to track changes in the new dataset. This will modify the contents of `data.dvc` file. This way, also Git will know that there has been a change.
- Add in a new `git commit` the `data.dvc` file and push it to the remote repository. Push with `dvc push` the newly added data to the data repository. This step is very important, so that other users not only download the new hashes from the Gitlab instance but can also pull the new data from the data repository.
- On computer #2, we will first pull the latest version from Gitlab.
- By running `dvc checkout`, DVC will notify that indeed the data has changed, but we do not physically have the new data. So, to fix this we pull the new data with `dvc pull`. Then, we run `dvc checkout` to switch to the new version of the dataset.

This is a brief exemplification of data versioning. The same protocol can be followed for any operations on the data: modification, addition, removal. Moreover, each software version will have its corresponding data version, meaning that we can replicate the system on any new workstation. Another great feature of DVC is that we can go through older commits and checkout the exact data version that was used in that commit, which not only ensures reproducibility, but also removes the need to create redundant copy of the datasets with small modifications.

Previously, we have presented in Figure 16 a generic pipeline for image enhancement and pointed out that in practice, one needs to analyse a variety of pipelines, and in particular analyse and keep track of the intermediate outputs. DVC offers a feature for creating data pipelines, which captures how data

changes in the pipeline and helps in recreating and sharing the workflows. Training, validation and testing datasets can be captured as data dependencies and included as stages in the pipelines. This means that any change in a dataset involved in the pipeline would trigger a re-run of the pipeline. In addition, the researchers can have a visual representation of all the dependencies and how the data flows from one module to another, while keeping track of any parameter change.



| Experiment                      | Created                  | params.yaml            |                            |                    |                        | swinir_inference.py | TXI.py  |
|---------------------------------|--------------------------|------------------------|----------------------------|--------------------|------------------------|---------------------|---------|
|                                 |                          | txi                    |                            | swinir             |                        |                     |         |
|                                 |                          | src_dir                | out_dir                    | src_dir            | out_dir                |                     |         |
| workspace                       |                          | -                      | -                          | -                  | -                      | a922ac0             | 31e9d0b |
| master<br>added denoise pip...  | 05:13 PM<br>Nov 23, 2022 | -                      | -                          | -                  | -                      | a922ac0             | 31e9d0b |
| v1.0<br>reverted back to ...    | 02:19 PM<br>Nov 18, 2022 | temp/Set3.3-Dx4-SwinIR | temp/Set3.3-Dx4-SwinIR-TXI | data/Set3.3/Set3.3 | temp/Set3.3-Dx4-SwinIR | c9f334a             | 337c4e3 |
| 6f8f1aa<br>added vscode and ... | 02:07 PM<br>Nov 18, 2022 | temp/Set3.3-Dx4-SwinIR | temp/Set3.3-Dx4-SwinIR-TXI | data/Set3.3/Set3.3 | temp/Set3.3-Dx4-SwinIR | c9f334a             | 337c4e3 |

Figure 18. Experiments view created with DVC.

Figure 18 presents the *Experiments view* created with the DVC extension in Visual Studio Code. The view shows one row per each experiment carried out during the ML model development process. The *Experiments view* enables access to all values in the *params.yaml*, including for example the hyperparameters of each experiment. Along with the value of the parameters, the commit ID of each source code file is recorded (TXI.py, swinir\_inference.py). Moreover, metrics (e.g. AUC, accuracy, etc.) can also be recorded for each experiment.



## 7. Conclusions

---

The objective of the AHMED project was to present best practices for applying agile software and DevOps approaches in medical device software development. Different perspectives were represented in the heterogeneous project consortium. The partners included (1) companies providing medical software based products for customers (Atostek, Bittium, Mylab, Solita), (2) healthcare service providers developing software primarily for inhouse use (Terveystalo), (3) companies providing software development tools (Taipuva Consulting) and (4) organisations carrying out research targeting at future medical software (VTT) or at new software development processes and tools (University of Helsinki). The AHMED project enabled partners to exchange experiences and knowledge related to regulated medical software and the specific challenges of AI/ML and data usage. The co-operation was smooth and open. All partners considered that the sharing of experiences and knowledge was highly beneficial and helped them in adopting new processes and tools to improve their MDR compliance.

This report summarizes the experiences and best practices of the project partners in MDD to MDR transition, AI/ML compliant RegOps lifecycle development, software development tools, personal health data usage and ML model development in the research context.

Transition from MDD to MDR requires significant efforts and resources from software providers. According to the new MDR regulation a large number of health related applications now fall in the MDR class IIa category and require certification by a notified body. A remarkable difficulty is caused by the fact that notified bodies lack resources and cannot respond to the high demand of certification services. Because of this, significant delays in bringing products to markets may result. Bittium, Terveystalo and Mylab upgraded their quality systems to match MDR requirements during the AHMED project. The experiences show that it takes at least one year to transit from a MDD based quality system to MDR compliance. Co-operation and exchange of experiences with other companies was considered very beneficial in this process. Also, early interaction with the notified body was considered important especially for understanding the specific requirements related to the use of AI/ML. The continuous documentation paradigm was seen essential in fulfilling the regulatory requirements efficiently.

For a company, RegOps framework can be seen as a way to meet requirements of large global customer companies and to create development partnerships. In the RegOps framework, AI/ML development practices deserve specific attention. AI/ML applications are problematic from the regulatory perspective as they may need to be updated when new data comes available. Unfortunately, the MDR does not explicitly address AI/ML based medical software. Furthermore, there is also lack of guidance documentation and harmonized standards addressing the use of AI/ML in regulated applications. Solita's approach has been to further develop their existing RegOps model based on the general idea of the AI system lifecycle model presented in the draft version of ISO/IEC DIS 5338:2022 standard. The model addresses specific AI-related aspects relevant from the regulatory perspective, especially highlighting that the AI-enabled technology needs to be justified and proven safe, clinically effective, and secure. The RegOps model pays attention to managing the inherent complexity and change dynamics of AI-based systems. The model also outlines the importance of data management practices and their alignment with regulatory requirements.

Software tools are available for all relevant parts of the RegOps cycle, including: project planning and management, documentation, software development, integration, testing, deployment and operations/monitoring. Traceability and continuous documentation are especially important features in the context of regulated medical software development. Tools should support the workflow ensuring, for example, that all regulatory documents are created. Typically several tools need to be integrated to support the RegOps cycle. A workflow automation demonstration involving integration of Polarion application lifecycle management tool with GitHub version control was carried out by Taipuva and University of Helsinki. Integration of ALM and version control tools enable, for example, syncing documentation with software changes. The integration also allowed the analysis of different change management scenarios. The analysis showed that integration and concept mapping between the tools needs to be carefully designed to support smooth co-operation of the different user groups. Specific tools for ML development were also considered. The Model Cards metadata tool (Google Research) was tested and observed to be

useful in supporting both engineering and regulatory needs. Also a reusable approach for supporting tool validation was developed and demonstrated in the framework of the AHMED project.

Usage of personal data is typically involved in medical AI/ML based applications. Access to personal data is highly regulated and subject to several preconditions depending on the type and amount of data requested. Atostek has applied the permission to use pseudonymized patient record data from the Kanta national electronic health record system. The identified use cases for data are: automated medical risk calculation and summarisation of data for medical professionals. The data permit has been currently granted by Findata and preparation of data for use is ongoing. Valuable experience of the data application process has already been gained and included in this report. For example, it is important to understand that individual-level data is only available for scientific research purpose, which implies the need to justify the study by presenting valid research questions and to publish the results of the study. The individual-level data can only be processed in a closed processing environment (Kapseli) with no direct connections outside. This reduces privacy risks, but also complicates the data analysis activity. All this, together with the long application process needs to be considered when planning data exploitation under the Finnish legislation on the secondary use health and social data.

Research institutes and universities are frequently carrying out research activities, which may later lead to components of regulated software. It is highly important to carry out the early research in a way compatible with regulation. VTT carried out three use cases highlighting the usage of customized and off-the-shelf tools in ensuring regulation compliant research project execution, e.g. collection and management of traceability data. The first use case demonstrates a solution for maintaining traceability when processing personal data in a closed secure processing environment (Findata/Kapseli). The presented approach is based on controlled ETL process and logging of anonymous data during research database generation and data analysis. The second use case involves a setup of tools for supporting AI/ML based health monitoring application development. The setup demonstrates requirements and issues management based on Jira/GitLab integration and continuous integration and deployment based on GitLab/Jenkins integration. The third use case demonstrates image data management in a medical image enhancement pipeline. The demonstration uses DVC for traceability of the AI/ML model development process. The approach enables linking each experiment with the used data sets, software version and obtained model performance results.

## References

---

- [1] M. Gokarna and R. Singh, “DevOps: A Historical Review and Future Works,” *Proc. - IEEE 2021 Int. Conf. Comput. Commun. Intell. Syst. ICCIS 2021*, pp. 366–371, Feb. 2021, doi: 10.1109/ICCIS51004.2021.9397235.
- [2] T. Melvin and M. Torre, “New medical device regulations: the regulator’s view,” *EFORT open Rev.*, vol. 4, no. 6, pp. 351–356, Jun. 2019, doi: 10.1302/2058-5241.4.180061.
- [3] L. Keutzer and U. S. H. Simonsson, “Medical Device Apps: An Introduction to Regulatory Affairs for Developers,” *JMIR mHealth uHealth*, vol. 8, no. 6, Jun. 2020, doi: 10.2196/17567.
- [4] J. Feng *et al.*, “Discussion on ‘Approval policies for modifications to machine learning-based software as a medical device: A study of bio-creep’ by Jean Feng, Scott Emerson, and Noah Simon,” *Biometrics*, vol. 77, no. 1, pp. 45–48, Mar. 2021, doi: 10.1111/BIOM.13381.
- [5] U. J. Muehlematter, P. Daniore, and K. N. Vokinger, “Approval of artificial intelligence and machine learning-based medical devices in the USA and Europe (2015–20): a comparative analysis,” *Lancet Digit. Heal.*, vol. 3, no. 3, pp. e195–e203, Mar. 2021, doi: 10.1016/S2589-7500(20)30292-2.
- [6] S. V. G. Subrahmanya *et al.*, “The role of data science in healthcare advancements: applications, benefits, and future prospects,” *Irish Journal of Medical Science*. 2021, doi: 10.1007/s11845-021-02730-z.
- [7] WHO, “Ethics and Governance of Artificial Intelligence for Health: WHO guidance,” *World Heal. Organ.*, pp. 1–148, 2021, Accessed: Feb. 25, 2022. [Online]. Available: <http://apps.who.int/bookorders>.
- [8] T. Lysaght, H. Y. Lim, V. Xafis, and K. Y. Ngiam, “AI-Assisted Decision-making in Healthcare,” *Asian Bioeth. Rev. 2019 113*, vol. 11, no. 3, pp. 299–314, Sep. 2019, doi: 10.1007/S41649-019-00096-0.
- [9] B. Babic, S. Gerke, T. Evgeniou, and I. Glenn Cohen, “Algorithms on regulatory lockdown in medicine,” *Science (80-. )*, vol. 366, no. 6470, pp. 1202–1204, Dec. 2019, doi: 10.1126/SCIENCE.AAY9547/ASSET/A4D80998-FB54-4CED-8458-D86F7F05BBB5/ASSETS/GRAPHIC/366\_1202\_F1.JPEG.
- [10] S. Gilbert, M. Fenech, M. Hirsch, S. Upadhyay, A. Biasiucci, and J. Starlinger, “Algorithm Change Protocols in the Regulation of Adaptive Machine Learning–Based Medical Devices,” *J. Med. Internet Res.*, vol. 23, no. 10, Oct. 2021, doi: 10.2196/30545.
- [11] S. Gilbert, M. Fenech, A. Idris, and E. Türk, “Periodic Manual Algorithm Updates and Generalizability: A Developer’s Response. Comment on ‘Evaluation of Four Artificial Intelligence–Assisted Self-Diagnosis Apps on Three Diagnoses: Two-Year Follow-Up Study,’” *J Med Internet Res 2021;23(6)e26514* <https://www.jmir.org/2021/6/e26514>, vol. 23, no. 6, p. e26514, Jun. 2021, doi: 10.2196/26514.
- [12] T. Granlund, · V. Stirbu, and · T. Mikkonen, “Towards Regulatory-Compliant MLOps: Oravizio’s Journey from a Machine Learning Experiment to a Deployed Certified Medical Product,” *SN Comput. Sci. 2021 25*, vol. 2, no. 5, pp. 1–14, Jun. 2021, doi: 10.1007/S42979-021-00726-1.
- [13] J. Lähteenmäki, J. Pajula, and E. Antikainen, “Development of medical applications based on AI models and register data – regulatory considerations,” in *Proceedings of the 18th Scandinavian Conference on Health Informatics*, 2022, pp. 141–146, [Online]. Available: <https://ecp.ep.liu.se/index.php/shi>.

- [14] “EUR-Lex - 52021PC0206 - EN - EUR-Lex.” <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206> (accessed Feb. 28, 2022).
- [15] V. Jormanainen and J. Reponen, “CAF and CAMM analyses on the first 10 years of national Kanta services in Finland,” *Finnish J. eHealth eWelfare*, vol. 12, no. 4, pp. 302–315–302–315, Dec. 2020, doi: 10.23996/FJHW.98548.
- [16] T. Granlund, T. Mikkonen, and V. Stirbu, “On Medical Device Software CE Compliance and Conformity Assessment,” *Proc. - 2020 IEEE Int. Conf. Softw. Archit. Companion, ICSCA-C 2020*, pp. 185–191, Mar. 2020, doi: 10.1109/ICSCA-C50368.2020.00040.
- [17] T. Granlund, J. Vedenpaa, V. Stirbu, and T. Mikkonen, “On Medical Device Cybersecurity Compliance in EU,” *Proc. - 2021 IEEE/ACM 3rd Int. Work. Softw. Eng. Heal. SEH 2021*, pp. 20–23, Jun. 2021, doi: 10.1109/SEH52539.2021.00011.
- [18] V. Stirbu and T. Mikkonen, “Introducing Traceability in GitHub for Medical Software Development,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 13126 LNCS, pp. 152–164, Oct. 2021, doi: 10.48550/arxiv.2110.13034.
- [19] V. Stirbu, M. Raatikainen, J. Rontynen, V. Sokolov, T. Lehtonen, and T. Mikkonen, “Toward Multiconcern Software Development With Everything as Code,” *IEEE Softw.*, vol. 39, no. 04, pp. 27–33, Jul. 2022, doi: 10.1109/MS.2022.3167481.
- [20] T. Granlund, V. Stirbu, T. Mikkonen, C. Pautasso, and O. Zimmermann, “Medical Software Needs Calm Compliance,” *IEEE Softw.*, vol. 39, no. 1, pp. 19–28, 2022, doi: 10.1109/MS.2021.3117292.
- [21] M. K. Higashi *et al.*, “Association Between CYP2C9 Genetic Variants and Anticoagulation-Related Outcomes During Warfarin Therapy,” *JAMA*, vol. 287, no. 13, p. 1690, Apr. 2002, doi: 10.1001/jama.287.13.1690.
- [22] V. Stirbu, T. Granlund, and T. Mikkonen, “Continuous Design Control for Machine Learning in Certified Medical Systems,” *Softw. Qual. J.*, Sep. 2022, doi: 10.1007/s11219-022-09601-5.
- [23] T. Granlund, A. Kopponen, V. Stirbu, L. Myllyaho, and T. Mikkonen, “MLOps Challenges in Multi-Organization Setup: Experiences from Two Real-World Cases,” *2021 IEEE/ACM 1st Work. AI Eng. - Softw. Eng. AI*, pp. 82–88, May 2021, doi: 10.1109/WAIN52551.2021.00019.
- [24] V. Stirbu, T. Granlund, J. Helen, and T. Mikkonen, “Extending SOUP to ML Models When Designing Certified Medical Systems,” *Proc. - 2021 IEEE/ACM 3rd Int. Work. Softw. Eng. Heal. SEH 2021*, pp. 32–35, Mar. 2021, doi: 10.48550/arxiv.2103.09510.
- [25] V. Männikkö, A. Kopponen, T. Mikkonen, K. Förger, and M. Torhola, “Strengthening Well-Being Based on National Health Records: Citizen Digital Twins with Personal Risk Assessment,” *SSRN Electron. J.*, Aug. 2022, doi: 10.2139/SSRN.4181368.
- [26] J. Lähteenmäki *et al.*, “Integrating data from multiple Finnish biobanks and national health-care registers for retrospective studies: Practical experiences,” *Scand. J. Public Health*, p. 140349482110044, Apr. 2021, doi: 10.1177/14034948211004421.
- [27] H. E. Kim *et al.*, “Changes in cancer detection and false-positive recall in mammography using artificial intelligence: a retrospective, multireader study,” *Lancet Digit. Heal.*, vol. 2, no. 3, pp. e138–e148, Mar. 2020, doi: 10.1016/S2589-7500(20)30003-0.
- [28] A. Rodriguez-Ruiz *et al.*, “Stand-Alone Artificial Intelligence for Breast Cancer Detection in Mammography: Comparison With 101 Radiologists,” *JNCI J. Natl. Cancer Inst.*, vol. 111, no. 9, pp. 916–922, Sep. 2019, doi: 10.1093/JNCI/DJY222.

- [29] C. Hassan *et al.*, "Performance of artificial intelligence in colonoscopy for adenoma and polyp detection: a systematic review and meta-analysis," *Gastrointest. Endosc.*, vol. 93, no. 1, pp. 77-85.e6, Jan. 2021, doi: 10.1016/J.GIE.2020.06.059.

## Annex 1: Terms and Acronyms

| Term / acronym | Description  |
|----------------|--|
| AI             | Artificial intelligence  |
| AI/ML          | Artificial intelligence or machine learning. In many cases, the categorization between AI and ML is not relevant and it is more appropriate to refer to the two technologies together.               |
| ALM            | Application lifecycle management   |
| API            | Application programming interface  |
| AUC            | Area under curve. In machine learning a measure of the ability of a classifier to distinguish between classes.   |
| CD             | Continuous deployment  |
| CE mark        | CE mark on a product means that the manufacturer or importer affirms the good's conformity with European health, safety, and environmental protection standards.                                     |
| CI             | Continuous integration   |
| CSV            | Comma-separated value. A text file format used for storing tabular data.   |
| DevOps         | Set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. |
| DL             | Deep learning  |
| DVC            | Data Version Control. Version control software for machine learning models, data sets and intermediate files.  |
| ETL            | Extract Transform Load. A process where data is extracted from its original source, transformed to a suitable form and loaded into an output data container where it is ready to be used.            |
| FDA            | Food and Drug Administration   |
| GDPR           | General Data Protection Regulation   |
| IVD            | In vitro diagnostic medical device   |
| IVDR           | In Vitro Diagnostic Medical Devices Regulation. Regulation (2017/746) by EU on placing in vitro diagnostic medical devices on the market.  |
| JUnit          | XML based format for storing software test data  |
| MD             | Medical device   |
| MDD            | Medical Device Directive. Former regulation (93/42/EEC) by EU concerning medical devices (Currently replaced by MDR and IVDR)  |
| MDCG           | Medical Device Coordination Group  |

|                         |  |
|-------------------------|--|
| MDR                     | Medical Device Regulation. Regulation (2017/745) by EU on placing medical devices on the market.   |
| Medical Device Software | Software that is intended to be used, alone or in combination, for a purpose as specified in the MDR or IVDR. In this report also referred shortly as “Medical software”.  |
| ML                      | Machine learning   |
| MLOps                   | Machine Learning Operations. Set of practices that aims to deploy and maintain AI/ML models in production reliably and efficiently.  |
| Notified Body           | A notified body is an organisation designated by an EU country to assess the conformity of certain products before being placed on the market. These bodies carry out tasks related to conformity assessment procedures set out in the applicable legislation, when a third party is required. |
| QMS                     | Quality management system  |
| RegOps                  | Regulatory Operations. Combination of cultural philosophies, practices, and tools that increases an organization's ability to ensure compliance of applications and services against regulatory standards at high velocity.  |
| REST                    | Representational state transfer  |
| SaaS                    | Software as a service  |
| SAFe                    | Scaled Agile Framework   |
| SaMD                    | Software as a Medical Device   |
| SOP                     | Standard Operating Procedure   |
| SSH                     | Secure shell protocol  |
| TPLC                    | Total product lifecycle  |
| V&V                     | Verification & Validation. The process of checking that a software system meets specifications and requirements so that it fulfils its intended purpose.   |