IVVES

Industrial-grade Verification and Validation of Evolving Systems

Labeled in ITEA3, a EUREKA cluster, Call 5

ITEA3 Project Number 18022

D4.1 – State of the Art on Data-Driven Engineering

Due date of deliverable: June 30, 2020
Actual date of submission: June 30, 2020

Start date of project: 1 October 2019 **Duration:** 39 months

Organisation name of lead contractor for this deliverable: RISE

Author(s): Mehrdad Saadatmand, Muhammad Abbas, Niclas Ericsson, Mahshid Helali Moghadam, RISE, SWE; Almira Pillay, Tijana Nikolic, Sogeti, NL; Matvey Pashkovskiy, F-Secure, FIN; Marcel Hogenhout, Bas Stoker, Praegus, NL; Ivar Simonsson, Ekkono Solutions, SWE; Tanja Vos, Pekka Aho, Open Univ. of the Netherlands, NL, Yaping Luo, ING, NL; Elio Saltalamacchia, Antonio Alcaide, David Diaz, SII Concatel, ES; Jesús Arce, Oscar Luis Gomez, Keyland, ES; Juan Corral, Fernando Tornero, Jesus Rio, Netcheck, ES; Juan L. Sanchez, Aunia, ES

Status: Final

Version number: 1.0

Submission Date: 30 June 2020

Doc reference: IVVES_Deliverable_D4.1_SoTA-DDE_V1.0.docx

Work Pack./ Task: WP4

Description:
(max 5 lines)

Nature:	☑ **R**=Report, ☐ **P**=Prototype, ☐ **D**=Demonstrator, ☐ **O**=Other		
Dissemination Level:	**PU**	Public	**X**
	PP	Restricted to other programme participants	
	RE	Restricted to a group specified by the consortium	
	CO	Confidential, only for members of the consortium	

DOCUMENT HISTORY

Release	Date	Reason of change	Status	Distribution
V0.1	26/05/2020	First draft	Draft	All
V0.2-0.8		Merging and alignment of partner inputs	Draft	WP4
V0.9	29/06/2020	Final merge and style updates	Concept	Submitted to PMT
V1.0	30/06/2020	Approved by PMT, to be submitted to ITEA3	Final	Uploaded to ITEA

Table of Contents

Glossary

Abbreviation / acronym	Description
ADA	Artificial Data Amplifier
AI	Artificial Intelligence
API	Application Programming Interface
CI	Continuous Integration
CIA	Change Impact Analysis
CoEST	Center of Excellence for Software and Systems Traceability
CD	Continuous Deployment
DIA	Dependency Impact Analysis
EDA	Exploratory Data Analysis
EIA	Experimental Impact Analysis
ES	Evolving Systems
ET	Exploratory Testing
FODA	Feature-Oriented Domain Analysis
FPP	Full Project Proposal
GUI	Graphical User Interface
HA	High availability
LSA	Latent Semantic Analysis
MTBF	Mean Times Between Failure
ML	Machine Learning
NLP	Natural Language Processing
OCL	Object Constraint Language
OSLC	Open Services for Lifecycle Collaboration
GDPR	General Data Protection Regulation
PII	Personally Identifiable Information
PLC	Programmable Logic Controller
RCA	Root Cause Analysis
RDL	Requirements Description Language
SAFe	Scaled Agile Framework
SLA	Service Level Agreement
SUT	System Under Test
TIA	Traceability Impact Analysis
UML	Unified Modelling Language
URN	User Requirements Notation
WP	Work Package

1. Executive Summary

WP4 in IVVES, titled ‘Data-Driven Engineering’, focuses on implementing solutions for identifying data correlations and behavioural patterns throughout the entire product life cycle with respect to component failures, and with the ultimate goal of enabling predictive maintenance and anomaly detection. In particular, application of machine learning techniques is considered to achieve this goal and *making sense* of the collected data. From this perspective, the scope of WP4 will cover both the development phases as well as system operation.

D4.1 is the first deliverable of WP4 and reports on the core concepts, techniques and topics that will be used and investigated for building the solutions of WP4, introducing the key terms and offering a brief state of the art overview in each topic area.

2. Introduction – Data Driven Engineering

Shortening the feedback cycles, particularly with respect to customer inputs, has always been one of the general challenges in engineering of software systems. Approaches such as continuous integration (CI) and continuous deployment (CD) are examples of solutions to perform more frequent tests and deployments in order to shorten such feedback cycles. Moreover, they can enable collection of diagnostic, performance and operational data in each iteration to quickly learn the implications of changes and new features and react accordingly [1]. In such a context, systematic collection and use of data is not only an effective way to replace opinion-based decision making with data-driven decision making about system performance and quality characteristics [1], but also can enable the automation of such a decision making process with the help of techniques such as machine learning.

Data-driven development is defined as “the ability of a company to acquire, process, and leverage data in order to create efficiencies, iterate and develop new products, and navigate the competitive landscape” [1], [2]. In IVVES and particularly in the scope of WP4, we define the term data-driven engineering to refer to **an engineering process based on systematic collection and processing of data and automation of data-driven decision making with the purpose of improving both the overall quality characteristics of a system and its development process**. For this to happen, various data sources and data artifacts can be considered from not only the development phases of a system, but also its operation and during its runtime (e.g., runtime logs and monitoring information); hence the focus on DevOps in this WP.

In connection to this, the work will especially include the application of AI and ML to find data correlations and behavioural patterns throughout the entire product life cycle, for instance with respect to component failures, and to avoid error prone manual labour to resolve problems earlier (left-shifting) and increase automation during development and in operations for specific parts of the project use cases.

To achieve the above-mentioned goals of WP4, in this deliverable the following topics are investigated as the baseline for the work and solutions that will be developed in this WP:

- Data handling, management, and privacy: dealing with aspects of privacy and security, anonymization and sharing of sensitive industrial data among different stakeholders
- Data sources and virtual sensors: discussing various sources of data that can be used in a data-driven engineering process
- Data quality: discussing quality of input data and its important role in performance, accuracy and output of machine learning algorithms and for data-driven decision making
- xOps: engineering processes suitable for rapid development and delivery of evolving systems while improving their quality at the same time
- Predictive maintenance: processing of data to predict future failures and plan maintenance activities accordingly
- Model synthesis, extraction and construction: using data to construct models of a system
- Fault prediction: using different software metrics, properties, and fault data to predict faulty modules typically before dynamic testing
- Change impact and root cause analysis: processing of data to identify and analyze the effects of a change and also the root cause of a problem
- Data collection in exploratory testing: use of data to guide and automate exploratory testing

3. Data Handling and Management

Data security and usage have, in recent light, become an important consideration for not only companies but also individuals whose data is being collected and subsequently protected. Every day, we produce around 2.5 quintillion bytes of data [3], be it in the form of social media posts, tweets, transactions, likes, web searches etc. This data is invaluable to companies as they use it to build and understand their customer profiles, look for trends, identify opportunities, tailor better services and products, and even anticipate events to capitalise on. However, this data can also be used to exploit, influence and abuse. This is why we need regulations, like GDPR, in place that govern and hold companies and individuals accountable for the way they use and gather the data.

3.1 GDPR implications

GDPR evolved from a rule to become a regulation – the first of its kind in the European Union. Under this regulation, personal data or PII is protected by restricting the processing and usage of the data. This regulation protects the end consumer and empowers them to be able to choose what happens with their data and understand how companies are using their data and for what purpose – this is known as 'right of access'. Under this regulation, individuals can choose whether or not companies can use their data for different purposes. Companies have to delete any data they might have from the individual if the individual decides to revoke their right of access.

Another feature of the GDPR focuses on the usage of the data and prohibits companies to use data other than for specific purposes that are inherent to their business models. The companies need to be able to state what data they collect and for what purpose, making their business model transparent. For example, if a company is using production data for testing, this could amount to unlawful processing and result in large financial fines. This is especially non-compliant if it was not explicitly stated what the data would be used for when getting the consent from the individual.

In contrast, in data analytics and AI model development, repurposing of data is a natural thing, meaning that more efforts need to be put in place in order to explain how the data will be used. Oftentimes AI models are called black boxes, leading to some unexpected conclusions when fed the data. It is a company's responsibility to account for these outcomes, making their models trustworthy and transparent [4].

3.2 Solutions on the market

3.2.1 Pseudonymisation

There are, of course, ways to avoid incurring high fines and one of those methods is to use pseudonymised/masked data. The usage of pseudonymised data is more relaxed under GDPR and does not have the strict regulations to comply with however, there is still a risk of a data breach. Even better is the use of anonymised data, which is not regulated by GDPR data; anonymisation techniques include encryption, unique identifiers (replacing a PII value with a symbol), data shuffling and premutation, perturbation, or removing sensitive data completely., Although this anonymised data is not regulated by GDPR, although this data it does comes with risks as well. Under GDPR, anonymised data is data that cannot be traced back to a certain individual, but recent studies have shown that anonymised data can still be traced back to identifying the underlying individuals, which makes this strategy still susceptible to adversarial attacks [5]. For example, in one study involving the analysis of anonymised credit card metadata, it was shown that 90% of the individuals in the dataset could be re-identified with only 4 random attributes [6]. The results of the study underlines how difficult it is for a dataset to be truly anonymous given the risk of re-identification, which poses a massive privacy risk for the company or institution. This is where the power of synthetic data shines.

3.2.2 Synthetic data

Synthetic data looks and feels just like the real data holding all the characteristics, and relationships and referential integrity present in the real data. Sogeti's AI team has developed a new solution to create synthetic data with AI called ADA – Artificial Data Amplifier. ADA uses state of the art, advanced neural networks to generate synthetic data that can then be used in place of real data. This method of generating data differs from current data generation tools that are often included in test data management solutions and other masking tools on the market, in that does not purely scramble data or use a rule-based algorithm. The AI technology in ADA trains on real data such as production data, learns the qualities of the data, and then generates completely new data based on those qualities. In this way, ADA can generate production like data for Testing and Development or even data to be used for building an AI model. Not only can ADA generate tabular data, but it can generate images including documents and unstructured text too. This expands the use of ADA to many applications beyond Testing. ADA is not a generic data management tool; it is a custom solution that needs to be trained on real data. Typically, ADA extracts a dataset used in an application, environment, database or report. It then trains on the data, generates synthetic data and pushes it back into your databases. The advantages of using synthetic data over real data are two-fold. First, the advantage of creating an entire dataset that looks and feels like your real data but without the security risk of any data breach is valuable for companies that operate in very highly regulated industries. This ensures that data can be processed, shared and analysed securely and compliantly. Secondly, this solution is scalable meaning that we can create endless amounts of data based on a small sample of the real data. The advantage here is that we can create enough data for testing, particularly performance testing, that is once again, GDPR compliant as it is purely synthetic. Scalable data also allows a dataset to be boosted in the case of needed more training for an AI model. An example of this is boosting images to improve the accuracy of a computer vision model since these types of deep learning models thrive on an abundance of data.

4. Data Sources

Testing evolving systems is a complex challenge. We have identified the following data sources that need to be considered.

Source Code. Source code is the core and beginning just about any software centric activity. In the context of IVVES, there are three dimensions to consider:

1) Full source code files, which can be meaningful for debugging, test planning, and so on. This is usually not a problem as the source code files form the essentials of software development in any case.

2) History view to the source code files, organized for instance in accordance to version information. It is well known that programmers use slices while debugging (Weiser 1982), and hence relying on this in IVVES seems a rational step to focus testing to certain features. Interfaces to such information is available in various systems. For instance, there is a GitHub API for code changes.

3) Details of contributions from the developers can be used to understand the evolution of the system, as well as the development style. This in turn can help in figuring out the best possible test policy.

Test plan and test reports. Test plan and test reports are commonly available for software developers. In the scope of IVVES, however, we assume a wider view than commonly associated in testing and include detailed information regarding test failures in unit and integration tests including in particular stack traces.

Logs, traces and monitoring data. Today, applications produce logs regarding their behavior. These logs can offer insights to the internal operations of an application, and hence support debugging its internals. These logs can be produced by the application itself -- in which case we typically refer them as logs -- they can be a result of instrumentation, in which case we more typically call them traces. Furthermore, also the hosting instrumentation can monitor the execution of the application and provide insight to its behavior. These sources of data can be taken into account in testing, as they indicate the profile of the application in terms of frequency of use, real-life use cases, and so on. Finally, the logs may be produced by the development tools, too, in which case they provide extra information for developers.

Human feedback. Human feedback can also help in defining tests for evolving systems. Potential sources of such feedback include code reviews, where systematic errors or parts that need extra testing attention can be identified, and user feedback via ticket/issue databases or discussion forums, which may flag problematic situations that were not considered by the developers.

While there are plenty of sources of data, this data is not compatible as such. Software tools used in the modern software development pipeline produce log or job result data typically in JSON or XML format, although some, for example build tools, often produce plain text logs. SoData models are typically tool specific, unit testing frameworks being an exception as the xUnit-format is widely adopted. To summarize these findings:

- Data models are tool specific, except in Unit Testing area
- JSON, XML and plain text formats
- Data rarely contain metadata or context information
- Data model/format specifications not always available.

OSLC (Open Services for Lifecycle Collaboration) is an emerging standard tool for interoperability. It has been widely used for tool integration in public funded research projects, and plugins are available for several tools. OASIS OSLC proposes the application of web principles to software interconnection. These specifications allow to conform independent software and product lifecycle tools to integrate the data and workflows in support of end-to-end lifecycle processes. However, the use of OSLC is not an industrial practice yet, and has not been used for data collection in a manner similar to the IVVES approach.

5. Virtual Sensors

A virtual sensor is used in place of a physical sensor when a physical sensor is for some reason undesirable or impossible to deploy. For example, the environment could be too hostile for a physical sensor or it could be too expensive to deploy and/or maintain. Available physical sensors are used to model virtual sensors, either by a model-based approach using known physical relationships, or by a data-driven approach to model the target signal [7] [8].

Basing on usage scenarios two separate groups of virtual sensors could be identified:

- Simulators – a type of software that, given the available information, processes what a physical sensor otherwise would. It learns to interpret the relationships between the different variables, and observes readings from the different instruments.
- Aggregators – is a software sensor as opposed to a physical or hardware sensor, providing indirect measurements of abstract conditions (that, by themselves, are not physically measurable) by combining sensed data from a group of heterogeneous physical sensors.

Basing on that the virtual sensor could be defined as a software representing and used together or by itself with the hardware device or group of devices and possibly extending its data processing capabilities.

5.1 Simulators

Naturally, many machine learning techniques have been used to model data-driven virtual sensors. In [9], vehicle sideslip angles are observed using four different virtual sensors, including Extended Kalman Filter. As in [10], linear regression models have been used to create virtual sensors in ventilation units, and [11] introduces virtual sensors that are trained with neural networks to replace physical sensors in diesel engines. In [12], more traditional statistical problems (missing data, outliers, collinearity) are discussed within the framework of virtual sensors.

Other application areas discussed in the literature include aviation [13], [14], NOx emission from industrial boilers [15], structural dynamics [16], and image analysis [17]. Additionally, with growing popularity for cloud-based solutions, architectures for incorporating virtual sensors in the cloud has been investigated [18], [19].

5.2 Aggregators

In existing deployments of sensor networks, data collection schemes commonly require sensors to relay raw data to sink nodes to perform further processing. This is not very efficient considering the resource constraints (e.g., battery and bandwidth) of sensor networks. Furthermore, the throughput at each node decreases as the network scales, due to the broadcasting of redundant data. Sensor network aggregation mechanisms [20], [21], [22], [23] offer in-network data processing algorithms that are successful in limiting resource usage. However, these approaches support only standard mathematical operators (e.g., MIN, COUNT, and AVG) over homogeneous data types but there are cases in which the desired physical quantity cannot be measured directly for cost, energy, convenience or other practical reasons. Or, however, a qualitative statement rather than a quantitative measurement is desired. In that cases aggregators can combine different quantities, which are measured by real sensors, and then deliver the desired result. The sensor networks of tomorrow will need to support localized cooperation of sensor nodes to perform complicated tasks and in-network data processing to transform raw data into high-level domain-dependent information. Another challenge facing sensor networks is reusability. Current efforts create application-specific solutions, but the future will see multipurpose nets deployed to support numerous applications. The cost of physically visiting each sensor to reprogram it is prohibitive, and therefore the ability remotely reprogram sensor networks to tailor them to particular applications will be essential.

As an example, aggregators could be used on an intelligent construction site where users may desire the cranes to have safe load indicators that determine if a crane is exceeding its capacity. Such a virtual sensor would take measurements from physical sensors that monitor boom angle, load, telescoping length, two-

block conditions, wind speed, etc. [24]. Signals from these individual sensors can be used in calculations within a virtual sensor to determine if the crane has exceeded its safe working load.

Aggregators could be deployed either on one of the sensors or on a separate device (e.g. laptop) outside of the sensors network.

6. Data Quality

Data is the foundation of any model, which means that data quality is the primary point of assessment when starting any model development project. Good quality data is data that is fit for the purpose you are intending to use it for, meaning it can represent the real-world correctly. Insight into handled data brings more value to the model development process, especially when we are talking about AI models, where data is used for training and any underlying assumptions and bias can lead to a wrong assumption of the model itself. To sum up, accurate representation in data means accurate models. Below is a review of methods which can be used to gain insight into data and assess if it represents the problem we are trying to solve correctly.

6.1 Quality AI Framework (QAIF)

Inspired by the CRISP-DM Framework [25] QAIF [26] is a cohesive, generic framework that can be tailored to a specific AI solution in a given business context. The framework is comprised of six gates that follow the process flow of the AI project development cycle (CRISP-DM), Business Understanding, Data Understanding, Data Preparation, Model Development, Model Evaluation and Model Deployment. The gates can be broken down into project phase, processes, outcomes, governance and people. In each gate, there are specific tasks that need to be complete for the gate to be passed through in order to enter the next gate. This ensures that each phase of the AI development cycle is validated thoroughly. For the purposes of this document, we are going to focus on Data Understanding and Data Preparation phases or gates, while the rest of the gates are further described in D2.1 in the scope of work package 2 of the IVVES project.

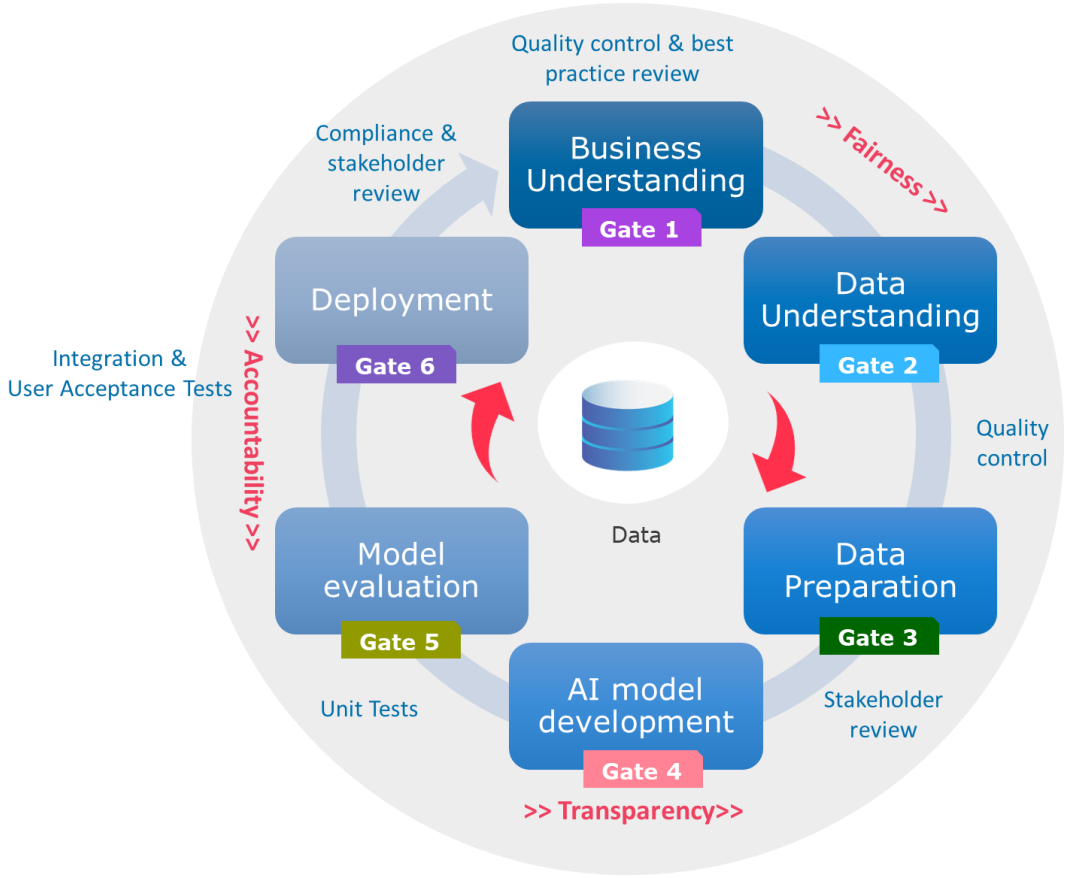


Figure 1 - Sogeti Quality AI Framework

Data Understanding phase defines Gate 2 in the QAIF, bringing model specifications from the first gate and domain knowledge and experience together in order to understand inherent biases or assumptions of the data this solution will be dealing with. In this phase, EDA is performed through methods like statistical parity and many more, results of which can be presented in form of a data Quality report. The data Quality report will aid the model development team in determining the initial model and the metrics it will be evaluated on.

EDA is a set of methods used to analyse data, often visually. EDA is performed to make sure the data aligns with our definition of quality and expectations, aiding in detecting mistakes, assumptions, dependencies and relationships of data variables. Of course, when working with tabular data format, these points can be addressed manually, but this is extremely tedious and error prone, making EDA a useful and helpful approach.

The EDA methods can be classified in a couple of ways, but the most high-level classification would be [27].

1) Graphical EDA, summarizing the data in a visual way, split into:

    a. Univariate, summarising one variable at a time. The graphic representation of univariate summary can be via line charts, bar, charts, stacked bar charts, scatter plots;
    b. Bivariate or multivariate, summarising two or multiple variables and analysing relationships. The analysis results are represented in the same way the univariate method does, with addition of area charts and histograms.

2) Non-graphical EDA, the calculation of summary statistics, split in the same manner.

The high-level classification is, more granularly, divided based on the role, type and variables examined during EDA.

The type of data or variable explored is of crucial importance when defining techniques executed during EDA. There are two types of data:

1) **Categorical data** or variable is data that can take on a limited number of values and is characterized by the range of values, their frequency or occurrence. This data is observed, not measured. The categorical variables are handled with below EDA techniques:

    a. **Tabulation**, a univariate non-graphical technique, where we count the number of groups in a categorical variable and calculate their frequency. One example is shown below, where the categorical variable of nocturnal forest animals has been split into its subgroups and analysed. Frequency is calculated by dividing the group count value with the total group count value.

Group	Group Count	Frequency (%)
Wolf	15	75
Fox	5	25
Total	20	100

    b. **Cross-tabulation**, a multivariate non-graphical method, is a two-way table with columns that match one variable and row levels that match the other. An example is shown below,

where we can see the count of 2compared variables is noted, giving a good overview of how they are correlated.

Sex/Species	Wolf	Fox
Male	10	2
Female	5	3
Total	15	5

c. **Histograms**, a univariate graphical EDA technique, is a barplot in which each bar shows the frequency or proportion of cases for a range of values. Histograms are a quick way to visually represent frequency and count of categorical data.

2) **Quantitative data** or variable is data in form of counts or numbers, which can and are measured. Their characteristics are spread, centre, modality, shape and outliers. These variables are handled with the following EDA techniques through sample statistics, where a sample is a set of individuals or objects selected from a population by defined procedure[1]:

a. **Central tendency** or location of a distribution, a univariate non-graphical method, is measured with mean (the sum of all of the data values divided by the number of values), median (middle value of an ordered sample) and mode (the kurtosis of data, how many peaks it has compared to Gaussian distribution). The skewness or asymmetry of data distribution is measured by looking at the mean and median, where a symmetry is reached when mean and median coincide. Similarly, if data is symmetric, it is unimodal with no outliers. If it's skewed, it has outliers.
b. **Spread** or a measure of similarity between sets in each sample is another univariate non-graphical method. It is characterized by variance (mean of individual squared deviations), standard deviation (the square root of the variance) and interquartile range (defined by quartiles, 3 values dividing the observed distribution into even fourths, showing dispersion of data). Mean and standard deviation give good insight into normality of data distribution, while interquartile range is a robust measure of spread.
c. **Boxplot,** a univariate graphical EDA technique, is a visual representation of spread, central tendency and skew. A side by side boxplot is a clear way to represent multivariate EDA findings.
d. **Quantile-normal plots,** another univariate graphical EDA technique, is a visual representation on skewness, kurtosis and bimodality.
e. **Correlation and covariance,** multivariate non-graphical methods, are used to explain how much similarly two random variables tend to deviate from expected values. Visually, they are represented in form of matrices, where relationships between variables can be seen clearly.
f. **Scatterplots**, a multivariate graphical method which shows relationships between quantitative variables, with one being on the x, and other on y axis.

Data Preparation: Bringing the insights from the previous gate, the Data Preparation phase can begin, where the data engineering team, domain experts and model developers play crucial roles. Tasks like data mining, data quality assessment and training data construction define this phase's process. The use of synthetic data is advocated when there is sensitive information, or the dataset needs to be boosted. The

outcome of this gate is high quality training data for the model, which enables the next phase of model development to begin.

Some data preparation techniques are defined below [28]:

1) **Handling missing or incomplete data**, a very important first step when starting data preparation is accounting for missing values. The developer can take a couple of routes here, removing the missing values, filling them in (imputing) based on mean, median and mode when handling quantitative or imputing based on multiple imputation and logistic regression in case of categorical variables [29].
2) **Handling improperly formatted data**, where data cannot be used in the format received and needs to be reformatted. Subject Matter Experts can be consulted in this case and data adjusted using their comments.
3) **Standardizing quantitative or qualitative variables**, where the format of quantitative variables can be in percentages, where it should be a number. Or, in case of qualitative variables, some naming conventions may misalign.
4) **Handling limited data or features**, where some features of the data may be present in one dataset, which is used to enrich another one by joining them together. Also, some data may be locked by GDPR regulations because it contains PII. In this case, the use of synthetic data for dataset boosting is required, please see paragraph 3 for more information.
5) **Feature engineering,** where additional variables are constructed using the original dataset to boost the model performance (A closer look at AI: data mining, https://brighterion.com/data-mining/). This step is important because it gives more insight into the data through data mining techniques such as association analysis (discovering correlation between variables), regression (discovering dependent variables), classification and prediction (grouping of variables in a given dataset).
6) **Imbalanced dataset,** where the dataset is not representative of real scenarios or there is a proportions bias (due to data collection methods) then it is advocated to collect more representative data where possible. If the imbalance is indeed representative of the context of the data, there are methods such as intelligent sampling and generating synthetic data. Undersampling will sample from the majority class with the aim of decreasing its points, oversampling will sample from the minority class in the aim of increasing its cardinality and synthetic data will generate new data points from the minority class to increase cardinality. (Baptiste Rocca, “Handling imbalanced datasets in machine learning” [30].

7. DevOps

DevOps is seen as a development methodology aimed at bridging the gap between Development (Dev) and Operations (Ops), emphasizing on communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices [31]. A lot of focus is aimed at continuously improve the system development life cycle, by applying CI and CD practices.

As DevOps often build on lean and agile practices, there are several ways to adopt this type of process improvements in an organization, for example, SAFe, that defines five core competences: Lean Agile Leadership, Team and Technical Agility, DevOps and Release on Demand Business Solutions and Lean System Engineering, and Lean Portfolio Management [32]. SAFe, blogs [33] and a recent report on the state of DevOps [34] all emphasize on the mix of technologies and organizational changes necessary to benefit and adopt the DevOps way of working.

7.1 State of DevOps

The Accelerate State of DevOps Report [34] present results from six years of research and data from over 31,000 professionals. Below is a high-level summary of the findings from the report:

- They recommend that organization start with the foundations and adopt a continuous improvement mindset
- Industry continues to improve, especially the elite performers
- The heart of the technology transformation and organization performance is quick, reliable and safe software delivery
- Best practices for scaling DevOps focus on community building structural solutions
- A differencing factor for elite performers continues to be the cloud
- Productivity improvements in DevOps can drive improved work/life balance
- Lightweight change approval process leads to improved speed and stability
- Four key metrics that drive improvement and capture effectiveness are identified:
    - Lead Time
    - Deployment Frequency
    - Change Fail
    - Time to Restore

The metrics, lead time and deployment frequency are related to software development, while change fail, and time to restore are related to software deployment. In addition to these four, are availability added to measure operational performance.

The report contains a lot of interesting findings regardless of whether or not a company has yet to adopt DevOps practices. Below is a list of findings and recommendations that are highly relevant and related to IVVES:

- A culture of psychosocial safety contributes to performance
- Investments in code maintainability, loosely coupled architecture and monitoring help performance
- Multi-cloud and hybrid-cloud solutions offer flexibility, control and availability, and in addition change how we think about costs for infrastructure and deployments
- Any team/organization can benefit in speed, stability and availability from using public, private or hybrid cloud
- As every organization is different with respect to technology, culture and processes, a holistic approach is recommended by first understanding the constraints in the current software delivery process
- Empowering teams to decide how to reach agreed short- and long-term goals, usually result in more flexible solutions and less need for detailed plans, thereby letting management focus more on high-level outcomes
- Change effort needs to be concurrent on team and organization levels, (especially large organizations with hierarchical structures often require organization-level efforts)
- A CI platform making it easy for teams to get feedback on automated tests can be beneficial for several teams
- CD at team level will have little impact, if they are depending on code from other teams

- Test automation have a significant impact on CD, and a positive impact on CI
- Loosely coupled architectures lead to that teams can independently change their systems on demand without affecting other teams
- Short-lived branches, small patches and automatic testing improves productivity
- Organizations with mission critical systems should have a disaster recovery plan, and perform disaster recovery testing (e.g., simulations disrupting the production systems, infrastructure and application failover)
- Disaster recovery exercises that are cross-organizational can improve more than the systems, for example improving the processes and communication related to the tested systems
- Change management is one of the biggest constraints in large organizations, responding to problems by adding more approvals will make things worse
- Measuring productivity using simple metrics such as lines of code, story points or nr of closed bugs result in unintended consequences that sacrifice the overall goal, e.g., preventing teams from helping others as it affects their results
- Usefulness and ease of use of engineering tools are highly important, but often tend to be ignores as they are expected to be experts
- Proprietary software tends to be costly to maintain and support, and seems to have a negative impact on performance, hence should companies evaluate which software is strategic and which is simply utility
- Access to internal and external information sources supports productivity, e.g., effective search in code repositories, ticketing systems and documents
- Technical dept negatively reduces productivity, and can be found in e.g., scripts, configuration files, infrastructure and application code

In addition to all of this the report also reflects on what really works, e.g., communities of practice and grassroots, while other such as training centers and centers of excellence tend to create silos and isolate the experts, that in addition may lack hands-on practice.

The report wraps up with an interesting quote that “DevOps is not a trend, and will eventually be the standard way of software development and operations, offering everyone a better quality of life.”

It should also be noted that there is no cookbook-style [35] or best strategy [36] to introduce DevOps in an organisation, since products and life-cycle processes differ [35]. In addition, implementing DevOps in a company should be approached as a long-term activity requiring a supportive culture and mind-set in addition to technical practices [37]. Therefore, we envision that the data driven engineering approach within IVVES will act as a catalyst for continuous improvements and DevOps practices.

7.2 xOps

In the following subsections, we briefly look at other related terms and methodologies derived from DevOps concepts that are also important for the data-driven engineering work in IVVES.

7.2.1 DataOps

DataOps is another xOps term that has been coined in recent years as big data and data analytics concepts and solutions have gained widespread recognition. Here we quote three of the definitions that have been offered for DataOps by various sources:

- DataOps (data operations) is the orchestration of people, processes, and technology to deliver trusted, business-ready data to data citizens, operations, applications and artificial intelligence (AI) fast, according to IBM [38].
- According to Gartner [39], DataOps is a collaborative data management practice focused on improving the communication, integration and automation of data flows between data managers and consumers across an organization. Much like DevOps, DataOps is not a rigid dogma, but a

principles-based practice influencing how data can be provided and updated to meet the need of the organization's data consumers.

- DataOps is the function within an organization that controls the data journey from source to value [40]

One of the key differences between DataOps and DevOps is that the main focus in DevOps is on application and software development, while DataOps focuses more on trusted and high quality data available for fast use [38]. According to Palmer [41], two trends that created the need for DataOps are: i) the democratization of analytics and giving more people access to data management and analytics solutions, and ii) the implementation of built-for-purpose database engines, basically enabling big data capabilities by improving performance and accessibility of large amount of data at fast speeds.

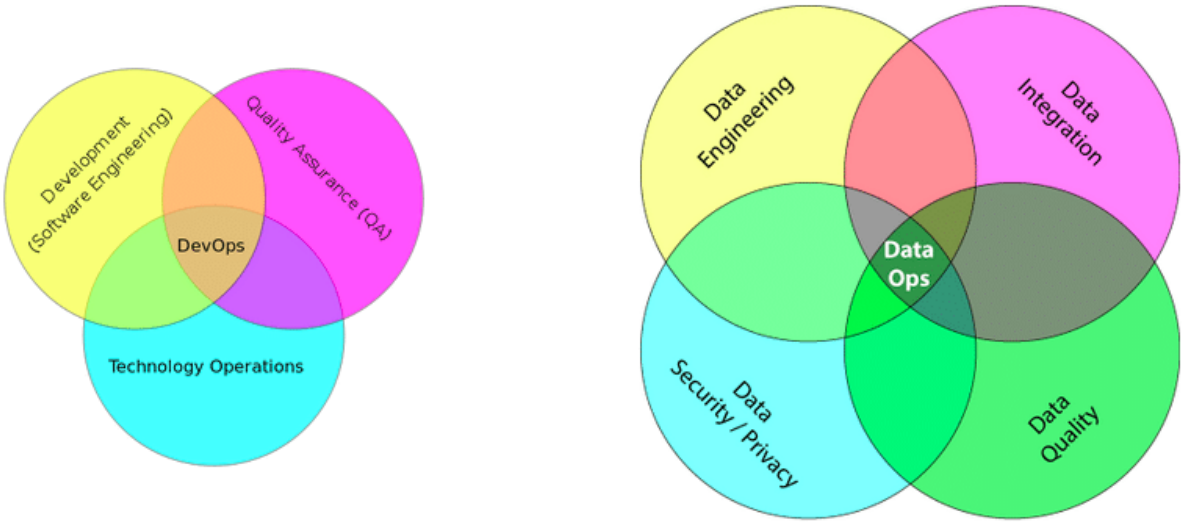


Figure 2 - DevOps in enterprise vs DataOps in enterprise [41]

To support DataOps and automate its processes, tooling and technology is also important. In its whitepaper on introducing DevOps methodology and practice, IBM offers the following supporting architecture for DataOps illustrated in Figure 3.

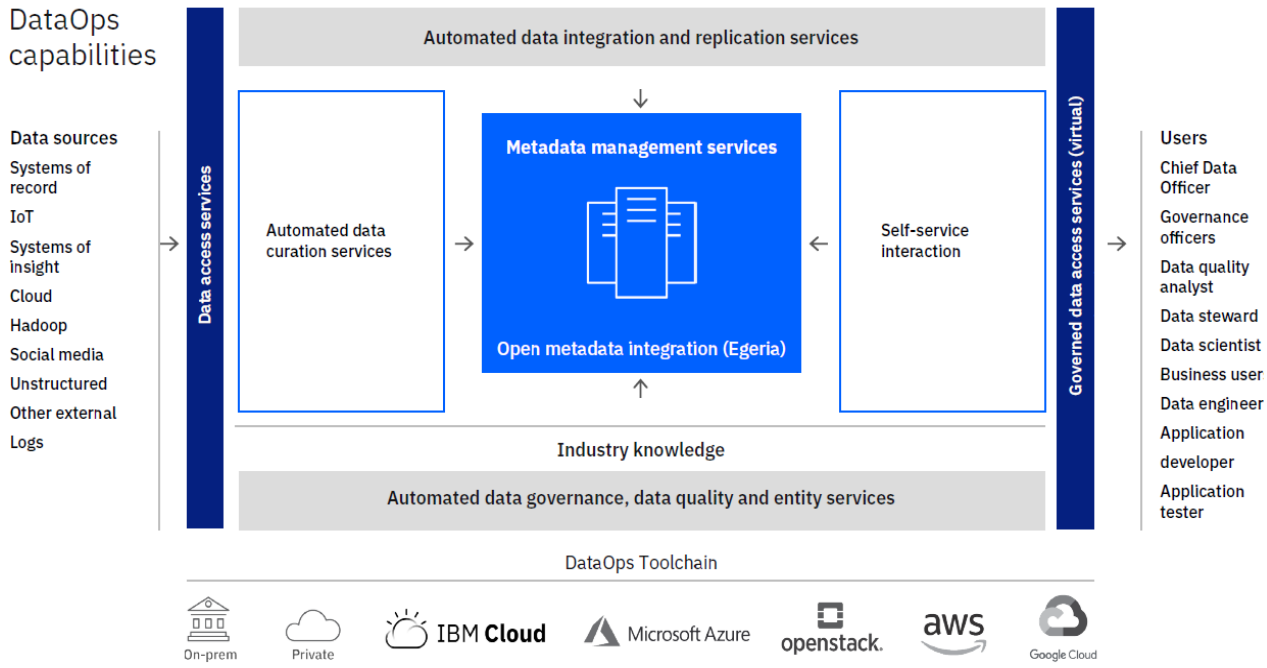


Figure 3 - Supporting Architecture for DataOps [38]

7.2.2 AIOps

AIOps is yet another xOps term that is envisioned to empower software and service engineers to build and operate online services and applications with AI and ML, with the aim of e.g., improving service quality, customer satisfaction and reducing operational cost [42]. AIOps is also referred to as AI for IT operations [43], [44], and was coined by Gartner [45] to address the challenges e.g., scale and complexity of DevOps by using AI.

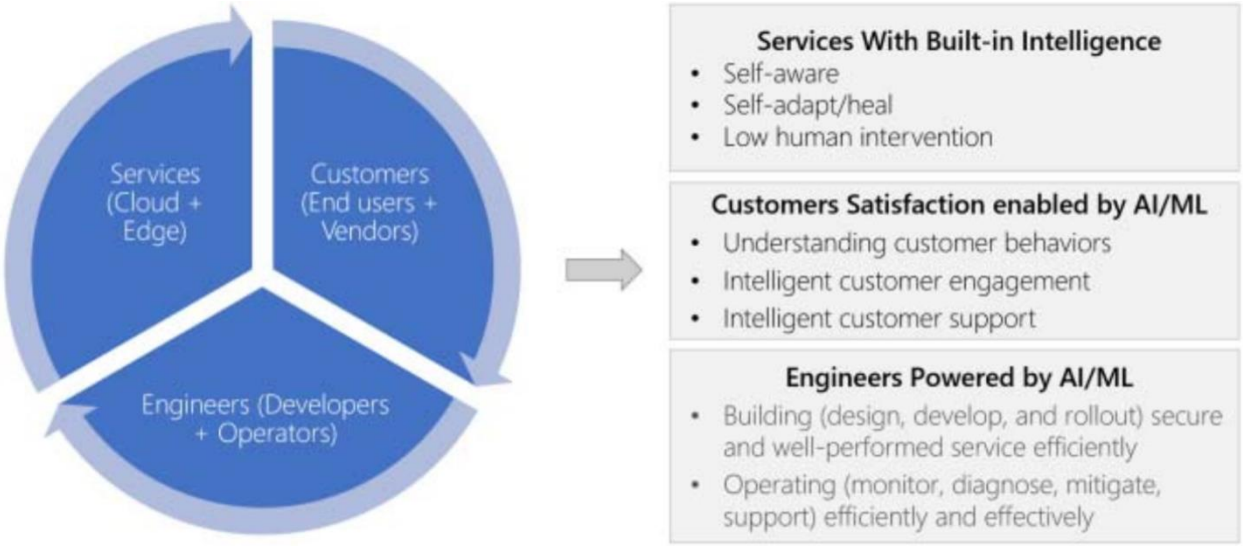


Figure 4 - A generic vision of AIOps [42]

AIOps has currently no agreed-upon definition similar to other recently coined buzz words, hence, may the vision and aims also differ depending on domain, organization and business model. The initial motivation of AIOps was related to ITOps and in its essence aiming at assisting engineering. There are however a set of challenges that needs to be resolved in order to innovate and adopt AIOps solutions [42], such as:

- Resolving gaps in innovation methodologies and mindset, e.g. understanding the whole problem space since guiding innovation methodologies are lacking. In addition is a change in mindset needed, as traditional approaches to e.g., debugging may change from analysing logs and reproducing a bug to automatically learning from history and predicting the future based on patterns
- Engineering practices may need to change, as more data labelling and quality monitoring might also be necessary to serve the needs of AIOps solutions
- Building supervised ML models for AIOps are expected to add additional challenges compared to other ML solutions, since it may be hard to find a clear ground truth

The overall benefits of AIOps are enabling engineers to reduce manual labour by identifying addressing, and resolving issues faster, more specifically [43]:

- Achieving faster mean time to resolution, by reducing noise and correlating data
- Go from reactive to proactive and predictive management, since it keeps on learning
- Modernize operations and the operations team, as filtering and correlations are expected to aid in better and faster corrections

7.2.3 MLOps

MLOps is an engineering practice that is mainly intended to apply DevOps principles to development of machine learning systems and unify the ML development (Dev) and ML system operation (Ops). It involves automation of all steps of ML development including integration, testing, deployment and infrastructure resource management. Building an ML is not a challenge itself, the main challenge is building an integrated ML system and operating it continuously in production. In addition to ML code, the main involved activities in the process of ML system development are data collection, data verification, testing and debugging, resource management, model analysis, process and metadata management, serving infrastructure, and monitoring [46], [47], [48].
Generally, an ML system is a software system, thus applying a DevOps-like practice to guarantee reliable ML building and operating at scale, is essential. In addition to the existing similarities between ML and other typical software systems in CI, and CD, there are a few important differences in MLOps which are as follows [46], [48]:

- CI does not involve only testing ML code and components, but also validating data quality and models.
- CD is not about delivering a software package or service but includes an ML (training) pipeline that automatically deploys another service, i.e., predication service.
- Continuous Training (CT) is a new concept which has emerged in MLOps, and is about automatically retraining and updating the models.

Overall, the main steps of an ML development pipeline could be summarized as follows [48], [49]:

- Data extraction: It involves selecting and extracting data from various sources.
- Data analysis: It includes identifying the data schema, data characteristics and alsothe required data preparation and feature engineering.
- Data preparation: It is about the involved activities in data preparation such as data cleaning, splitting the data into training and validation sets.
- Model training: The ML engineer implements different algorithms to train an ML model with the prepared data. The step might also include hyperparameter tuning in the algorithm.
- Model evaluation: The trained model is evaluated on a validation or holdout test data set.
- Model validation: The performance of the model is validated and confirmed to be good enough to be deployed.
- Model serving: The validated model is deployed to serve the purposes. The deployment can be done in one of the following ways:
    - As a microservice serving the purpose
    - A model embedded in an edge device.
    - A part of a batch system
- Model monitoring: The performance of the model is monitored continuously and in case of performance degradation a new iteration of training might be invoked.

There are three levels of MLOps based on the level of automation in the ML pipeline, which defines the maturity of the ML process. The levels of MLOps are as follows [49]:

MLOps Level 0: A Manual Process

In level 0 although state-of-the-art ML models might be built by ML engineers, the whole process of ML deployment is manual. There is no CI and CD practice and performance monitoring in the process. The ML design and operations stay separated. The data scientists hand over the trained model to the ML engineering team to deploy on their infrastructure and it is assumed that there is no frequent release iterations in the process. Figure 5 shows the steps of MLOps level0.

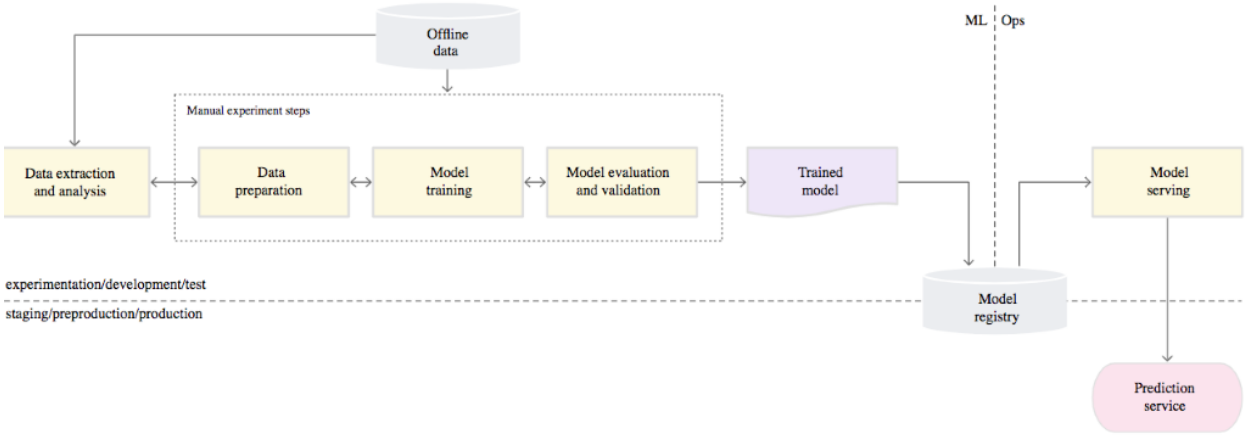


Figure 5 - Manual steps of MLOps level 0 [49]

Currently MLOps level 0 is common in the businesses starting to apply ML to their use cases. The first level of MLOps might be sufficient in the cases where the models are not changed or retrained frequently. However, in many cases of real world the models are required to adapt to changes of the environment or changes in the data. To address these challenges, it is essential to actively monitor the performance of ML models, fre3quently retrain the models upon detecting performance degradation, and also continuously experimenting with new techniques and implementations. Therefore, the MLOps processes including CI/CD and CT practices can play a key role. MLOps level 1 and level 2 are the ones that could be helpful [49].

MLOps level 1: An ML pipeline automation

MLOps level 1 provides continuous training by automating the ML pipeline. It introduces automated data and model validation steps, pipeline triggers and metadata management to MLOps level 0. Rapid experimentation, online CT of the model in the production, continuous delivery of the model on new data, and pipeline deployment instead of model deployment are the special characteristics of MLOps level 1. Figure 2 presents the steps involved in MLOps level 1.

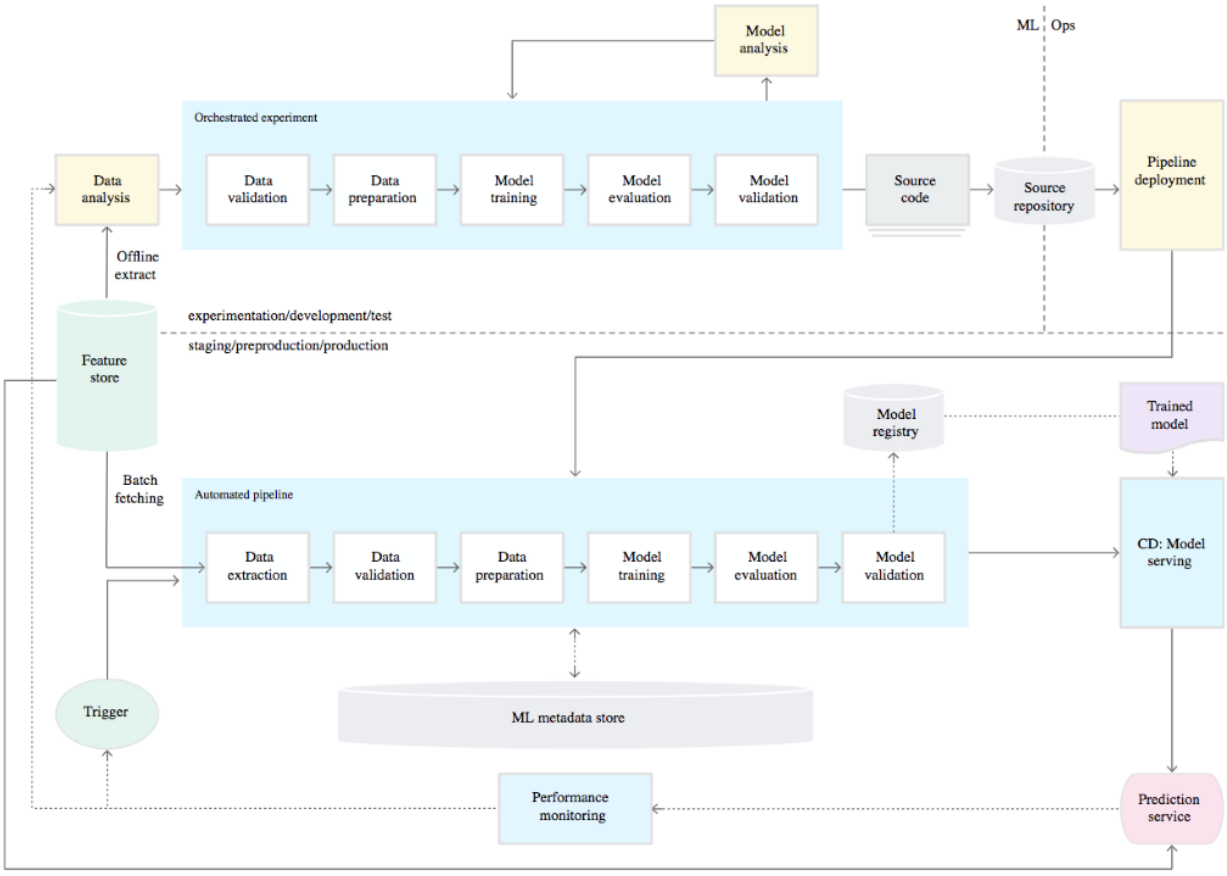


Figure 6 - MLOps level [49]

MLOps level 1 is sufficient, in case new ML implementations and techniques are not frequently deployed and new models are deployed based on new data, rather than new ML techniques.
However, in many businesses it is essential to be able to try new ML ideas and deploy new techniques for ML components rapidly. This is also of importance in case of managing many ML pipelines. To address this challenge, MLOps level 2 provides a CI/CD pipeline automation [49].

MLOps level 2: A CI/CD pipeline automation
MLOps level 2 provides an automated CI/CD process for the pipelines in the production. Following MLOps level 2, the data scientists are able to implement the new ideas, automatically build, test, and deploy in the target environment. It benefits both the ML pipeline automation and automated CI/CD routines.
MLOps level 2 involves the following stages:

- Development and experimentation: The output of this stage is the ML source code that is then stored in source repository.
- Pipeline CI: The outputs of this stage are the pipeline components including packages, executables, and artifacts that are deployed in a later stage.
- Pipeline CD: In this stage the outputs of CI stage are deployed in the target environment.
- Automated triggering: This stage involves automatic execution of the pipeline in the production in response to a trigger.
- Model CD: This stage deploys the trained model.
- Monitoring: This stage monitors the model performance on live data. The output is a trigger to execute the pipeline or execute a new experiment iteration.

Figure 6 shows the stages and the involved components in MLOps level 2 [49].

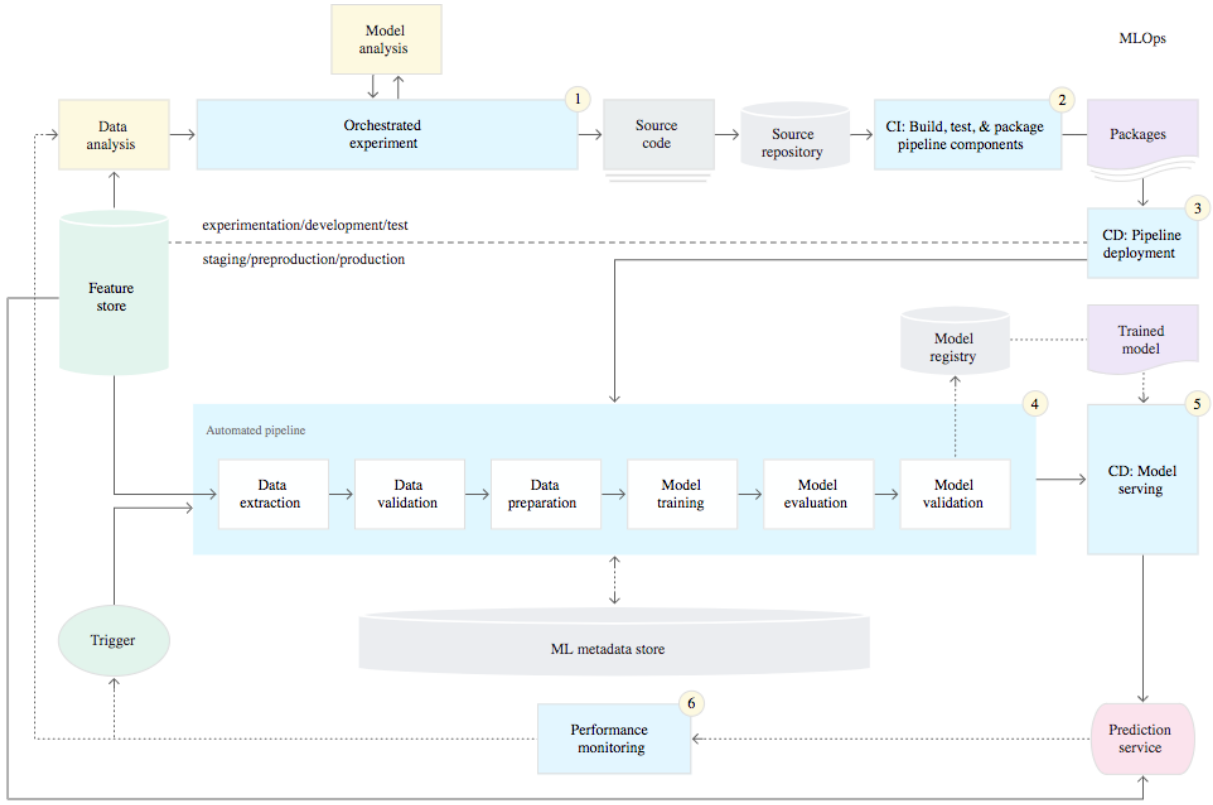


Figure 7 - MLOps level 1 [49]

MLOps vs AIOps

At first glance the terms MLOps and AIOps may sound synonym, and in some places and contexts they may be considered interchangeable, or even MLOps might be considered as a subset of AIOps. However, there are key differences in the way they are defined and the intention behind each. In simple terms, AIOps is about bringing in AI/ML techniques to solve Ops challenges and application of AI/ML in IT processes, while MLOps is more on the application DevOps principles for development of ML-based systems and the process of bringing machine learning to production [50].

8. Predictive Maintenance

Predictive maintenance is the process of collecting real life operational data, analyzing them in order to predict future failures and then planning maintenance accordingly to prevent those failures from happening. Predictive maintenance differs from preventive maintenance because it relies on the actual condition of the equipment, rather than average or expected life statistics, to predict when maintenance will be required. Condition-based maintenance is an extension of predictive maintenance, during which data from sensors is being analyzed in real time and alarms are triggered if the algorithm predicts an upcoming failure [51]. All three types of maintenance aim to achieve higher up-time and more efficient and less costly maintenance.

Some of the main components that are necessary for implementing predictive maintenance are data collection and pre-processing, early fault detection, fault detection, time to failure prediction, maintenance scheduling and resource optimization [51]. Predictive maintenance has also been considered to be one of the driving forces for improving productivity and one of the ways to achieve "just-in-time" in manufacturing.

8.1 ML-enabled techniques applied to Predictive Maintenance

The growing availability of data sources (Source Code, Test plan and test reports, Logs, traces and monitoring data, Human feedback…) is enabling timely and accurate prediction of maintenance requirements in industrial sector. Multiple classifiers in supervised learning have been used to solve imbalanced data sets; predicting frequency of unexpected breaks and also to exploit heterogenous data (as cutting forces, vibration signals and acoustic emissions). The dynamic nature of systems, and the fact that operational data change constantly, may be solved by applying the Diversity for Dealing with Drift [52].

Unsupervised learning, based on the ingestion of structured and unstructured data is being widely used for predictive maintenance. The combination of big data analysis and data fusion for predictive maintenance has guided research activities in the application of Machine Learning to almost all sectors. Data-driven techniques mainly focused in the wider availability of time series, combining different clustering and recurring neural networks have successfully provided predictive maintenance solutions. Below as follows, a table-summary of an exhaustive survey provides a reference of ML models and applications in industry [53]:

Reference	ML Solution
Susto G.A., Schirru A., Pampuri S., McLoone S., Beghi A. Machine Learning for Predictive Maintenance: A Multiple Classifier Approach. IEEE Trans. Ind. Inf. 2015;11:812–820. doi: 10.1109/TII.2014.2349359	MC supervised method
Yan J., Meng Y., Lu L., Li L. Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance. IEEE Access. 2017;5:23484–23491. doi: 10.1109/ACCESS.2017.2765544	Multi-scale analysis (envelope, time-frequency)
Wu D., Jennings C., Terpenny J., Gao R.X., Kumara S. A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests. ASME J. Manuf. Sci. Eng. 2017;139:071018. doi: 10.1115/1.4036350	RandF
Shin H.J., Cho K.W., Oh C.H. SVM-Based Dynamic Reconfiguration CPS for Manufacturing System in Industry 4.0. Wirel. Commun. Mob. Comput. 2018;2018:5795037. doi: 10.1155/2018/5795037.	SVM
Kuo C.J., Ting K.C., Chen Y.C., Yang D.L., Chen H.M. Automatic Machine Status Prediction in the Era of Industry 4.0: Case Study of Machines in a Spring Factory. J. Syst. Archit. 2017;81:44–53. doi: 10.1016/j.sysarc.2017.10.007	NN-based for online feature dimensionality

	reduction and automated prediction
Lin C., Shu L., Deng D., Yeh T., Chen Y., Hsieh H. A MapReduce-Based Ensemble Learning Method with Multiple Classifier Types and Diversity for Condition-based Maintenance with Concept Drifts. IEEE Cloud Comput. 2017;4:38–48. doi: 10.1109/MCC.2018.1081065.	Ensemble learning with MC types and diversity
Yu W., Dillon T.S., Mostafa F., Rahayu W., Liu Y. A Global Manufacturing Big Data Ecosystem for Fault Detection in Predictive Maintenance. IEEE Trans. Ind. Inf. 2019 doi: 10.1109/TII.2019.2915846.	K-means, DPCA-based T-squared and SPE
Peres R.S., Rocha A.D., Leitao P., Barata J. IDARTS—Towards Intelligent Data Analysis and Real-Time Supervision for Industry 4.0. Comput. Ind. 2018;101:138–146. doi: 10.1016/j.compind.2018.07.004.	K-means
Yan H., Wan J., Zhang C., Tang S., Hua A., Wang Z. Industrial Big Data Analytics for Prediction of Remaining Useful Life Based on Deep Learning. IEEE Access. 2018;6:17190–17197. doi: 10.1109/ACCESS.2018.2809681Yan et al.,	DL-based DECG
Sun C., Ma M., Zhao Z., Tian S., Yan R., Chen X. Deep Transfer Learning Based on Sparse Autoencoder for Remaining Useful Life Prediction of Tool in Manufacturing. IEEE Trans. Ind. Inf. 2019;15:2416–2425. doi: 10.1109/TII.2018.2881543	DTL with SAE
Cheng Y., Zhu H., Wu J., Shao X. Machine Health Monitoring Using Adaptive Kernel Spectral Clustering and Deep Long Short-Term Memory Recurrent Neural Networks. IEEE Trans. Ind. Inf. 2019;15:987–997. doi: 10.1109/TII.2018.2866549.	AKSC with LSTM-RNN
Panoutsos G., Luo B., Liu H., Li B., Lin X. Using Multiple-Feature-Spaces-Based Deep Learning for Tool Condition Monitoring in Ultra-Precision Manufacturing. IEEE Trans. Ind. Inf. 2019;66:3794–3803. doi: 10.1109/TIE.2018.2856193	Feature spaces-based DL

9. Model Synthesis and Construction

Model synthesis is the process of generating structural or behavioural models of the system from the input artefacts. The input artefacts can vary from natural language requirements, to models at other levels of abstraction, and to source code. In this section, we provided an overview of model synthesis approaches available in the literature.

Reider *et al.* Presented an approach for feature testing and similarity-based test case prioritization [54]. The presented approach relies on formal requirements, written in RDL. A UML model is synthesized from the RDL specification and test cases are generated and prioritized for feature testing.

Whittle and Schumann proposed an algorithm for synthesizing UML state charts from a set of UML sequence diagrams [55]. The algorithm generates the state charts by merging the similar behaviour in the given sequence diagrams and detects conflicts while merging.

GK-Tail was proposed to generate finite state machine models using interaction traces of the system [56]. GK-Tail also generates condition on the data values for the edges of the finite state machine.

Prähofer *et al.* used execution traces to synthesize a high-level behavioural model of PLC programs [57]. The synthesized model represents the timing and transition behaviours of the PLC component. The synthesized model can then be used to check the conformance of future executions.

Feature model synthesis is also an open area of research. Most of feature model synthesis approaches look for commonalities and variability to suggest features and extract feature models from the input. In some cases, public documents (such as brochures and reviews) are being used for mining common domain terminologies and their variability [58], [59]. These approaches focus on aggregating natural language requirements to extract a high-level system feature model. Such approaches help in the FODA [60] and helps the industry in the adoption of product line engineering.

Arborcraft [61] uses LSA to calculate the similarity between requirements pairs. Arborcraft then clusters the requirements based on similarity (shared concepts in the requirements) and extract feature tree from the clustered requirements. Arborcraft uses a fuzzy feature diagram which allows the features to be in more than one sub-tree. This means that the feature can be part of more than one cluster, to deal with the imprecise nature of natural language.

The Semantic and Ontological Variability Analysis [62] uses semantic role labelling to calculate similarity-based semantic roles. The roles extracted in the approach reflect behavioural information and bases similarity metrics on behaviour. The approach uses a hierarchical clustering algorithm to cluster the requirements based on behavioural similarity. The approach then generates a feature model based on the extracted similarities and variabilities.

Bottom-up technologies for reuse (BUT4Reuse [63]) is another tooled approach for extractive adoption of SPL. BUT4Reuse work with variety of artefacts for variability analysis (such as source code, and models etc.) to reverse engineer the variability (feature models).

Automated Machine Learning (AutoML)

Automated machine learning (AutoML) systems are used to find the best machine learning (ML) pipeline matching the task and data at hand, typically classification or regression. This includes model selection and hyperparameter optimization. Finding good models and hyperparameters are hard and time-consuming tasks for human experts, and they frequently involve a lot of trial-and-error experimentation. The promise of AutoML is that computers can automate these repetitive tasks and come up with good pipelines with little human effort. The drawback is that AutoML systems require a lot of computing power and the quality of the results varies. A recent overview of different AutoML systems [64] echoes these issues.

AutoML systems are meta-level machine learning algorithms, which use other ML solutions as building blocks for finding the optimal ML pipeline. In this context, an ML pipeline means the set of algorithms and their hyperparameters that the ML system uses to infer results from data. An AutoML system has to consider multiple ML pipelines and search values for their parameters. It needs to optimize each candidate pipeline to an adequate level but also ensure that enough time and resources are used to experiment with alternative pipelines. As a result, using AutoML systems can consume huge amounts of time and computing resources.

There is also the question of how much automation you want from the AutoML system. While an experienced data scientist would know, for example, the optimal hyperparameter spaces to search, an average user with little ML experience would want good results with minimal configuration. Many tools support both groups -- they allow heavy customization, but are easy although less efficient to use with the default settings. It is also known that the systems can recommend different optimal solutions for the same problem [65]. This can be caused by tight time limits imposed on the system disabling further optimization or that two pipelines offer equal performance. In this study, we took the approach of using the systems with default settings

Typical tasks that many AutoML systems support are classification and regression. In various examples and benchmarks, typically image or text data are used. Some AutoML system like AutoKeras [66] even offer specialized image and text classifiers. This is because images are typically represented as pixel arrays and in case of grayscale images the value of each pixel can be represented with an 8-bit integer. Pretty much all classification systems support this kind of input out of the box, and even in the worst-case scenario, type conversion and array reshaping should be all the required pre-processing steps.

10. Fault Prediction

A fault is a problem in a system that may cause a failure. A failure occurs as long as a system is not delivering a service consistent with its specifications. A fault can potentially cause a failure.

Many of the tactics to keep faults from becoming failures -or mitigating the effects-, are available within standard execution environments. All approaches for fault management involve a combination of techniques for redundancy, health monitoring, and recovery when a failure is detected.

High Availability

Other related concept is HA. It refers to systems that are durable and likely to operate continuously, without failure, for extended periods of time. HA is mainly measured as the up time per year.

SLA is the common formal model to define the contract between customers and providers with respect to HA. In this agreement, key aspects as quality, availability, responsibilities are agreed between the parties, and usually it is linked to economical and legal consequences. The target availability percentages are usually defined by the number of 9s: 90.0% -one 9- to 99.9999% -six 9s, for the maximum downtime accepter per year, month, week, day…

Fault prevention

Beside the traditional techniques for fault detection and recovery, fault prevention is a key topic. In mission-critical systems, is a fundamental concept. Also, in industrial environments, Fatigue prediction has also been an area of increasing research. Gaunaru Ana, et al., [67] present an overview of failures. They define "Fatigue" of a component as a result of cyclic stress. Three phases are related to fatigue failures: initiation, propagation, and catastrophic overload failure. The duration of every phase is impacted by many potential variables: raw material, stress applied, processing history... All industrial sectors work in different techniques for detecting critical situation in advance, to trigger preventive measures and mitigate the effect of a failure. Two levels related to online failures prediction are present in literature. One of them is focusing in observing components and the other is focused in analysing different systems parameters.

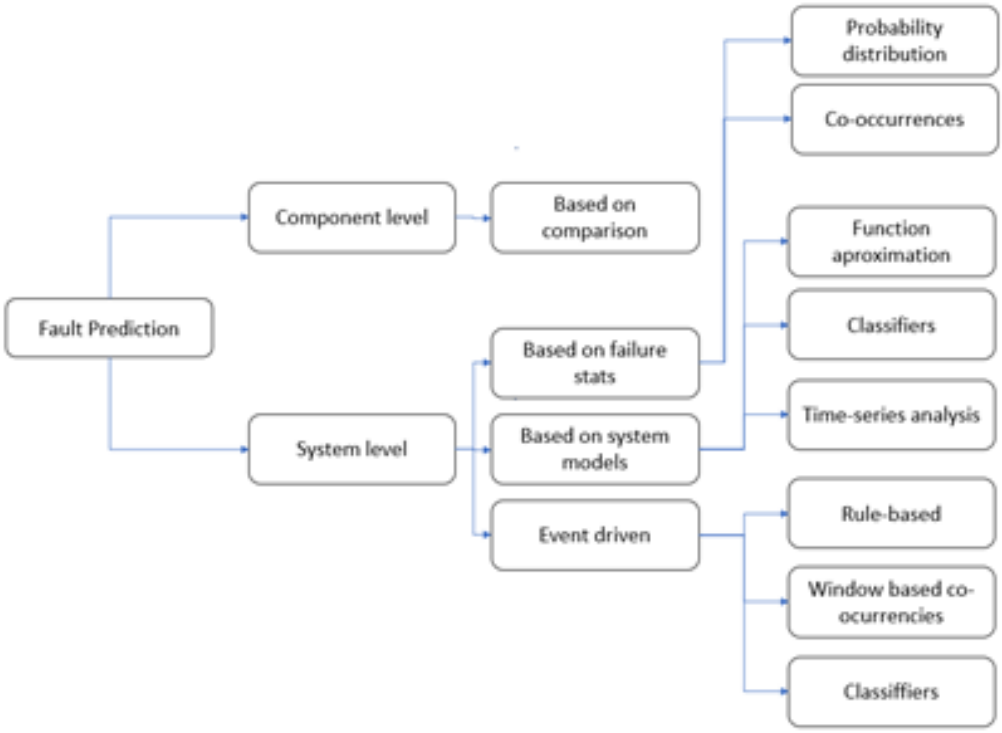


Figure 8 - Fault Prediction levels

In recent years, the advent of the DevOps approach, and more recently AIOps, have brought new techniques to the fore, that are taking into account the whole lifecycle to find correlations and behavioural patterns.

10.1 ML-based Fault Prediction Techniques

Relevant surveys and approaches related to the application of ML for detection of faults in Industry 4.0 have been published during the last years. Angelopoulos et al. [53] summarized recent approaches and solutions, highlighting the need to timely retrieve and process data from monitoring systems in order to correctly detect abnormal operation and fault. Fault prediction usually involves three steps, that is, data collection, data processing for feature extraction and finally, fault classification.

Relevant solutions for fault detection, based on supervised, unsupervised and deep learning methods can be found in the next table:

Reference	ML Solution
Lee T., Lee K.B., Kim C.O. Performance of Machine Learning Algorithms for Class-Imbalanced Process Fault Detection Problems. IEEE Trans. Semicond. Manuf. 2016;29:436–445. doi: 10.1109/TSM.2016.2602226.	ANNs, SVMs and WMV
Jin S., Ye F., Zhang Z., Chakrabarty K., Gu X. Efficient Board-Level Functional Fault Diagnosis With Missing Syndromes. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2016;35:985–998. doi: 10.1109/TCAD.2015.2481859.	SVM, ANN, Naive Bayes, and Decision Tree
Mathew J., Pang C.K., Luo M., Leong W.H. Classification of Imbalanced Data by Oversampling in Kernel Space of Support Vector Machines. IEEE Trans. Neural Netw. Learn. Syst. 2018;29:4065–4076. doi: 10.1109/TNNLS.2017.2751612	WK-SMOTE SVM
Lin C., Deng D., Kuo C., Chen L. Concept Drift Detection and Adaption in Big Imbalance Industrial IoT Data Using an Ensemble Learning Method of Offline Classifiers. IEEE Access. 2019;7:56198–56207. doi: 10.1109/ACCESS.2019.2912631	Ensemble learning with various offline classifiers
Lee T., Lee K.B., Kim C.O. Performance of Machine Learning Algorithms for Class-Imbalanced Process Fault Detection Problems. IEEE Trans. Semicond. Manuf. 2016;29:436–445. doi: 10.1109/TSM.2016.2602226	Comparison of three sampling-based, four ensemble, four instance-based, and two SVM methods
Syafrudin M., Alfian G., Fitriyani N.L., Rhee J. Performance Analysis of IoT-Based Sensor, Big Data Processing, and Machine Learning Model for Real-Time Monitoring System in Automotive Manufacturing. Sensors. 2018;18:2946. doi: 10.3390/s18092946.	DBSCAN-based RandF
Lei Y., Jia F., Lin J., Xing S., Ding S.X. An Intelligent Fault Diagnosis Method Using Unsupervised Feature Learning Towards Mechanical Big Data. IEEE Trans. Ind. Electron. 2016;63:3137–3147. doi: 10.1109/TIE.2016.2519325	Two-stage NN with sparse filtering and softmax regression
Yang Z.-X., Wang X.-B., Zhong J.-H. Representational Learning for Fault Diagnosis of Wind Turbine Equipment: A Multi-Layered Extreme Learning Machines Approach. Energies. 2016;9:379. doi: 10.3390/en9060379.	Multiple hierarchical ELMs
Diaz-Rozo J., Bielza C., Larrañaga P. Machine Learning-Based CPS for Clustering High Throughput Machining Cycle Conditions. Procedia Manuf. 2017;10:997–1008. doi: 10.1016/j.promfg.2017.07.091.	K-means, hierarchical, agglo-merative and Gaussian mixture

Pan J., Zi Y., Chen J., Zhou Z., Wang B. LiftingNet: A Novel Deep Learning Network With Layerwise Feature Learning From Noisy Mechanical Data for Fault Classification. IEEE Trans. Ind. Electron. 2018;65:4973–4982. doi: 10.1109/TIE.2017.2767540.	DL-based LiftingNet
Sohaib M., Kim C.H., Kim J.M. A Hybrid Feature Model and Deep-Learning-Based Bearing Fault Diagnosis. Sensors. 2017;17:2876. doi: 10.3390/s17122876.	SAE-based DNNs.
Luo B., Wang H., Liu H., Li B., Peng F. Early Fault Detection of Machine Tools Based on Deep Learning and Dynamic Identification. IEEE Trans. Ind. Electron. 2019;66:509–518. doi: 10.1109/TIE.2018.2807414	DL-based dynamic properties extraction
Tao Y., Wang X., Sánchez R., Yang S., Bai Y. Spur Gear Fault Diagnosis Using a Multilayer Gated Recurrent Unit Approach With Vibration Signal. IEEE Access. 2019;7:56880–56889. doi: 10.1109/ACCESS.2019.2914181.	MGRU-based NN
Wen L., Gao L., Li X. A New Snapshot Ensemble Convolutional Neural Network for Fault Diagnosis. IEEE Access. 2019;7:32037–32047. doi: 10.1109/ACCESS.2019.2903295	SECNN with MMCCLR
Iqbal R., Maniak T., Doctor F., Karyotis C. Fault Detection and Isolation in Industrial Processes Using Deep Learning Approaches. IEEE Trans. Ind. Inf. 2019;15:3077–3084. doi: 10.1109/TII.2019.2902274.	DL-based FDI with DAEs

10.2 ML-based Fault Prediction for Evolving Systems

Classical verification and validation approaches that have been subject of research in computer science for a long time like testing, model checking and theorem proving are generally limited when applied to evolving systems. Fault Prediction for ML-based Evolving Systems is its very early stages.

In software-intensive systems, traditionally source code has been considered as the main artefact for analysis and extraction of metrics for different purposes such as fault prediction, product quality evaluation, debugging and fault localization, certification, and taking maintenance decisions. However, as the size and complexity of systems have grown, the role of analytics over other development artefacts has gained attention as well to provide more accurate insights on the behavior of a software system, that is tightly related to fault prediction. In this context, the big data analytics has come to light, and has derived in several research and engineering challenges [68] such as: the issue of outdated data and providing real time and actionable insights, sharing of information and insights, privacy issues, hardware platforms for big data processing, instrumentation mechanisms, probes and sensors for collecting relevant data. Additionally, traceability of artefacts and data elements [69], transforming raw data and extraction of patterns are also of significant importance to provide accurate insights into the behavior of a system, that could be the reference for fault prediction. The link between traceability and fault prediction must be deeper explored. A commonly agreed upon definition of software traceability is “the ability to interrelate any uniquely identifiable software engineering artefact to any other, maintain required links over time, and use the resulting network to answer questions of both the software product and its development process”, and was first formulated by the CoEST.

OSLC is an initiative in the form of an open community developing standards for integrating tools and making it easy and practical for software lifecycle tools to share data with one another using traceability links, this could be the basis for fault prediction in ES. The references discuss [70] the importance of traceability in achieving continuous integration and delivery, and also introduce Eiffel as a framework to provide real time traceability. Eiffel was developed by Ericsson to address the challenges of scalability and traceability. A reference regarding the adoption of Eiffel protocol, and the technical and organization needs and obstacles has been described by Hramyka and Winsqvist [71], remarking the best practices and recommendations. Bathae, Yavar described the main challenges related to the black box problem related to Machine Learning [72].

A very interesting concept was described by Jeffrey Mogul [73], suggesting potential mitigation of emergent misbehaviours in complex systems. Mogul stated that emergent misbehavior might often not be the result of component failure, traditional failure prediction techniques, such as those based on MTBFs or fault trees, might be inapplicable [73]. MTBF data would only be useful if system-wide failures were primarily caused by component failures. In this context, a potential approach would be the creation of a corpus of "signatures" based on observed events leading up to detected emergent misbehaviour in real systems. The detection of these signatures in a running system could be used to indicate that potentially, a misbehaviour may occur. A taxonomy of causes should guide the creation of signatures.

In any case, the black-box problem, related to ML-enabled will likely lead fault prediction to be based on a combination of big data analytics and a regulatory process to take into account degree of the AI's transparency as well as the extent to which the AI should be supervised by humans. Hence, ES for security markets may have lower liability thresholds that mission critical system.

11. Change Impact Analysis

Change impact analysis (CIA) is used to reason about effects. That is, 'the process of identifying the potential consequences of a change, or estimating what needs to be modified to accomplish a change' [74].

Kilpinen classified the CIA approaches into three different classes, namely TIA, DIA, and EIA [75]. TIA approaches utilize cross-abstraction-level traceability links to identify and propagate the impact of a change in an artifact or entity. An extensive review of the role of traceability in impact analysis can be found in the literature [76]. DIA approaches are focused on identifying the impact of a change using the dependencies among software artifacts (e.g., class dependencies). DIA approaches are more focused on source code-level and use program slicing and static program analysis (e.g., [77]). However, the same approaches can be applied to other levels of abstractions. Finally, EIA approaches use manual reviews and code inspection to find the impact of changes on other software artifacts. EIA approaches are dependent on the experience of practitioners and involve substantial manual efforts. Studies in the literature show that EIA approaches might detect fewer impacts than other classes of CIA approaches [75].

The CIA approaches can support the impact analysis at different levels of abstractions (e.g., requirements, models, source code). In the rest of this section, we provide examples of CIA approaches focused on different levels of abstractions.

Requirements-level CIA

Arora *et al.* proposed an approach for the CIA for natural language requirements [78]. This uses NLP techniques to identify inter-requirements impact resulting from a change in requirement. In some cases, the requirements might be written in a structured requirement modeling notation. Alkaf *et al.* Proposed an automated approach for CIA for URN models [79]. This automated CIA approach helps in identifying the impacted URN elements by a change in specification.

One specific topic within Requirements-level CIA, based on the combination of ML-based systems (mainly NLP) and Regulatory Technology is currently driving research efforts in fintech sector. The growing regulation -as "Dodd-Frank", the second "Markets in Financial Instruments Directive" and the revised "Payment Services Directive" – as well as the GDPR and the associated growing costs for compliance have deep implications in the application of ML-based evolving systems in finance sector. The application of Requirement-level CIA, when applied to ML-enabled Evolving Systems for fintech environment, should take into any change in compliance and regulatory framework. Some solutions that automatically search, monitor and track regulatory content and APIs are available: compliance.ai and cube.global are key references.

Model-level CIA

The research of Piotr is focussed on the effect of changes in effect of changes in class diagrams to the structure of state machines [80]. To do so, the UML and OCL are used. It shows how the preservation of logical formulas is corresponding to the structure of state machines.

Source code-level CIA. Several other studies on change impact analysis were conducted. The major part is focussed on source code. We reviewed multiple (approaches to) change impact analyses. An example of a change impact analysis is found in the prediction of impact of software changes on build times in a continuous integration (CI) environment [81]. Adding dependencies on modules can increase build times and therefore slow down releases. The impact is measured as the product of properties that are deltas in build times, a longest critical path and dependency graphs. Its purpose is to inform developers by stating the chances that their change may influence future build times. Shari Lawrence and Joanne M. describes change impact analysis as 'the evaluation of many risks associated with the change, including estimates of effects on resources, effort and schedule' [82]. Costs are mentioned explicitly as motivation to do an analysis as it can help to gain control while doing maintenance.

The discussed research shows several forms of impact and related metrics. Each change impact analysis is equal by the interest in a certain effect on a particular object. Build times, (relations between) affected objects and property preservation were used to analyse impact.

Test case-level CIA

CIA approaches on the test-case level are generally focused on finding the impact of a change on the test suite. Most of the approaches in this category classify test cases into broken, repairable, and executable test cases based on a change in source code of a system. Many tools and approaches have also been proposed to automatically evolve test cases classified as repairable. An extensive review of these approaches can be found in the literature [83].

Praegus, with its extensive experience in testing, test automation and the application of artificial intelligence and machine learning in this domain, offers so-called 'Augmented Testing' within the IVVES project as an accelerator within a CI / CI pipeline to accelerate (automatic) feedback and a self-learning system for prioritizing tests. From an integrated dashboard, it will provide teams and organizations with insight into all the tests they perform, to make the test automation process more efficient. From this dashboard, it will be possible to collect all test results in one place, which provides a single overview of all different types of test (such as unit, integration, system and E2E-tests). All test runs of all teams within an organization can be viewed at a glance. In addition, this also provides insight into the test history and provides detailed test information for each test. Besides this, it will be also visible (per test set) who has tested what, where, so that everyone within the team can view each other's results. Everything in real time.

The system will be self-learning, with the ability to learn from previous test results. The system recognizes trends in failures and will perform auto-analyses, reducing the time required to manually analyse test results. This means that error analyses take less time. With 'Augmented Testing' the system learns to recognize which tests belong to which code independently searches for the correlation between previously checked code and executed tests. With a code change, the system searches only for the tests that are relevant to this specific code change, so that the test time can be significantly reduced without compromising on quality requirements. So only to test what is needed, resulting in a faster release.

Test automation is of course essential in this process. Once test automation has started, the number of automated tests often only increases. When tests are automated by different teams with different tools, it becomes increasingly difficult to keep an overview and to keep the test set effective. By collecting all test results from all tools in the system, the dashboard provides insight into the quality of the underlying test set. In this way one does get a view of the total quality effort again. Testers spend a lot of time in their daily work figuring out why tests fail. This search time is reduced by the system by means of the auto analysis. This gives testers more room to spend their time even more effectively. To be able to go live quickly in a continuous delivery environment, a regression test that takes, for example, 8 hours is not ideal. However, is every test in that regression set really necessary? In the event of minor changes, you do not end up hitting the entire system. Manually prioritizing tests is no longer an option in a continuous delivery environment. Based on previous test runs, 'Augmented Testing' learns which tests are relevant when and optimizes your test set.

During the initiation of the IVVES project, when when preparing the Full Project Proposal and Project Plan, no such solution existed! In the meantime, we see various initiatives starting up, including Facebook [84] and Launchable (https://launchableinc.com), confirming that we are on the right track and with IVVES can lead the way with this innovation.

12. Data collection in Exploratory Testing

Exploratory testing (ET) – simultaneous learning, test design, and test execution – is an applied practice in industry but it has received less attention in the research context. In the scope of project IVVES, there are two primary directions to investigate in the field of data collection using exploratory testing:

- Data collection during manual use of the system.
- Data collection during automated system/program exploration, and

Itkonen and Rautiainen have identified five properties that describe when testing is exploratory testing [85]:

1. Tests are not defined in advance as detailed test scripts or test cases. Instead, exploratory testing is exploration with a general mission without specific step-by-step instructions on how to accomplish the mission.
2. Exploratory testing is guided by the results of previously performed tests and the gained knowledge from them. An exploratory tester uses any available information of the target of testing, for example a requirements document, a user's manual, or even a marketing brochure.
3. The focus in exploratory testing is on finding defects by exploration, instead of systematically producing a comprehensive set of test cases for later use.
4. Exploratory testing is simultaneous learning of the system under test, test design, and test execution.
5. The effectiveness of the testing relies on the tester's knowledge, skills, and experience.

Hence, it can partly be regarded that ET is not only an activity but also a mindset that the testers adopt while performing the actual testing.

TESTAR (www.testar.org) is an open source tool for scriptless test automation through GUI. By scriptless, it is meant that the tool generates the test sequences in an exploratory way, one step at a time during test execution, based on observed state of the SUT. Without any system specific configuration, with its default settings, TESTAR can test a system as a monkey testing tool, randomly clicking the available clickable widgets, and typing pseudo-random input string into the input fields of the SUT. In the IVVES project, the TESTAR tool will be extended with AI-based action selection strategies and active learning of state models during the automated exploration of the SUT. In addition, the goal is to capture actions of a human user into the state model for learning and evaluating manual exploratory testing.

Data collection during manual use of the system

In general, data collection during automated system/program exploration can utilise the prime data sources listed in above in previous sections. In addition, and probably more importantly, as ET is partly guided intuition and experience of the testers, monitoring tools can be added in the process so that the same exploratory procedures can be executed later on either to reproduce the same errors or to check that the program has been fixed.

As for implementing the necessary analytics and monitoring procedures, and resorting to real usage based testing, the reader is referred to IVVES deliverable D3.1 *State of the art of validation methods and techniques on validation techniques for complex evolving systems*, sections 3.3.1 and 3.3.3 respectively, which provide an extended discussion on the topics.

Data collection during automated system/program exploration

While exploratory testing is most commonly associated with human testers, there are use cases where automated system/program exploration is desired. In particular, since continuous software development, manifested in e.g. DevOps, aims at automating all the steps that are needed for a successful deployment, it is often necessary to replace human testers with automation and robots.

Approaches that can be used for automated system/program exploration are many. In the simplest case, one can take the system under test, its previous testware, and use the version history to determine which test cases are potential subjects for exploratory testing. Furthermore, techniques that are possible to apply when gathering input for automated system/program exploration mainly consists of monitoring the behaviour of human testers, as well as performing source code analysis for the system under test.

As for support for automated test generation, selection and prioritization, the reader is referred to IVVES deliverable D3.1 *State of the art of validation methods and techniques on validation techniques for complex evolving systems*, sections 3.2.4 and 3.2.5 respectively, which provide an extended discussion on the topics.

13. Root Cause Analysis of failures

The traditional practice of root cause analysis (RCA) is a form of deductive analysis, working like a detective, beginning with a known problem and working backward, sifting through the available evidence to identify the culprit.

First, we identify the problem, establishing a mental timeline from where the happy flow of the application was interrupted and a problem occurred. Then we correlate different events (in today's application landscape this will often cross different processes, all interconnected with e.g. RESTful API's). Given all this input we try to compose a corrective action to prevent the problem from reoccuring.

We can identify two types of RCA: 1) The RCA of test failures, like failing automated tests in a software delivery pipeline, and 2) Failures that occur in a production-like environment.

Interesting to note is that the classical dichotomy between 'Test' and 'Production' starts to diminish by new trends like Chaos Engineering [86], [87]. Chaos Engineering practices apply changes (e.g. killing a pod or node inside a kubernetes cluster) to see if the production system is resilient enough to auto-repair itself.

13.1 RCA of failures in a production environment

In order to perform RCA of a production failure we have first to identify the problem. Then we try to find as much data as possible to find what caused it. In order to identify a problem, we can identify two types of monitoring: **Black Box** monitoring and **White Box** monitoring. With Black Box monitoring we use the system through the same means as an end user i.e. a web application can be tested by instrumenting a browser, and an API can be monitored by using the API itself. This can be used to *identify* a failure but does not help much with the RCA of the failure. That is where White Box monitoring comes in. With White Box monitoring we mean the idea of exposing the internals of the system, like a software application, by letting each component log what is going on. Obviously, this is an important thing to have in place in order to perform the RCA of failures.

Using structured logs

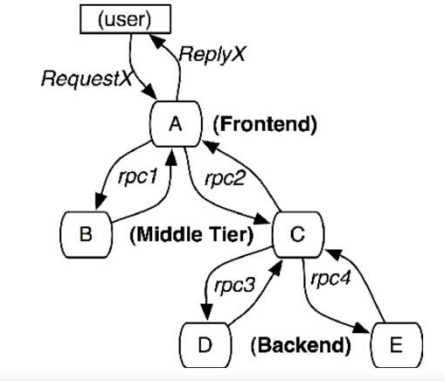
A common industry standard is to use structured logs as main audit trail for figuring out what is going on inside our production software deployments. But in complex application landscapes comprising of dozens of microservices parsing all this data, and making sense of it, becomes a time-consuming task. In this case anything that can speed up the RCA of a failing component is obviously very valuable.

Using distributed tracing and metrics

In addition to logs it is becoming more and more popular to apply distributed tracing. Where logging provides an overview to a discrete, event-triggered log, tracing encompasses a much wider, continuous view of an application. The goal of tracing is to follow a program's flow and data progression. By tracing through a stack, developers can identify bottlenecks Tracing can help by automatically give insight in the runtime dynamics of an application. Examples of tools to make tracing easier are Dapper [88], Zipkin[1] and Micrometer[2]. These tools all want to help to give insight in flows like the following:

[1] https://zipkin.io

[2] http://micrometer.io/



Where it is important to spot bottlenecks and sources of errors.

Making sense of it

In order to perform RCA we need data. As described earlier in this document we have the availability of logs, traces and monitoring data. A common industry best practice is to collect all this logdata into an indexed cluster. Splunk and the Elasticsearch-stack are examples of this. using tools like Logstash. The Elasticsearch-stack is commonly refered to as the ELK-stack[3]. For a good introduction of these tools, Atlassian has given a good presentation about this topic with the name *Find the root cause*[4]. This takes care of tokenizing (grokking) the raw input data and indexing and visualizing the data using a dashboard.

This can help during the manual RCA-process: anomalies can be visually detected using graphs and messages logged as an error can be highlighted.

But this can also lead to an overload of information. Google opted the term 'Golden signals' in order to find the best filters for the raw data. Amongst others they have identified **Latency, Traffic, Errors** and **Saturation** [89] as good starting points for effective signals. Based on these signals thresholds can be defined at which alerts are send to Ops personnel. Next to that dashboards can provide tools for long-term trend analysis.

Interesting to note is that Google [89], in general, has trended toward simpler and faster monitoring systems, with better tools for post hoc analysis. They tend to avoid "magic" systems that try to learn thresholds or automatically detect causality.

13.2 RCA of test failures

"Testing leads to failure, and failure leads to understanding."

- Burt Rutan

In addition to the tools used in production environments, in the testing phase of an application we have additional specialized questions to ask: Is this test failing because of a product bug, or are the assertions of a test case not correct (e.g. not conforming to specifications) or is the test environment itself unstable?

Classification of test failures can be a tedious task. Tools like Elasticsearch can be applied to automatically classify automation test failures in order to prevent expensive software engineers to parse a lot of application logs to triage and classify a test failure [90].

Using Test Log Output and Stack Traces

A full-text search solution like Elasticsearch can be applied to perform root cause analysis of e.g. test failures in automated test suites using the log-output of frameworks like xUnit/JUnit or FitNesse acceptance tests [90]. These tools can cover all parts of the proverbial test pyramid.

[3] ELK-stack: https://www.elastic.co/what-is/elk-stack

[4] Find the root cause presentation by Atlassian: https://www.youtube.com/watch?v=SQ2pyS6aTpM

It is good to note that a e.g. a unit test, authored by a programmer, is easier to troubleshoot since the programmer is already working on that part of the software and does not have to make a mental 'context switch' most of the time. But an acceptance test that fails can be caused by any part of the application, and in that case an automated RCA can be of tremendous value to a development team.

Test Case Shrinking

A lesser known tool, but a very powerful one if applied correctly, is the use of Test Case Shrinking. To assist with root-cause analysis the system uses test shrinking to report much smaller versions of failing test cases.

This technique is used as a part of property-based testing frameworks like QuickCheck [91]. How this works is that the tool reports much smaller versions of failing test cases that are easier to parse for humans. This is possible since property-based testing uses random 'fuzzy' testdata and has heuristics that can try to pin-down the cause of the failure as best as possible.

Examples of application of this technique are *Property-Based Testing of Browser Rendering Engines with a Consensus Oracle* [92] and an explanation of shrinking with the Java library *JQwik* [93] in the blogpost by its creator: the-importance-of-being-shrunk [94].

Natural Language Processing

Another approach to speed-up and automate RCA is the use of NLP for failures [95]. This uses the problem description as given by human testers to perform RCA when no other input is given, next to a problem description composed by a tester or end-user.

14. Code Quality and Static Analysis

In the software industry, it is standard practice to perform reviews on the code of other developers to preserve the quality of developed software. This practice includes mitigating the increase of risk that may result from adaptations of source code. It is essential to observe that changes to specific parts of a project may affect the functionality of other (unchanged) pieces of software that depend on the adapted code. Such dependencies are hard to identify, and as such, they present an increased risk.

Over the past decades, code analysis tools have been developed to make it easier for software developers to do code reviews and develop high-quality software. These tools cover all kinds of different code metrics and integration with other platforms. Some examples of the most advanced tools are SonarSource [96] and Checkmarx [97]. SonarSource supports 27 different languages and has commercial plans that include integration with GitLab. It performs source code analysis on the entire code base in repositories, as well as on individual merge requests using a plethora of code metrics (500+). Next to commercial version, their extendable open source community edition allows other developers to create plugins to increase the capabilities of the software. Lists of other tools [98] that perform static code analysis have been created as well.

However, while other tools exist, code reviews are often done manually by developers through version control systems like GitLab. GitLab is a web-based, fully-integrated DevOps lifecycle tool that provides a Git-repository manager with an integrated wiki, issue-tracking and continuous integration/continuous deployment pipeline features. The problem is that these static analysis tools are external to the GitLab environment, and therefore they are often disregarded by the users. Consequently, these manual code reviews still take much time and are hard to perform when merge requests contain large numbers of changes.

Thus, there is a need for an appropriate selection of a set of tools and metrics which can aid code reviews and optimize the process of development. Furthermore, the outputs of the tools and the effect of certain metrics need to be analysed and adapted to suit the needs of the developers and provide optimal results. The goal is to detect important issues, conflicts and weak spots of the code and merge requests and save the most time and effort for the developers while eliminating false positives as much as possible.

In the context of IVVES, we could start with investigating data sources such as different code analysis tools and unifying the output of these for a specific codebase into a generic format so that this data can be processed and analysed. After that machine learning and statistical techniques should be applied to enable predictive maintenance, fault analysis and anomaly detection, which would assist developers in their work.

15. Conclusions

In this deliverable we provided a high level introduction and brief state of the art report on the core topics and techniques that build the foundation of the work in WP4 of IVVES on data-driven engineering. It can also serve as a quick guide for other projects that would like to adopt a data-driven engineering process to identify the key concerns, topics and techniques to focus on.

16. References

[1] H. Holmström Olsson and J. Bosch, “Data Driven Development: Challenges in Online, Embedded and On-Premise Software,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11915 LNCS, pp. 515–527.

[2] “Building data science teams - O’Reilly Radar.” [Online]. Available: http://radar.oreilly.com/2011/09/building-data-science-teams.html. [Accessed: 26-Jun-2020].

[3] “How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read.” [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#38f7a89960ba. [Accessed: 25-Jun-2020].

[4] “Guide to the General Data Protection Regulation (GDPR) | ICO.” [Online]. Available: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/. [Accessed: 25-Jun-2020].

[5] L. Rocher, J. M. Hendrickx, and Y.-A. de Montjoye, “Estimating the success of re-identifications in incomplete datasets using generative models,” *Nat. Commun.*, vol. 10, no. 1, p. 3069, 2019.

[6] “Researchers spotlight the lie of ‘anonymous’ data | TechCrunch.” [Online]. Available: https://techcrunch.com/2019/07/24/researchers-spotlight-the-lie-of-anonymous-data/. [Accessed: 25-Jun-2020].

[7] S. Kabadayi, A. Pridgen, and C. Julien, “Virtual sensors: Abstracting data from physical sensors,” in *Proceedings - WoWMoM 2006: 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2006, vol. 2006, pp. 587–592.

[8] L. Liu, S. M. Kuo, and M. C. Zhou, “Virtual sensing techniques and their applications,” in *Proceedings of the 2009 IEEE International Conference on Networking, Sensing and Control, ICNSC 2009*, 2009, pp. 31–36.

[9] J. Stephant, A. Charara, and D. Meizel, “Virtual sensor: Application to vehicle sideslip angle and transversal forces,” *IEEE Trans. Ind. Electron.*, vol. 51, no. 2, pp. 278–289, Apr. 2004.

[10] C. G. Mattera, J. Quevedo, T. Escobet, H. R. Shaker, and M. Jradi, “Fault Detection and Diagnostics in Ventilation Units Using Linear Regression Virtual Sensors,” in *International Symposium on Advanced Electrical and Communication Technologies, ISAECT 2018 - Proceedings*, 2019.

[11] K. Rastogi and N. Saini, “Virtual Sensor Modelling using Neural Networks with Coefficient-based Adaptive Weights and Biases Search Algorithm for Diesel Engines,” Dec. 2017.

[12] B. Lin, B. Recke, J. K. H. Knudsen, and S. B. Jørgensen, “A systematic approach for soft sensor development,” *Comput. Chem. Eng.*, vol. 31, no. 5–6, pp. 419–425, May 2007.

[13] A. Lerro, A. Brandl, M. Battipede, and P. Gili, “A Data-Driven Approach to Identify Flight Test Data Suitable to Design Angle of Attack Synthetic Sensor for Flight Control Systems,” *Aerospace*, vol. 7, no. 5, p. 63, May 2020.

[14] A. N. Srivastava, “Greener aviation with virtual sensors: A case study,” *Data Min. Knowl. Discov.*, vol. 24, no. 2, pp. 443–471, Mar. 2012.

[15] M. Shakil, M. Elshafei, M. A. Habib, and F. A. Maleki, “Soft sensor for NOx and O2 using dynamic neural networks,” *Comput. Electr. Eng.*, vol. 35, no. 4, pp. 578–586, Jul. 2009.

[16] J. Kullaa, “Bayesian virtual sensing for full-field dynamic response estimation,” in *Procedia Engineering*, 2017, vol. 199, pp. 2126–2131.

[17] Y. Iwashita, A. Stoica, K. Nakashima, R. Kurazume, and J. Torresen, “Virtual Sensors Determined Through Machine Learning,” in *World Automation Congress Proceedings*, 2018, vol. 2018-June, pp. 318–321.

[18] S. Madria, V. Kumar, and R. Dalvi, “Sensor cloud: A cloud of virtual sensors,” *IEEE Softw.*, vol. 31, no. 2, pp. 70–77, 2014.

[19] S. Bose, A. Gupta, N. Mukherjee, and S. Adhikary, “Towards a sensor-cloud infrastructure with sensor virtualization,” in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2015, vol. 2015-June, pp. 25–30.

[20] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, “Beyond average: Toward sophisticated

sensing with queries," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2634, pp. 63–79, 2003.

[21] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1. pp. 122–173, Mar-2005.

[22] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and Beyond: New Aggregation Techniques for Sensor Networks," *SenSys'04 - Proc. Second Int. Conf. Embed. Networked Sens. Syst.*, pp. 239–249, Aug. 2004.

[23] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *SIGMOD Record*, vol. 31, no. 3. pp. 9–18, Sep-2002.

[24] R. L. Neitzel, N. S. Seixas, and K. K. Ren, "A Review of Crane Safety in the Construction Industry," *Appl. Occup. Environ. Hyg.*, vol. 16, no. 12, pp. 1106–1117, Dec. 2001.

[25] "Managing Data Science Projects Using CRISP-DM process framework." [Online]. Available: https://medium.com/@aqureshi/managing-data-science-projects-using-crisp-dm-2b0682c0d894. [Accessed: 25-Jun-2020].

[26] "An Artificial Intelligence Quality Framework | Sogeti." [Online]. Available: https://www.sogeti.nl/nieuws/artificial-intelligence/blogs/artificial-intelligence-quality-framework. [Accessed: 25-Jun-2020].

[27] H. J. Seltman, "Experimental design and analysis," 2015.

[28] "Data Preparation for Machine Learning | DataRobot Artificial Intelligence Wiki." [Online]. Available: https://www.datarobot.com/wiki/data-preparation/. [Accessed: 25-Jun-2020].

[29] "How to Handle Missing Data - Towards Data Science." [Online]. Available: https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4. [Accessed: 25-Jun-2020].

[30] "Handling imbalanced datasets in machine learning - Towards Data Science." [Online]. Available: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28. [Accessed: 25-Jun-2020].

[31] R. Jabbari, N. Bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A systematic mapping study on definitions and practices," in *ACM International Conference Proceeding Series*, 2016, vol. 24-May-201, pp. 1–11.

[32] "SAFe 5.0 Framework - SAFe Big Picture." [Online]. Available: https://www.scaledagileframework.com/. [Accessed: 25-Jun-2020].

[33] "Sharing the DevOps journey at Microsoft | Azure Blog and Updates | Microsoft Azure." [Online]. Available: https://azure.microsoft.com/en-us/blog/sharing-the-devops-journey-at-microsoft/. [Accessed: 25-Jun-2020].

[34] "State of DevOps," 2018. [Online]. Available: https://services.google.com/fh/files/misc/state-of-devops-2019.pdf. [Accessed: 25-Jun-2020].

[35] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Softw.*, vol. 33, no. 3, pp. 94–100, 2016.

[36] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A Survey of DevOps Concepts and Challenges," *ACM Comput. Surv.*, vol. 52, no. 6, Nov. 2019.

[37] L. E. Lwakatare *et al.*, "DevOps in practice: A multiple case study of five companies," *Inf. Softw. Technol.*, vol. 114, pp. 217–230, 2019.

[38] "DataOps IBM." [Online]. Available: https://www.ibm.com/se-en/analytics/dataops. [Accessed: 25-Jun-2020].

[39] "Gartner Hype Cycle for Data Management Positions Three Technologies in the Innovation Trigger Phase in 2018." [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-09-11-gartner-hype-cycle-for-data-management-positions-three-technologies-in-the-innovation-trigger-phase-in-2018. [Accessed: 25-Jun-2020].

[40] "What is DataOps? | Platform for the Machine Learning Age | Nexla." [Online]. Available: https://www.nexla.com/define-dataops/. [Accessed: 25-Jun-2020].

[41] "From DevOps to DataOps - DataOps Tools Transformation | Tamr." [Online]. Available: https://www.tamr.com/blog/from-devops-to-dataops-by-andy-palmer/. [Accessed: 25-Jun-2020].

[42] Y. Dang, Q. Lin, and P. Huang, "AIOps: real-world challenges and research innovations," in *2019*

IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2019, pp. 4–5.

[43] "What is AIOps? | IBM." [Online]. Available: https://www.ibm.com/cloud/learn/aiops. [Accessed: 25-Jun-2020].

[44] "AIOps in 2020: A Beginner's Guide – BMC Blogs." [Online]. Available: https://www.bmc.com/blogs/what-is-aiops/. [Accessed: 25-Jun-2020].

[45] "How to Get Started With AIOps - Smarter With Gartner." [Online]. Available: https://www.gartner.com/smarterwithgartner/how-to-get-started-with-aiops/. [Accessed: 25-Jun-2020].

[46] "GigaOm-Delivering on the Vision of MLOps." [Online]. Available: https://azure.microsoft.com/en-gb/resources/gigaom-delivering-on-the-vision-of-mlops/. [Accessed: 25-Jun-2020].

[47] "MLOps with a Feature Store - Towards Data Science." [Online]. Available: https://towardsdatascience.com/mlops-with-a-feature-store-816cfa5966e9. [Accessed: 25-Jun-2020].

[48] "MLOps: CI/CD for Machine Learning Pipelines & Model Deployment with Kubeflow - Growing Data." [Online]. Available: https://growingdata.com.au/mlops-ci-cd-for-machine-learning-pipelines-model-deployment-with-kubeflow/. [Accessed: 25-Jun-2020].

[49] "MLOps: Continuous delivery and automation pipelines in machine learning." [Online]. Available: https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning. [Accessed: 25-Jun-2020].

[50] "AIOps vs MLOps | Ravi Kalia Blog." [Online]. Available: https://project-delphi.github.io/blog/aiops-vs-mlops/. [Accessed: 25-Jun-2020].

[51] "Fault Class Prediction in Unsupervised Learning using Model-Based Clustering Approach." [Online]. Available: https://www.researchgate.net/publication/322900854_Fault_Class_Prediction_in_Unsupervised_Learning_using_Model-Based_Clustering_Approach?channel=doi&linkId=5a74cec40f7e9b41dbce3114&showFulltext=true. [Accessed: 25-Jun-2020].

[52] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, 2012.

[53] A. Angelopoulos *et al.*, "Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects," *Sensors (Switzerland)*, vol. 20, no. 1. MDPI AG, 01-Jan-2020.

[54] M. Reider, S. Magnus, and J. Krause, "Feature-based testing by using model synthesis, test generation and parameterizable test prioritization," in *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*, 2018, pp. 130–137.

[55] J. Whittle and J. Schumann, "Generating statechart designs from scenarios," 2000, pp. 314–323.

[56] D. Lorenzoli, L. Mariani, and M. Pezzè, "Automatic generation of software behavioral models," in *Proceedings - International Conference on Software Engineering*, 2008, pp. 501–510.

[57] H. Prähofer, R. Schatz, and A. Grimmer, "Behavioral model synthesis of PLC programs from execution traces," in *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, 2014.

[58] N. H. Bakar, Z. M. Kasirun, N. Salleh, and H. A. Jalab, "Extracting features from online software reviews to aid requirements reuse," *Appl. Soft Comput. J.*, vol. 49, pp. 1297–1315, Dec. 2016.

[59] A. Ferrari, G. O. Spagnolo, and F. Dell'Orletta, "Mining commonalities and variabilities from natural language documents," in *ACM International Conference Proceeding Series*, 2013, pp. 116–120.

[60] J. A. Hess, E. William, and A. Novak, "Feature-Oriented Domain Analysis (FODA) Feasibility Study Kyo C. Kang, Sholom G. Cohen," 1990.

[61] J. Noppen, P. Van Den Broek, N. Weston, and A. Rashid, "Modelling Imperfect Product Line Requirements with Fuzzy Feature Diagrams," Universität Duisburg Essen, 2009.

[62] N. Itzik and I. Reinhartz-Berger, "SOVA - A Tool for Semantic and Ontological Variability Analysis," in *CAiSE*, 2014.

[63] J. Martinez, T. Ziadi, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Bottom-Up adoption of software product lines - A generic and extensible approach," in *ACM International Conference Proceeding*

Series, 2015, vol. 20-24-July, pp. 101–110.

[64] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, “Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools,” in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 2019, vol. 2019-Nov, pp. 1471–1479.

[65] “Using TPOT - TPOT.” [Online]. Available: https://epistasislab.github.io/tpot/using/. [Accessed: 25-Jun-2020].

[66] H. Jin, Q. Song, and X. Hu, “Auto-Keras: An Efficient Neural Architecture Search System,” *arXiv Prepr. arXiv1806.10282*, Jun. 2018.

[67] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, “Fault prediction under the microscope: A closer look into HPC systems,” in *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, 2012.

[68] T. Menzies and T. Zimmermann, “Software analytics: So what?,” *IEEE Softw.*, vol. 30, no. 4, pp. 31–37, 2013.

[69] D. Stahl, K. Hallen, and J. Bosch, “Continuous Integration and Delivery Traceability in Industry: Needs and Practices,” in *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, 2016, pp. 68–72.

[70] M. Saadatmand and A. Bucaioni, “OSLC tool integration and systems engineering-the relationship between the two worlds,” in *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014*, 2014, pp. 93–101.

[71] A. Hramyka, M. Winqvist, H. H. Olsson, and Y. Dong, “Traceability in continuous integration pipelines using the Eiffel protocol.”

[72] Y. Bathaee, “The Artificial Intelligence Black Box and the Failure of Intent and Causation,” *Harv. J. Law Technol.*, 2018.

[73] J. C. Mogul, “Emergent (mis)behavior vs. complex software systems,” in *Proceedings of the 2006 EuroSys conference on - EuroSys ’06*, 2006, p. 293.

[74] R. S. Arnold, “Software Change Impact Analysis,” IEEE Computer Society Press, Washington, DC, USA, 1996.

[75] M. S. KILPINEN, “The Emergence of Change at the Systems Engineering and Software Design Interface,” pp. 1–270, 2008.

[76] A. De Lucia, F. Fasano, and R. Oliveto, “Traceability management for impact analysis,” in *Proceedings of the 2008 Frontiers of Software Maintenance, FoSM 2008*, 2008, pp. 21–30.

[77] M. Acharya and B. Robinson, “Practical change impact analysis based on static program slicing for industrial software systems,” in *Proceedings - International Conference on Software Engineering*, 2011, pp. 746–755.

[78] C. Arora, M. Sabetzadeh, A. Goknil, L. C. Briand, and F. Zimmer, “Change impact analysis for Natural Language requirements: An NLP approach,” in *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, 2015, pp. 6–15.

[79] H. Alkaf, J. Hassine, T. Binalialhag, and D. Amyot, “An automated change impact analysis approach for User Requirements Notation models,” *J. Syst. Softw.*, vol. 157, p. 110397, Nov. 2019.

[80] P. Kosiuczenko, “The impact of class model redesign on state machines,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7307 LNCS, pp. 264–279.

[81] M. Tufano, H. Sajnani, and K. Herzig, “Towards predicting the impact of software changes on building activities,” in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER 2019*, 2019, pp. 49–52.

[82] S. L. Pfleeger, *Software engineering - theory and practice (3. ed.).* 2006.

[83] J. Imtiaz, S. Sherin, M. U. Khan, and M. Z. Iqbal, “A systematic literature review of test breakage prevention and repair techniques,” *Inf. Softw. Technol.*, vol. 113, pp. 1–19, Sep. 2019.

[84] M. Machalica, A. Samylkin, M. Porth, and S. Chandra, “Predictive Test Selection,” in *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, 2019, pp. 91–100.

[85] J. Itkonen and K. Rautiainen, “Exploratory testing: A multiple case study,” in *2005 International Symposium on Empirical Software Engineering, ISESE 2005*, 2005, pp. 84–93.

[86] “Principles of Chaos Engineering.” [Online]. Available: https://principlesofchaos.org/?lang=ENcontent. [Accessed: 26-Jun-2020].

[87] A. Basiri *et al.*, “Chaos Engineering,” *IEEE Softw.*, vol. 33, no. 3, pp. 35–41, May 2016.

[88] B. H. Sigelman *et al.*, “Dapper, a Large-Scale Distributed Systems Tracing Infrastructure.” 2010.

[89] “Google - Site Reliability Engineering.” [Online]. Available: https://landing.google.com/sre/sre-book/chapters/monitoring-distributed-systems/. [Accessed: 26-Jun-2020].

[90] “Selenium Conf 2018 - Smart Test Failure Analysis with ELK (Elastic Search, Log Stash & Kibana) | ConfEngine - Conference Platform.” [Online]. Available: https://confengine.com/selenium-conf-2018/proposal/6146/smart-test-failure-analysis-with-elk-elastic-search-log-stash-kibana. [Accessed: 26-Jun-2020].

[91] “QuickCheck,” *Software.legiasoft.com*.

[92] J. Martin and D. Levine, “Property-Based Testing of Browser Rendering Engines with a Consensus Oracle,” in *Proceedings - International Computer Software and Applications Conference*, 2018, vol. 2, pp. 424–429.

[93] “jqwik.” [Online]. Available: https://jqwik.net/. [Accessed: 26-Jun-2020].

[94] “Property-based Testing in Java: The Importance of Being Shrunk - My Not So Private Tech Life.” [Online]. Available: https://blog.johanneslink.net/2018/04/20/the-importance-of-being-shrunk/. [Accessed: 26-Jun-2020].

[95] “Test Intelligence demo: Defect Root Cause Analysis using NLP - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=x2Lz0dtYNJw. [Accessed: 26-Jun-2020].

[96] “Code Quality and Security | Developers First | SonarSource.” [Online]. Available: https://www.sonarsource.com/. [Accessed: 26-Jun-2020].

[97] “Checkmarx - Application Security Testing and Static Code Analysis.” [Online]. Available: https://www.checkmarx.com/. [Accessed: 26-Jun-2020].

[98] “List of tools for Java software metrics.” [Online]. Available: https://www.monperrus.net/martin/java-metrics. [Accessed: 26-Jun-2020].