Cloud Native Transformation for Pharmaceutical Company, ElixirCure

Table of Contents

Table of Figures 3
Abbreviations 4
1. Organization Insights 5
1.1. Organization Structure 6
1.2. Existing System Study 6
1.2.1. Drug Development Process 6
1.2.2. Existing System 7
1.2.3. Bottlenecks and Improvement Areas 9
2. Cloud Native Computing to Address the Pain Points of Existing System 10
2.1. Recommendations for Technology Modernization 10
2.1.1. Microservices 10
2.1.2. Containerization using Docker 11
2.1.3. Continuous Integration, Continuous Delivery and Continuous Deployment 12
2.1.4. SAP Business Technology Platform for Data Analytics 14
2.1.5. Context-aware and Multifactor Authentication for Secure Collaboration 14
2.1.6. Proposed system for ElixirCure 15
2.2. Project Management using Agile and DevOps Approach 16
2.2.1. Best Practices for Effective Sprint Planning 16
2.2.2. Key factors for Efficient Daily Stand-up Meetings and Retrospectives 16
2.2.3. Moving from Agile towards DevOps 17
2.3. Benefits of CNC 17
3. Action Plan for ElixirCure 18
4. Discussion 19
5. Conclusion 19
6. References 20

Table of Figures

Figure 1. Drug development process and supporting third-party software applications (AiCure,no date;Argus,no date;Eichler,2018; Genomics,no date;GenomOncology,no date;Mak and Pichika,2019;Margreth and Neufeld,no date;SAP,no date a;Sparta Systems,2022;VelQuest,no date;Waters,no date) 7
Figure 2. Examples of in-house software applications used in the drug development process. 8
Figure 3. Examples of databases used for the drug development process (Wu et al.,2020). .8
Figure 4. High-Level Architecture of Existing System (Boudlal,Serrhini and Tahiri,2022) 8
Figure 5. Bottlenecks and Improvement Areas 9
Figure 6.Main Cloud Service Models 10
Figure 7. Microservices Implementation Challenges and Mitigation Actions (Bucchiarone et al.,2020; Kanjilal,2020) 11
Figure 8. Benefits of Microservices (Bucchiarone et al.,2020; Thones,2015;Viggiato et al.,2018) 11
Figure 9. Benefits of Containerization(Abdollahi et al.,2018; Bentaleb et al.,2021;Microsoft,no date) 11
Figure 10. Comparison of Docker Swarm and Kubernetes (Aqua,2022;Hira,2022;Rosen,2022) 12
Figure 11. CICD Practices 13
Figure 12. Continuous Deployment Tools in Existing System (IBM, no date) 13
Figure 13. Benefits of CICD (Arachchi and Perera,2018) 13
Figure 14. Risks involved and Mitigation actions for CICD (Ganguly,2022; Khalid,2022) 14
Figure 15. Proposed System for ElixirCure 15
Figure 16. Examples for Microservices in the Proposed System 15
Figure 17. Key Differences between Agile, CICD and DevOps (Steven,2021) 17
Figure 18. Action Plan for Implementation of Proposed System 18

Abbreviations

ADMET	Absorption, Distribution, Metabolism, Excretion, and Toxicity
BTP	Business Technology Platform
CaaS	Container as a Service
CI/CD	Continuous Integration and Continuouss Delivery
CNC	Cloud Native Computing
DevOps	Development Operations
DevSecOps	Development Security Operations
ERP	Enterprise Resource Planning
FDA	Food and Drug Administration
GCP	Google Cloud Platform
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
MBT	Model-Based Testing
MFA	Multi-Factor Authentication
MVP	Minimum Viable Product
PaaS	Platform as a Service
QA	Quality Assurance
R&D	Research and Development
SaaS	Software as a Service
SDLC	Software Development Life Cycle

1. Organization Insights

ElixirCure is a medium-sized UK pharmaceutical company that develops medicines for various diseases, especially cancer. Although the company had a strong presence in the market, recently it struggles to deliver high-quality medicines to patients faster than its competitors, which resulted in lower revenue (Newstex,2022). Furthermore, the lengthy process of new drug development contributes to greater costs while customers demand new effective and cheaper medicines (Raja and Sambandan,2015). According to the root cause analysis, the difficulty lies in securely collaborating between different clinical trial centres and the R&D departments, which could significantly improve the drug development process (Newstex,2022). Additionally, the research outcomes are further delayed by remaining compliant with changing regulations and obtaining approvals from quality and regulatory departments (ibid).

ElixirCure's CEO has sensed the need to seek a more sophisticated strategy to speed up the drug development process with the help of software applications. He has allocated a good budget for the new strategy and hired an external consultant to recommend the most effective solution for the company. Through this assessment, ElixirCure's owner attempts to find the answers to the following questions:

- How to streamline the existing complex and long process of the software development lifecycle(SDLC) effectively to deliver high-quality drugs in less time with reduced costs?
- How to collaborate securely with partner companies and operation centres spread across different countries seamlessly?
- Provide a high-level plan for the implementation of the proposed system.
- What potential risks are involved in the proposed solution, and how can they be mitigated?

The answers to the above questions will be covered in the upcoming sections. For an effective solution, it is crucial to understand the current organisation structure and existing system architecture to identify the bottlenecks and improvement areas.

1.1. Organization Structure

ElixirCure has different operation centres across Europe and has partnerships with various companies within and outside Europe. Partnerships have benefited ElixirCure in several ways, such as support in diagnosis, sharing the financial risk and offering royalties and payments to partners when a breakthrough drug discovery occurs and the IP rights are held by the company (Raja and Sambandan,2015; Reinhardt, Oliveira and Ring,2020). The R&D department in Ireland uses different software applications to support the drug development process.

The company has a dedicated IT team including Developers, QA and SAP teams for building, testing, deploying, and maintaining software applications. As part of adopting Industry-4.0, the company introduced Agile methodology in the SDLC, facilitating better communication between team members (Reinhardt, Oliveira and Ring,2020). However, the team is unable to utilize its full potential since it's poorly implemented. Further analysis revealed that obtaining approvals and inputs from other teams takes significantly longer time than completing the tasks. The next section will discuss the existing system in detail.

1.2. Existing System Study

It's important to have a high-level understanding of the sequential and feedback-driven process of the new drug development and how software can assist in different phases before diving into the existing system.

1.2.1. Drug Development Process

As illustrated in Fig.1, the first phase of drug development is target validation which studies the underlying cause of the disease under consideration and identifies the relevant clinical compounds for drug intervention (Gill et al.,2016). The second phase, compound screening involves testing compounds for enzyme interaction and when a compound is active towards the given target then it is classified as a 'hit' (Mak and Pichika,2019). The lead compound that has the potential for drug development is identified in the lead identification phase (ibid). A lead compound is further modified during the lead optimisation process in order to enhance its biological response while reducing its toxicity and adverse effects (ibid).

The preclinical development examines the effect of the developed compound on animals (Mak and Pichika,2019). Clinical research commences once the respective authorities approve the drug is safe to test on humans and it may also lead to drug repurposing (ibid). Upon receiving the approvals for all four phases of clinical research, the drug is manufactured for clinical purposes and patients are monitored regularly through different applications including wearable devices to improve the drug further based on the feedback (ibid). Pharmacovigilance tracks the FDA approvals of medications and their long-term and short-term adverse effects of medicines on a variety of patients, primarily resulting from ADMET characteristics (Mak and Pichika,2019; Wu et al.,2020).

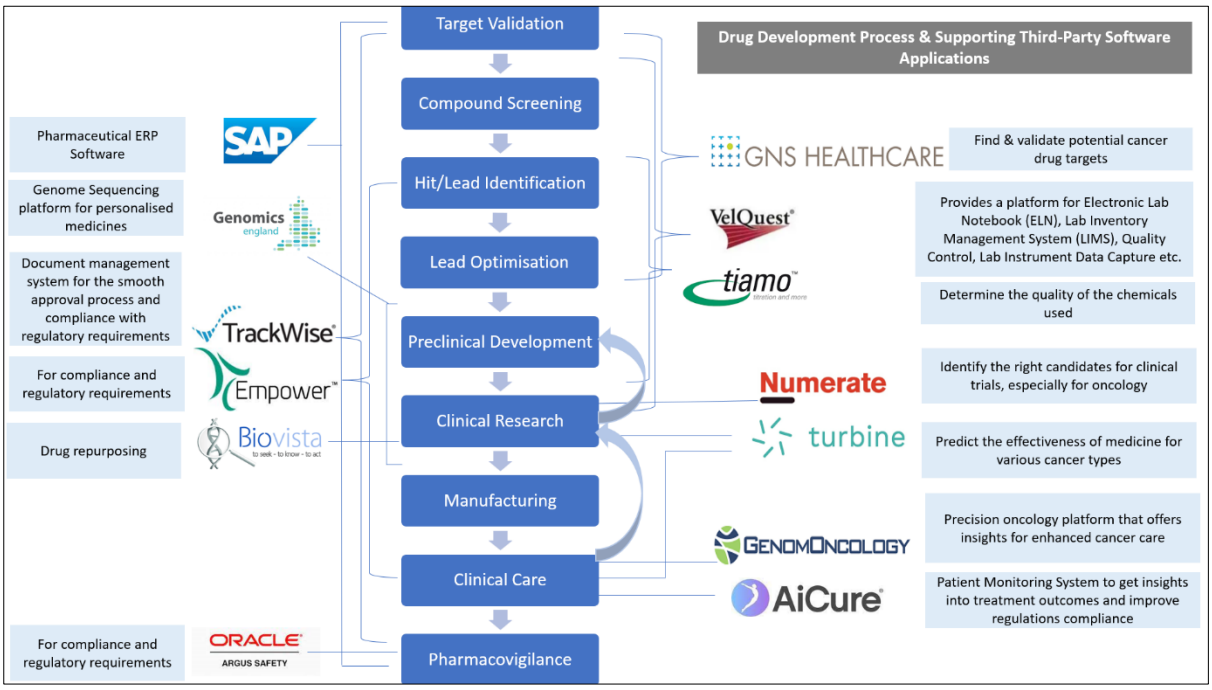

Figure 1. Drug development process and supporting third-party software applications (AiCure,no date;Argus,no date;Eichler,2018; Genomics,no date;GenomOncology,no date;Mak and Pichika,2019;Margreth and Neufeld,no date;SAP,no date a;Sparta Systems,2022;VelQuest,no date;Waters,no date)

1.2.2. Existing System

The existing system leverages both in-house and third-party software applications throughout the drug development process. Fig.1 illustrates different third-party software applications, Fig.2 depicts examples of in-house software applications and Fig.3 shows examples of databases used in different phases of drug development.

Software Application	Functionality
De-Tox	Determine and predicts the toxicity of new compounds formed during drug development process (Mak and Pichika,2019).
External DB Connector	Connects the other software applications to external databases such as ADME, DrugBank etc. through API calls.
Patient Management System	Platform to manage the patient medical and health information across different operation centres.
Reporting Tool	Generates different types of reports for approvals, regulatory compliance and auditing.
SAP Connector	Provide an API interface to access data from SAP for R&D and reporting purposes.
Clinic Trials Management System	Platform to manage the outcomes of clinical trials from different operation centres.
Data Migration Tool	Migrates data from the legacy systems such as SQL Server to Cloud Database Services by ensuring data privacy and security.
Data Analytics Tool	Perform data analytics on the migrated data to get insights into the R&D process.

Figure 2. Examples of in-house software applications used in the drug development process.

Database Name	Description	Software Used	Reference Link
ADME	Third-party paid database for studying drug interactions and ADME.		https://www.fujitsu.com/global/solutions/business-technology/tc/sol/admedatabase/
Liceptor	Third-party free database containing 2D structures, related molecular descriptors and bioactivity data.		http://www.evolvus.com/data.html
DSSTox	Third-party free database containing chemical structure information with existing toxicity data for supporting better predictive toxicology.		https://www.epa.gov/chemical-research/distributed-structure-searchable-toxicity-dsstox-database
DrugBank	Third-party free database to store drug and related target information.		https://www.drugbank.ca/
MetaADEDB	Third-party free database to store the adverse effects of the drugs.		http://lmmd.ecust.edu.cn/online_services/metaadedb/
RegulatoryDB	In-house database for storing regulatory requirements.	Microsoft SQL Server	
ClinicalTrialDB	In-house database for storing data generated during clinical trials.	Microsoft SQL Server	
PatientDB	In-house database for storing patient data.	Microsoft SQL Server	
StaffDB	In-house database for storing staff details	Microsoft SQL Server	
LookupDB	In-house database for software application lookup data	MySQL	

Figure 3. Examples of databases used for the drug development process (Wu et al.,2020).

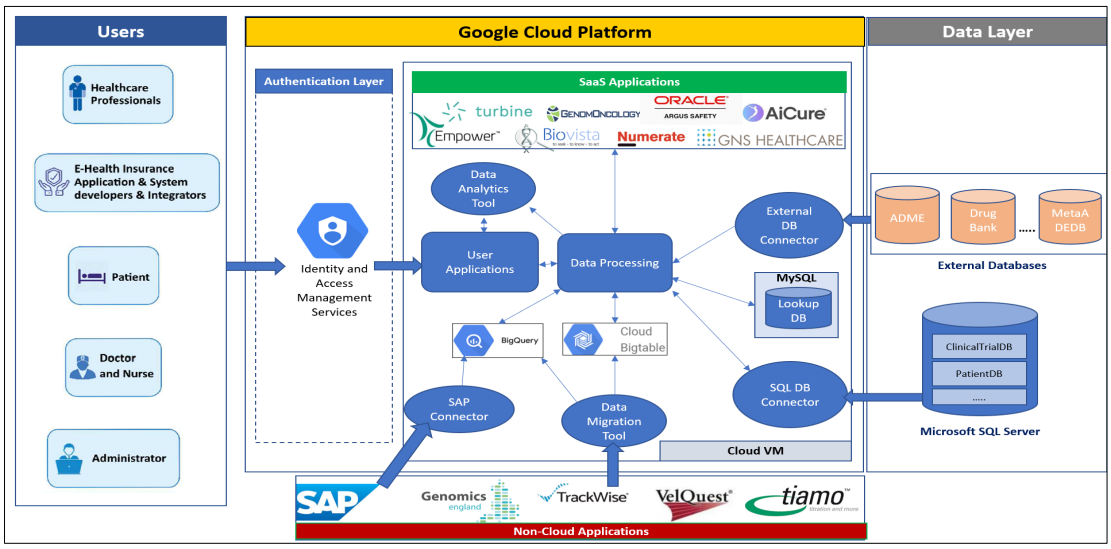

Figure 4. High-Level Architecture of Existing System (Boudlal,Serrhini and Tahiri,2022)

As part of embracing Industry-4.0, most applications are already deployed in a hybrid model (a combination of public and private cloud deployments) in the GCP. Additionally, a data migration project is underway which migrates data from traditional applications such as TrackWise to cloud databases including BigQuery and BigTable. However, most data still reside in the SAP ERP system and Microsoft SQL Server.

1.2.3. Bottlenecks and Improvement Areas

Fig.5 lists the bottlenecks and improvement areas of the existing system that are of high priority. As illustrated in Fig.5, currently the project management process follows a few agile anti-patterns that are delaying the MVP and final product which could add value to the business (Milne,2022). Furthermore, the delay in receiving data from non-cloud applications such as the SAP ERP system and Microsoft SQL Server could accelerate the pace of the drug development process. Additionally, all software applications in the cloud are not broken down into microservices which could help in containerization and poor orchestration is degrading the performance of the existing system. Although the existing system uses an authentication layer for role-based access, it is not leveraging MFA and context-awareness access which enables secure collaboration across different operation centres and partner companies.

Category	Bottlenecks and Improvement Areas	Priority
Project Management Process	Identified the following Agile anti-patterns in the existing system: • Lack of proper sprint planning led to unclear requirements, missed deadlines, quality issues and over workload as tasks were not properly estimated and broken into stories and epics (Milne,2022; Pedersen,2022). • Daily stand-up meetings taking longer time than necessary, reducing team productivity (Milne,2022). • Sprint Retrospectives are not conducted effectively, which would have helped the team to identify pain points and improve efficiency (Pedersen,2022).	High
Non-cloud application Dependency	There is a significant delay in getting data from SAP ERP system and Microsoft SQL Server which could provide valuable insights to speed-up the R&D process.	High
Containerization and Orchestration	The in-house software applications deployed in the cloud is not properly broken down into microservices which can be containerized and poorly orchestrated which is also affecting the overall performance of the existing system.	High
Collaboration and Authentication	Although the existing system has the authentication layer, it doesn't fully utilize its features such as Multi Factor Authentication (MFA) and context-awareness access to users which enables secure collaboration across operation centres and partner companies.	High

Figure 5. Bottlenecks and Improvement Areas

Having identified the pain points of the existing system, the next section will focus on the recommendation of new changes using a cloud-native computing approach to accelerate the drug development process by addressing the identified issues.

2. Cloud Native Computing to Address the Pain Points of Existing System

Before recommending the new changes, it is good to have an understanding of Cloud Native Computing(CNC). The CNC approach enables the business to develop and deliver scalable, resilient, manageable and noticeable applications that can accelerate business velocity and growth by leveraging containers, microservices and infrastructure that are optimised for the cloud computing model along with adopting an agile culture and organizing teams around these principles (CNCF,2022;Beda,2019). Fig.6 depicts the main service models offered by CNC.

The next sections will cover how CNC can address the pain points of the existing system.

Service	Description	Citation
Software-as-a-Service (SaaS)	Enables users to use cloud-based applications over the internet at a minimal upfront cost and pay-as-you-go pricing model, ensuring security and availability without managing infrastructure for it.	(Microsoft, no date b)
Platform-as-a-Service (PaaS)	Offers the resources including infrastructure, storage and networking to build, deploy and manage the full lifecycle of services in the cloud by avoiding the costs and overhead of managing software licenses.	(Azure, no date a)
Containers-as-a-Service (CaaS)	Allows users to manage containers through OS virtualization and configuration for better customization and control.	(Buchanan, no date)
Infrastructure-as-a-Service (IaaS)	Enables users to reserve and provision remote computational resources from their hosting provider without managing the hardware themselves.	(Buchanan, no date)

Figure 6.Main Cloud Service Models

2.1. Recommendations for Technology Modernization

2.1.1. Microservices

Microservices are the smallest single-purpose application that can be built, tested, deployed, and scaled independently (Thones,2015). In the existing system, software components are not decomposed into microservices. As illustrated in Fig.8, the implementation of microservices offers several benefits to the overall system. However, '*microservitization*' can be often challenging due several reasons and it can be mitigated by adopting best practices as depicted in Fig.7 (Bucchiarone et al.,2020,p.10).

Category	Challenges Involved	Mitigation Actions
Design	• Lack of design patterns and practice. • Difficult to identify the right level of granularity. • Vulnerable to attacks since microservices open to APIs that are remotely accessible.	• Let-it-crash models, Circuit-breaker pattern or Bulkhead can be used as design patterns. • Performing technical readiness assessment and trade-off analysis will help to identify the appropriate level of granularity. Trade-off analysis should include key factors such as business feature isolation, network communication expense, and distribution complexity. • Each request to microservice should be authenticated using proper access-control mechanisms.
Development	• Selecting appropriate communication methods such as message queue or API calls between microservices while maintaining low coupling. • Data persistency issues • Testing becomes complex as the business logic is partitioned and distributed over independent services.	• The communication method compatibility between microservice and its consumers should be ensured before selection. • Data persistency issue can be resolved using distributed database systems. • Resilience testing or reusable acceptance testing are recommended.
Operational	• Complexity in managing microservices. • Tracking microservices errors can be challenging since its difficult to determine where the instance is running in a distributed system.	• Complexity increases with the number of microservices and its crucial to assess the trade-off in decomposing the software components into microservices. • Aggregation of logs in a persistent storage in a structured format with relevant details including correlation ID, service name, time, method name can help in better error tracking.

Figure 7. Microservices Implementation Challenges and Mitigation Actions (Bucchiarone et al.,2020; Kanjilal,2020)

Benefits of Microservices	
Agility	A part of the application can be built, tested and deployed independently and quickly.
Scalability	Easy to scale up and down based on demand. Allows replication of specific services rather than entire system.
Reliability	Each service execute independently and designed to be tolerant to the failure of individual service.
Flexibility	Offers the flexibility to select appropriate tools for developing each service, rather than bounded to a single tool or language.
Maintainability	Easy to maintain and update individual service without impacting the entire software application.

Figure 8. Benefits of Microservices (Bucchiarone et al.,2020; Thones,2015;Viggiato et al.,2018)

2.1.2. Containerization using Docker

Through containerization, microservices are packaged with their dependencies and runtime into a binary, enabling them portable and deployable across different environments consistently and quickly without preconfiguring (Microsoft,no date). Fig.9 illustrates the benefits of containerization. Container management is the major challenge in containerization, but it can be mitigated by using proper orchestration tools as discussed in the following paragraphs.

Benefits of Containerization	
Agility	Lightweight, start up more quickly than virtual machines.
Scalability	Easily scalable based on demand and efficient resource utilization.
Reliability	Quickly restart after failure recovery and upgrades.
Portability	Easily deployable across different environments without pre-configuration costs.
Maintainability	Maintainable without impacting the entire software application and manageable using orchestration tools.

Figure 9. Benefits of Containerization(Abdollahi et al.*,2018; Bentaleb* et al.*,2021;Microsoft,no date)*

Docker is a PaaS platform to manage containers easily (Campbell,no date). However, Docker can only manage containers efficiently at a certain scale without orchestration tools(ibid). Container orchestration tools are responsible for managing and maintaining containers based on resource availability and demand (Docker,2023). Fig.10 depicts the comparison between Docker Swarm and Kubernetes the commonly used container orchestration tools (ibid).

Docker Swarm is recommended for ElixirCure due to medium workloads and applications are not too complex. The service is also PaaS, lightweight, easy to set up, and has an internal load balancer (Hira,2022). Furthermore, Docker Swarm chose the optimum node for container deployment and utilize resources efficiently (ibid). Although Kubernetes is more powerful than Docker Swarm, its installation and configuration require a high level of expertise, which may result in high costs and time (Aqua,2023).

Features	Docker Swarm	Kubernetes
Installation and Configuration	Easy with limited customization.	Complex and customizable.
Suitable for	Few workloads and simple applications.	Huge workflows and complex applications.
Load Balancing	Internal load balancer.	Doesn't support automatic load balancing but it's easy to integrate with third-party load-balancing tools.
Monitoring	No built-in monitoring mechanisms, but it supports monitoring using third-party tools.	Supports built-in monitoring mechanisms and integration with third-party tools.
Scaling	Quick autoscaling	Supports horizontal autoscaling, vertical scaling requires configuration.
High Availability	Offers availability controls and containers can be easily duplicated.	Self-healing, intelligent scheduling and high availability through replication.
Security	Network-level security including TLS and security certificate rotation between nodes in fixed intervals.	Enterprise-level security controls including authentication, pod security policies, secrets management and network policies.

Figure 10. Comparison of Docker Swarm and Kubernetes (Aqua,2022;Hira,2022;Rosen,2022)

2.1.3. Continuous Integration, Continuous Delivery and Continuous Deployment

Continuous Integration, Continuous Delivery and Continuous Deployment(CICD) model automates software builds and deployments, enabling agile teams to deliver software frequently (Arachchi and Perera,2018). Fig.11 depicts the CICD practices,

Fig.13 shows the benefits of this approach and Fig.14 illustrates the risks involved in this approach and the mitigation actions for them.

CICD Practices	Description
Continuous Integration	A software development practice that involves integrating team members' work on a regular basis and automating testing to identify and fix bugs quickly to enhance software quality (Arachchi and Perera,2018).
Continuous Delivery	Allows rapid introduction of all kinds of changes including feature additions, configuration changes and defect fixes in a safe and sustainable way, which results in enhanced productivity, reliable releases, customer satisfaction, faster time to market, and better software product quality (Arachchi and Perera,2018).
Continuous Deployment	Automates the release of code changes into the production environment by running predefined tests, and once those tests are passed, the updates are directly released to software end-users (IBM, no date)

Figure 11. CICD Practices

Tool Type	Description	Tools in Use
Version Control	Improves the visibility of project code changes, promotes team collaboration, and facilitates continuous integration by tracking revisions to a project's assets.	Git
Code Review	Tests the source code to improve software integrity by identifying coding errors and bugs, allowing developers to fix them before deploying updates.	GitLab
Continuous Integration	Minimizes development roadblocks and enables collaboration among multiple developers in a project	Jenkins
Configuration Management	Ensure that hardware and software maintain a consistent state, including proper configuration and automation.	Ansible
Application Release Automation	Allows continuous deployment automation and relies on process orchestration tools to ensure developers follow all the required steps before pushing updates into production, in addition to configuration management processes to ensure all project deployment environments are provisioned properly for high performance.	Jenkins
Infrastructure Monitoring	Helps to analyse application performance to determine the impact of changes made.	DataDog

Figure 12. Continuous Deployment Tools in Existing System (IBM, no date)

Fig.12 illustrates the continuous deployment tools used in the existing system. Docker Swarm works well with the existing tools used for achieving CICD. Docker integration in DataDog allows seamless monitoring of container performance (DataDog,2023). Jenkins offers Docker Swarm plug-ins for continuous integration and release automation(Jenkinsci,2021). Ansible can be integrated with Docker Swarm for better configuration management(Patterson,2017).

Benefits of CICD	
Agility	Faster releases of the software products to the end-users.
Reliability	Offers improved software quality by fixing defects quickly.
Maintainability	Easy to maintain configuration changes, new feature additions and defect fixes. Offers code stability using version control.

Figure 13. Benefits of CICD (Arachchi and Perera,2018)

Category	Risks Involved	Mitigation Actions
Reliability	Software quality can be negatively affected by flaws in automated testing.	• Use the right tools and implement them correctly. • Use model-based testing (MBT) for more accurate and faster testing.
Performance	Improper implementation of the CICD pipeline may degrade software performance, stability and scalability.	Automated testing systems or reliable tools like Selenium to identify performance issues by load testing and performance testing.
Compatibility	Auto-updating certain software in the CICD cycle may cause compatibility issues with other software in the pipeline, resulting in the entire pipeline failing.	Auto-updates of software should be disabled to prevent unauthorised updates.
Security	CICD pipeline are vulnerable and are more prone to cyberattacks.	• Use proper authentication and authorisation mechanisms. • Effective system monitoring will help to detect the threats in the initial stage and defend the pipeline. • Limit the amount of PII data in the CICD pipeline. • Remove the vulnerable code using a code analysis tool such as SonarQube.

Figure 14. Risks involved and Mitigation actions for CICD (Ganguly,2022; Khalid,2022)

2.1.4. SAP Business Technology Platform for Data Analytics

SAP ERP system is being used for master data management in the existing system, and an internal SAP Connector application is being used to extract data from SAP to the cloud for data analytics during the drug development process. Data access is delayed significantly for various reasons, including approvals from SAP and file format changes. SAP Business Technology Platform(BTP) is a SaaS platform that automates data workflow from SAP to the cloud in real-time with a drag-and-drop interface and offers meaningful insights on SAP data (SAP,no date b). BTP can reduce data access delays from SAP and accelerate the drug development process through data analytics. Hence, it's recommended to replace SAP Connector with SAP BTP.

2.1.5. Context-aware and Multifactor Authentication for Secure Collaboration

A context-aware access approach allows cloud resources to be accessed securely based on the context of the user, including the user's identity, location, IP address, date, and time (Sullivan,2020). Context-aware access offers greater security than traditional security methods, which do not consider user context for authentication. GCP Cloud IAM is a SaaS platform that provides context-aware identity and access management service for controlling which users can access GCP resources (ibid). A multifactor authentication (MFA) process uses passwords and access codes to authenticate users(Google,no date). Since the existing system is not using MFA or context-aware authentication, enabling these features in the authentication layer could enhance collaboration with global partners and ElixirCure.

2.1.6. Proposed system for ElixirCure

As discussed in the previous section, Fig.15 illustrated the high-level architecture of the proposed system for ElixirCure. Docker and Docker Swarm have been added for containerization. Components are decomposed into microservices and a few examples are listed in Fig.16. SAP Connector is replaced with SAP BTP and the authentication layer is updated with context-aware and multifactor authentication. The data from external databases and MS SQL Server is slowly migrated to cloud storage services for better performance.

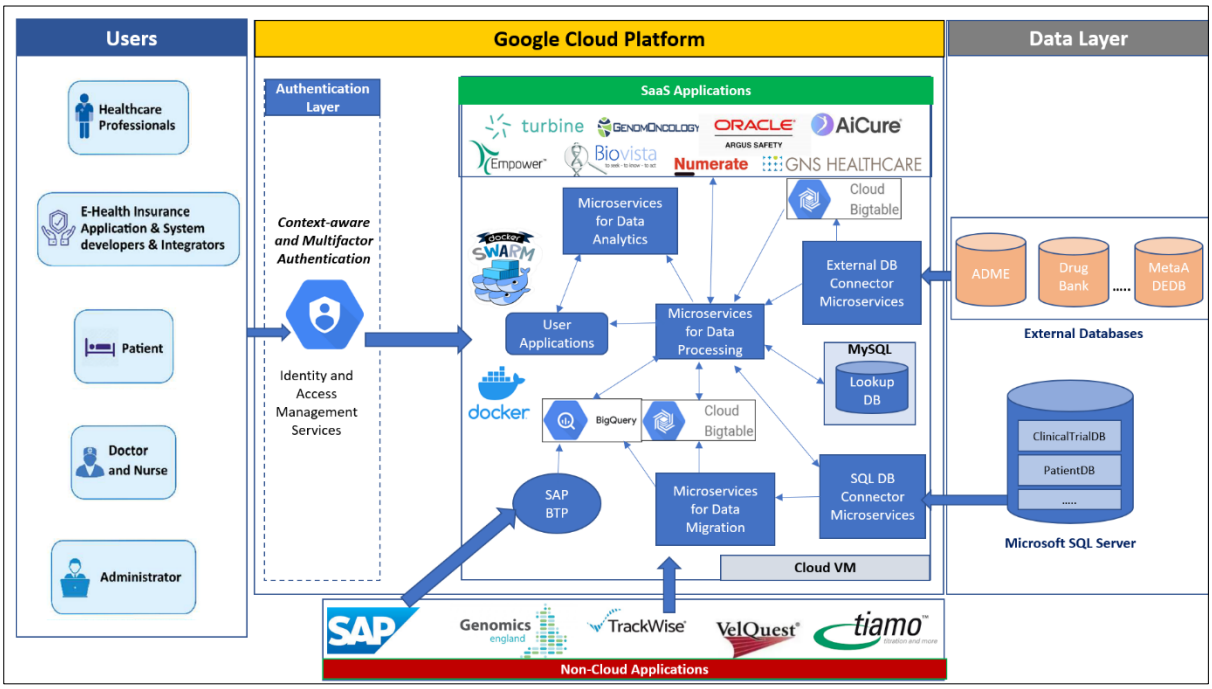

Figure 15. Proposed System for ElixirCure

Microservice Name	Functionality
TrackWise Data Extractor	Extracts data from TrackWise
Data Transformer	Transforms the data into target format
Data Validator	Validates the transformed data
Data Loader	Loads data into the target system
DB Authenticator	Authenticates DB connections
DB Query Executor	Execute database queries

Figure 16. Examples for Microservices in the Proposed System

2.2. Project Management using Agile and DevOps Approach

Despite adopting the Agile approach, ElixirCure's IT team still has room for improvement as discussed previously. The following sections recommend steps to overcome the identified problems to exploit the full potential of this approach.

2.2.1. Best Practices for Effective Sprint Planning

Poor planning resulted in a lot of reworks, accumulating technical debt, and delaying software releases (Milne,2022). For a software project to succeed, sprint planning is critically important, it is often challenging as it requires the proper estimation of complexity, time, business value and user stories that should be included in each sprint (Boschetti et al.,2014).

For better planning, firstly, the product owner should document the requirements, get them approved by relevant stakeholders, and communicate them clearly to the agile team to reduce rework and ensure the timely delivery of the right software product (Milne,2022). Secondly, user stories should be included in a sprint based on their priority, business value, complexity, and dependencies on other stories to produce deliverables that meet customer expectations (Boschetti *et al.*,2014). Thirdly, the product owner should facilitate communication between stakeholders and the team frequently to avoid unnecessary work (Pedersen,2022). Finally, leverage collaboration tools such as JIRA or Trello to keep track of sprint progress and maintain transparency of tasks that need to be achieved (Milne,2022).

2.2.2. Key factors for Efficient Daily Stand-up Meetings and Retrospectives

Ideal sprint daily stand-up meeting lasts for 15 minutes or less and its main objective is to track the progress and identify blockers of members in an agile team (AXELOS,2015). However, in the existing system, stand-up meetings last an hour discussing the blockers, reducing team productivity. Collaboration tools such as Slack or MS Teams can be used to discuss the blockers and find solutions from the team instead of meetings(Milne,2022). Furthermore, the number of active members during stand-up meetings should be limited, ideally not exceeding nine (ibid).

Sprint retrospectives should be conducted after every sprint without fail, to identify ways to improve the efficiency and quality of the team (Pedersen,2022). The scrum

master should ensure that every team member could express their suggestions and take them into consideration (ibid). Since the current agile team is following a two-week sprint, ideally, 1.5 hours should be allocated for sprint retrospectives (Adobe,2022).

2.2.3. Moving from Agile towards DevOps

Agile practices bridge the gap between customers and developers to ensure that software meets business requirements (Hemon *et al.*,2019). However, there is a gap between the operations (Ops) team, responsible for software releases and the development (Dev) team (ibid). As a result, end-users may experience delays in receiving their software (ibid). Testing automation is the key to bridging this gap by enabling efficient collaboration and integration between Dev and Ops teams (ibid). DevOps combines practices, automated tools, and a team-oriented mindset to integrate software development, quality assurance, and operations for faster, more efficient, and high-quality software development aligned with business goals by leveraging Agile and CICD approaches (Gitential,2023). Fig.17 illustrates the key differences between Agile, CICD and DevOps.

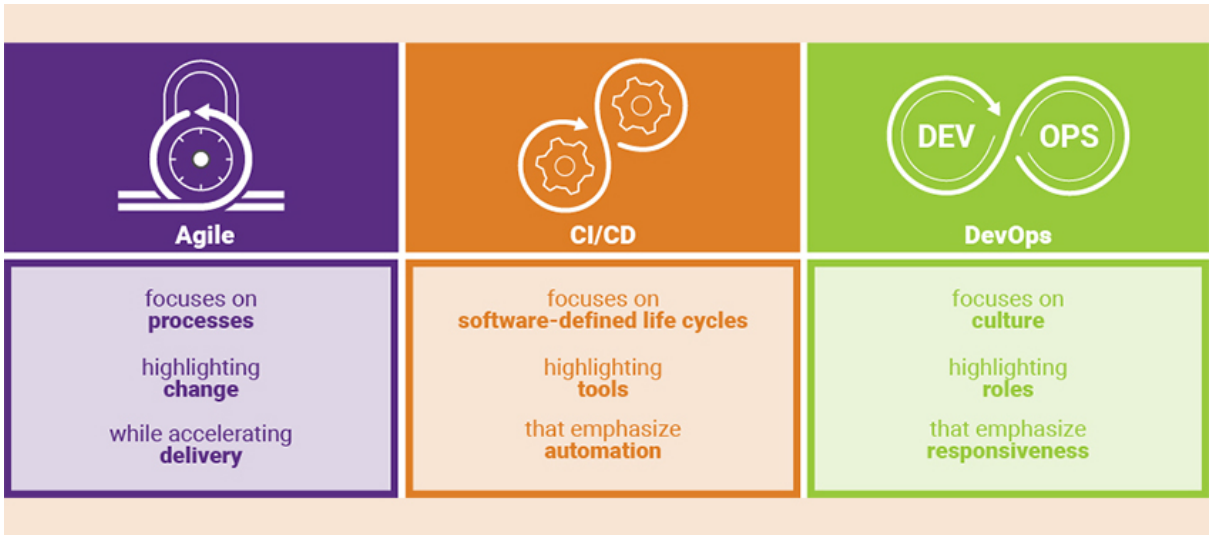

Figure 17. Key Differences between Agile, CICD and DevOps (Steven,2021)

2.3. Benefits of CNC

During the drug development process, ElixirCure deals with a huge amount of heterogeneous and sensitive data captured from a wide variety of sources including laboratory equipment, wearable devices, and sensors, which need to be integrated effectively to derive meaningful insights from it (Ranchal et al.,2020;Boudlal,Serrhini

and Tahiri,2022). As discussed in the previous sections, CNC is capable of catering for these business requirements as well as ensuring security, availability, and failure recovery (Ranchal et al.,2020).

Although the drug development process is following a Waterfall approach, adopting the CNC approach for the development and delivery of supporting software can speed up the process to a certain extent. This will help the company to launch new effective medicines more quickly and at reduced costs as per the customer's expectations. Furthermore, the company can utilize the patent period effectively to generate more revenue before the patent expires (Raja and Sambandan,2015).

3. Action Plan for ElixirCure

The high-level implementation plan of the proposed system is illustrated in Fig.18. Firstly, the IT team should be trained on the CNC approach and the benefits of following it. The management should ensure that agile is practised in the right way and avoid anti-patterns. Thirdly, the technical team should enable proper authentication and authorisation for effective collaboration. Fourthly, technical analysis should be carried out before implementation including determining the level of granularity to implement microservices. Once the analysis is completed, the suggested technical changes should be implemented including microservices and containerization. Sixthly, IT managers should take action to reduce non-cloud dependencies. Seventhly, the CICD pipeline should be optimised for better performance. The team would eventually move to DevOps culture by adopting best practices. The complete implementation of the action plan might take from 1 year to 3 years.

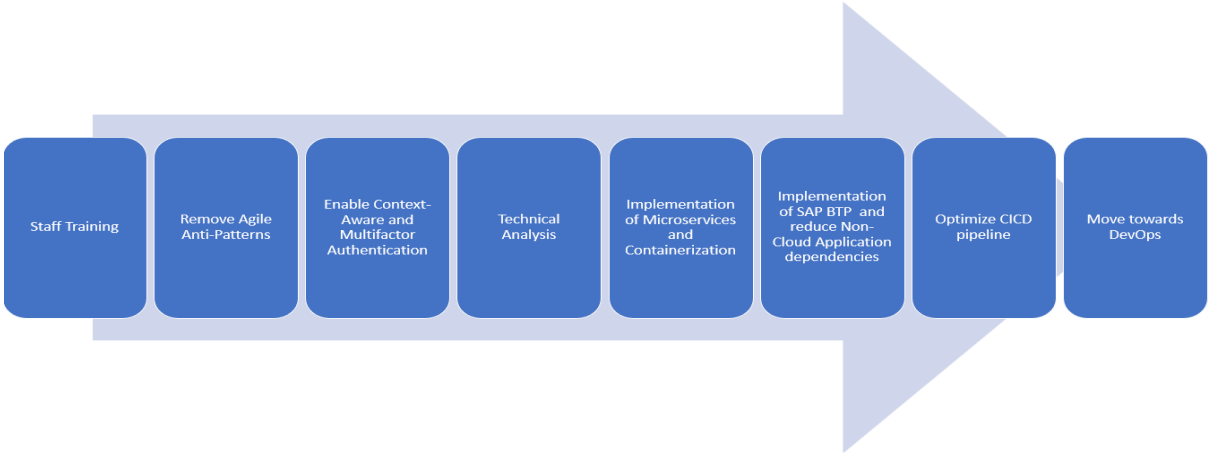

Figure 18. Action Plan for Implementation of Proposed System

4. Discussion

DevOps mainly focus on the quality and speed of software releases (Sugandhi,2022). However, it does not integrate security into the process (ibid). DevSecOps combines the security into the SDLC by including security teams along with development and operation teams (ibid). As the organisation grows, the microservices management may become too complex. In that case, Google Kubernetes Engine will be more suitable for container management because it's a CaaS platform which is more powerful than Docker Swarm.

5. Conclusion

The requirement of ElixirCure's CEO was to propose a strategy to speed up the drug development process with the help of software applications. As the initial step, this study analysed the existing system, organisation structure, work culture and identified improvement areas. CNC was propofor technical modernization and project management. The key elements covered in technical modernization were Microservices, Containerization, and CICD. As part of project management improvisation, this study covered best practices for sprint planning, daily stand-up meetings and retrospectives. This report highlighted the benefits, challenges, and mitigation actions of the suggested approach. Finally, the report outlined a high-level action plan for ElixirCure.

6. References

- Abdollahi Vayghan, L. *et al.* (2018) “Deploying microservice based applications with Kubernetes: Experiments and Lessons Learned,” *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 970–973. Available at: https://doi.org/10.1109/cloud.2018.00148
- Adobe, C.T. (2022) *Sprint retrospective: Meeting. agenda. understanding | Adobe Workfront*, *Adobe Experience Cloud Blog*. Adobe. Available at: https://business.adobe.com/blog/basics/sprint-retrospective (Accessed: February 25, 2023).
- AiCure (no date) *AiCure*, *AiCure | SourceForge*. SourceForge. Available at: https://sourceforge.net/software/product/AiCure/ (Accessed: February 21, 2023).
- Aqua (2022) *Docker orchestration: Swarm vs kubernetes in 2021*, *Aqua*. Available at: https://www.aquasec.com/cloud-native-academy/docker-container/docker-orchestration/ (Accessed: March 1, 2023).
- Aqua (2023) *10 kubernetes alternatives and why you need them*, *Aqua*. Available at: https://www.aquasec.com/cloud-native-academy/kubernetes-101/kubernetes-alternatives/ (Accessed: March 1, 2023).
- Arachchi, S.A.I.B.S. and Perera, I. (2018) “Continuous integration and continuous delivery pipeline automation for Agile Software Project Management,” *2018 Moratuwa Engineering Research Conference (MERCon)*, pp. 156–160. Available at: https://doi.org/10.1109/mercon.2018.8421965
- Argus, O. (no date) *Argus - safety case management*, *Argus - Safety Case Management | Oracle United Kingdom*. Oracle. Available at: https://www.oracle.com/uk/industries/life-sciences/pharmacovigilance/argus-safety-case-management/ (Accessed: February 21, 2023).
- AXELOS (2015) “An overview of Agile - Agile Basics,” in *Prince2 AGILETM*. Norwich, England: AXELOS, pp. 26–27. Available at: https://r1.vlereader.com/EpubReader?ean=1780113314911
- Azure, M. (no date a) *What is Paas? platform as a service: Microsoft Azure*, *Platform as a Service | Microsoft Azure*. Microsoft. Available at:

https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas/ (Accessed: March 1, 2023).

- Azure, M. (no date b) *What is Saas? software as a service: Microsoft Azure*, *Software as a Service | Microsoft Azure*. Microsoft. Available at: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas/ (Accessed: March 1, 2023).
- Beda, J. (2019) *What are cloud native applications?*, *VMware Tanzu*. Available at: https://tanzu.vmware.com/cloud-native (Accessed: February 24, 2023).
- Bentaleb, O. *et al.* (2021) "Containerization technologies: Taxonomies, applications and challenges," *The Journal of Supercomputing*, 78(1), pp. 1144–1181. Available at: https://doi.org/10.1007/s11227-021-03914-1
- Boschetti, M.A. *et al.* (2014) "A lagrangian heuristic for sprint planning in Agile Software Development," *Computers & Operations Research*, 43, pp. 116–128. Available at: https://doi.org/10.1016/j.cor.2013.09.007
- Boudlal, H., Serrhini, M. and Tahiri, A. (2022) "Cloud computing for Healthcare Services: Technology, application and Architecture," *2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC)* [Preprint]. Available at: https://doi.org/10.1109/isivc54825.2022.9800212.
- Bucchiarone, A. *et al.* (2020) *Microservices: Science and engineering*. Cham, Switzerland: Springer, p. 46-47.
- Buchanan, I. (no date) *Containers as a Service*, *Atlassian*. Atlassian. Available at: https://www.atlassian.com/microservices/cloud-computing/containers-as-a-service#:~:text=Containers%20as%20a%20service%20(CaaS)%20is%20a%20cloud%2Dbased,by%20using%20container%2Dbased%20virtualization (Accessed: March 1, 2023).
- Campbell, J. (no date) *Kubernetes vs. Docker*, *Atlassian*. Edited by C. Harris. Available at: https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker#:~:text=While%20Docker%20is%20a%20container,CRI%20(Container%20Runtime%20Interface). (Accessed: February 28, 2023).

- CNCF (2022) *Foundation/charter.md at Main · CNCF/Foundation*, *GitHub*. Available at: https://github.com/cncf/foundation/blob/main/charter.md (Accessed: February 24, 2023).
- Datadog (2014) *Docker monitoring*, *Datadog*. Available at: https://www.datadoghq.com/blog/monitor-docker-datadog/ (Accessed: March 5, 2023).
- Docker (2023) *Orchestration*, *Docker Documentation*. Docker. Available at: https://docs.docker.com/get-started/orchestration/ (Accessed: March 1, 2023).
- Eichler, G. (2018) *Kraft Precision Medicine Accelerator - Landscapes Data and AI*. rep. Oak Health Partners, LLC. Available at: https://www.hbs.edu/kraft-accelerator/assets/downloads/leaders/KPMA%20HBX%20Landscape%20-%20Data%20and%20AI.pdf (Accessed: February 21, 2023).
- Ganguly, S. (2022) *7 CI/CD challenges & their must-know solutions*, *BrowserStack*. Available at: https://www.browserstack.com/guide/ci-cd-challenges-and-solutions (Accessed: March 5, 2023).
- Genomics, E. (no date) *Origins*, *Genomics England*. Department of Health and Social Care. Available at: https://www.genomicsengland.co.uk/about-us/origins (Accessed: February 21, 2023).
- GenomOncology (no date) *Precision Oncology Software and Data Solutions*, *GenomOncology*. GenomOncology. Available at: https://www.genomoncology.com/#:~:text=GenomOncology%20provides%20the%20healthcare%20community,insights%20to%20drive%20measurable%20results (Accessed: February 21, 2023).
- Gill, S.K. *et al.* (2016) *Emerging role of bioinformatics tools and software in evolution of clinical research*, *Perspectives in clinical research*. U.S. National Library of Medicine. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4936069/ (Accessed: February 21, 2023).
- Gitential (2023) *An order of Agile, CI/CD and devops to go, please*, *Gitential*. Available at: https://gitential.com/an-order-of-agile-ci-cd-and-devops-to-go-please/ (Accessed: March 5, 2023).

- Google , C. (no date) *Enforce uniform MFA to company-owned resources | cloud identity | google cloud*, *Google*. Google. Available at: https://cloud.google.com/identity/solutions/enforce-mfa (Accessed: March 1, 2023).
- Hemon, A. *et al.* (2019) “From agile to devops: Smart skills and collaborations,” *Information Systems Frontiers*, 22(4), pp. 927–945. Available at: https://doi.org/10.1007/s10796-019-09905-1
- Hira, Z. (2022) *Kubernetes vs docker swarm – what is the difference?*, *freeCodeCamp.org*. freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/kubernetes-vs-docker-swarm-what-is-the-difference/#:~:text=The%20major%20difference%20between%20the,preferable%20choice%20for%20simple%20applications. (Accessed: March 1, 2023).
- IBM (no date) *Continuous deployment: An essential guide*. IBM. Available at: https://www.ibm.com/uk-en/topics/continuous-deployment#:~:text=Continuous%20deployment%20is%20a%20strategy,directly%20to%20the%20software's%20users. (Accessed: March 5, 2023).
- Jenkinsci (2021) *Jenkinsci/docker-swarm-plugin: Jenkins plugin which allows to add a docker swarm as a cloud agent provider*, *GitHub*. Available at: https://github.com/jenkinsci/docker-swarm-plugin (Accessed: March 5, 2023).
- JetBrains (no date) *CI/CD in Agile Development: Teamcity CI/CD guide*, *JetBrains*. Available at: https://www.jetbrains.com/teamcity/ci-cd-guide/agile-continuous-integration/#:~:text=Enabling%20Agile%20with%20CI%2FCD&text=Continuous%20integration%2C%20delivery%20and%20deployment%20are%20DevOps%20practices%20that%20aim,to%20release%20software%20to%20users. (Accessed: March 5, 2023).
- Kanjilal, J. (2020) *Microservices logging best practices every team should know: TechTarget*, *App Architecture*. TechTarget. Available at: https://www.techtarget.com/searchapparchitecture/tip/5-essential-tips-for-logging-microservices#:~:text=When%20an%20error%20occurs%2C%20the,Service%20name (Accessed: February 26, 2023).

- Khalid, T. (2022) *Scaling and optimizing CI/CD*, *Geekflare*. Available at: https://geekflare.com/scaling-and-optimizing-ci-cd/#:~:text=Scaling%20CI%2FCD&text=It%20also%20makes%20the%20management,quality%20of%20the%20delivered%20code. (Accessed: March 5, 2023).
- Mak, K.-K. and Pichika, M.R. (2019) "Artificial Intelligence in drug development: Present status and future prospects," *Drug Discovery Today*, 24(3), pp. 773–780. Available at: https://doi.org/10.1016/j.drudis.2018.11.014 (Accessed: February 19, 2023).
- Margreth, M. and Neufeld, R. (no date) *Metrohm tiamo - networked titration data system software by Metrohm AG: Environmental XPRT*, *Metrohm tiamo - Networked Titration Data System Software*. Environmental Expert . Available at: https://www.environmental-expert.com/software/metrohm-tiamo-networked-titration-data-system-software-62410 (Accessed: February 21, 2023).
- Microsoft (no date) *What is cloud native?*, *Microsoft Learn*. Available at: https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/definition (Accessed: February 24, 2023).
- Milne, A. (2022) *Five agile anti-patterns to avoid with solutions*, *Insights - Web and Mobile Development Services and Solutions*. Available at: https://www.netsolutions.com/insights/agile-anti-patterns/ (Accessed: February 23, 2023).
- Newstex (2022) "*Google Cloud Blog - Google: Data: the Rx to faster, patient-centric clinical trials.*" Mountain View. Available at: https://www.proquest.com/docview/2747041067?accountid=14888&parentSessionId=mvJUUF0kGOH9LH160FRYzhVHCUFISvtxGpM%2BrBP92zU%3D (Accessed: February 12, 2023).
- Patterson, J. (2017) *Deploying docker swarm with Ansible*, *Medium*. Medium. Available at: https://medium.com/@cantrobot/deploying-docker-swarm-with-ansible-a991c1028427 (Accessed: March 5, 2023).
- Pedersen, S. (2022) *How to avoid 8 common agile anti-patterns hurting your team*, *BuildingBetterSoftware*. Available at:

https://www.buildingbettersoftware.com/common-agile-anti-patterns-hurting-teams-how-to-avoid/ (Accessed: February 23, 2023).

- Raja, B.H. and Sambandan, P. (2015) "*Open Innovation in Pharmaceutical Industry: A case study of Eli Lilly"*. thesis. KTH School of Industrial Engineering and Management (ITM) Department of Industrial Economics and Management. Available at: https://www.diva-portal.org/smash/record.jsf?dswid=5512&pid=diva2%3A824465 (Accessed: February 18, 2023).
- Ranchal, R. *et al.* (2020) "Disrupting healthcare silos: Addressing data volume, velocity and variety with a cloud-native healthcare data ingestion service," *IEEE Journal of Biomedical and Health Informatics*, 24(11), pp. 3182–3188. Available at: https://doi.org/10.1109/jbhi.2020.3001518.
- Reinhardt, I.C., Oliveira, D.J. and Ring, D.D. (2020) "*Current perspectives on the development of Industry 4.0 in the pharmaceutical sector*," Journal of Industrial Information Integration, 18, p. 100131. Available at: https://doi.org/10.1016/j.jii.2020.100131 (Accessed: February 12, 2023).
- Rosen, C. (2022) *Docker Swarm vs. Kubernetes: A comparison*, *IBM*. Available at: https://www.ibm.com/cloud/blog/docker-swarm-vs-kubernetes-a-comparison (Accessed: March 1, 2023).
- SAP, L.S.S. (no date a) *Life Sciences Solutions: Software and Technology*, *SAP Life Sciences Solutions*. SAP. Available at: https://www.sap.com/industries/life-sciences.html (Accessed: February 21, 2023).
- SAP (no date b) *SAP Business Technology Platform*. SAP. Available at: https://www.sap.com/products/technology-platform.html?btp=35dcce81-d42e-4941-86c4-550fa9a2b56b (Accessed: March 1, 2023).
- Sparta Systems, H. (2022) *Trackwise QMS software and solutions*, *Sparta Systems*. Honeywell. Available at: https://www.spartasystems.com/trackwise-qms-software/ (Accessed: February 21, 2023).
- Steven, J. (2021) *What's the difference between agile, CI/CD, and DevOps?*, *Application Security Blog*. Available at: https://www.synopsys.com/blogs/software-security/agile-cicd-devops-difference (Accessed: March 5, 2023).

- Sugandhi, A. (2022) *DevOps VS DevSecOps: Top differences*, *KnowledgeHut*. Knowledgehut. Available at: https://www.knowledgehut.com/blog/devops/devops-vs-devsecops (Accessed: March 5, 2023).
- Sullivan, D. (2020) “Designing for Security and Compliance,” in *Google Cloud Certified Professional Data Engineer*. S.l.: WILEY-SYBEX, pp. 139–141.
- Thones, J. (2015) “Microservices,” *IEEE Software*, 32(1), pp. 116–116. Available at: https://doi.org/10.1109/ms.2015.11
- VelQuest, C. (no date) *VelQuest - Crunchbase Company Profile & Funding*. Crunchbase. Available at: https://www.crunchbase.com/organization/velquest (Accessed: February 21, 2023).
- Viggiato, M. *et al.* (2018) *Microservices in Practice: A Survey Study* [Preprint]. Available at: https://doi.org/ https://doi.org/10.48550/arXiv.1808.04836
- Waters, E. (no date) *Empower Chromatography Data System*, *Waters*. Available at: https://www.waters.com/waters/en_IE/Empower-Chromatography-Data-System/nav.htm?locale=en_IE&cid=513188 (Accessed: February 21, 2023).
- Wu, F. *et al.* (2020) “Computational approaches in preclinical studies on drug discovery and development,” *Frontiers in Chemistry*, 8, pp. 1–32. Available at: https://doi.org/10.3389/fchem.2020.00726 (Accessed: February 21, 2023).