EC5_N22 Data platform and lifecycle workflow

Methodological Guideline for Data Engineering in the context of Machine Learning and critical systems

Document reference number for ANR

contact@confiance-ai.fr | www.confiance.ai

Code de champ modifié

Formalization of the data lifecycle in the context of machine learning and critical systems

Document reference: L5.1.4.2 & L2.3.2.3

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Benoit Langlois	Thales	Author
Scientific responsible	Benoit Langlois	Thales	Author
Co-authors	Boris Robert	IRT Saint Exupéry	Reviewer
	Raphaël Braud	IRT System X	Reviewer
	Xavier Le Roux	Thales	Reviewer
	Amélie BOSCA	SOPRA STERIA	Reviewer

Document control

Revision	Date	Commentary	Author
1.0	19/12/2022	Final version Batch 2	B. Langlois
1.1	29/09/2023	Intermediary version for Batch 3	B. Langlois
2.0	21/12/2023	Final version Batch 3	B. Langlois

A. Introduction and abstract 5
A.1 General introduction to trustworthy AI challenges 5
A.2 Objective and structure of the document 5
B. Description of the method 6
B.1 Drivers of this guide 6
B.2 Notation and instructions for reading this document 6
B.3 Out of scope of this version of the document 6
C. Overview 7
C.1 Position 7
C.2 The Data lifecycle 9
D. Phase: Orient Data 14
D.1 Understand the context 15
D.2 Determine data objectives 16
D.3 Identify data risks 16
D.4 Analyze data value 17
D.5 Refine operational, system expectations for data 17
E. Phase: Specify and Architect data development 20
E.1 Consider the learning strategy 23
E.2 Architect Data 23
E.3 Specify data acquisition 24
E.4 Structure dataset 25
E.5 Specify Data KPI 26
E.6 Specify Annotations 26
E.7 Produce Data Specification Document 28
E.8 Define the test strategy 28
E.9 Improve data architecture and design 29
F. Phase: Develop Data 30

F.1 Pilot dataset relevance with expectations 33
F.2 Acquire data 34
F.3 Generate synthetic data 37
F.4 Identify missing data 47
F.5 Prepare data – Basic activities 49
F.6 Annotate dataset 52
F.7 Perform feature engineering 53
F.8 Detect and solve anomalies 53
F.9 Prepare data – Techniques 57
F.10 Constitute datasets 63
F.11 Execute development data pipeline 64
F.12 Improve developed data 65
G. Evaluate Data Trustworthiness 67
G.1 Characterize evaluation 71
G.2 Perform evaluation 73
G.3 Improve data quality 74
H. Phase: Test the dataset 76
H.1 Produce dataset test plan 79
H.2 Execute test plan 81
H.3 Execute test data pipeline 81
I. Phase: Release dataset 82
J. Support data 83
J.1 Store data 84
J.2 Version data 84
J.3 Visualize data 84
J.4 Dashboard data 85
J.5 Report data 85
J.6 Trace data 85
K. Combination of techniques 87
L. Conclusion 92
M. Bibliography 93

A. Introduction and abstract

A.1 General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness…).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices.

These are the challenges, among others, that the Confiance.ai program addresses.

A.2 Objective and structure of the document

In AI (Artificial Intelligence) and ML (Machine Learning), data is the complementary part of algorithms. Algorithm performance depends on data quality. Mastering the data life cycle in an End-to-End process, to provide trustworthy data, is fundamental in the development of critical/complex systems. This methodological guide explains how to perform data engineering activities for this purpose.

The structure of this guide is as follows:

- The Overview section introduces the reason to stress on the data lifecycle, presents an overall view of the data activities described in this guide, and the position of this data lifecycle in the context of the End-to-End process, led by EC2 [Robert et al., 2023].
- Next sections detail each activity organized by phase.
- The last section is exploratory and indicates how to combine data and ML techniques.

B. Description of the method

B.1 Drivers of this guide

The drivers of this guide are:

- Coherence with the End-to-End document [Robert et al., 2023] produced by EC2;
- Coherence with the Data Platform and with the EC5's activities;
- Application of the Arcadia method [ARCADIA] implemented by Capella [Capella];
- Respect of the Methodological Guideline for Trustworthy AI Assessment [Sohier et al., 2023];
- Cover of the early steps of the data lifecycle, from data orientation, to data release.

B.2 Notation and instructions for reading this document

The used notation and vocabulary respect those defined in the End-to-End document [Robert et al., 2023].

Regarding the diagrams:

- An activity diagram is a graphical representation of activities;
- Relationships between activities represent input / output flow exchanges;
- A complex activity is broken down in activities;
- An activity is amber in general, white when it concerns an activity being studied, and green when it is an external activity with a consumption and production relationship;
- The AND / OR operators, and the Start / End elements illustrate a process with the involved actors.

B.3 Out of scope of this version of the document

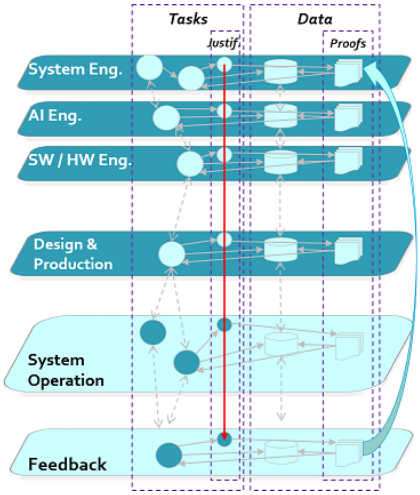
Parts not covered in this document are:

- The transversal activities of Documentation, Assurance Case, Reusability;
- The introduction of hybrid AI because most of Confiance.ai's works address ML.

C. Overview

C.1 Position

In the Machine Learning (ML), which represents a large part of the artificial intelligence (AI) community, most of the attention has been on model development. Despite the importance of data in the AI activities, the data development domain and its lifecycle are a topic of lesser interest. ML researchers and engineers usually consider only some main steps of data workflow, ignoring for instance architecture or validation concerns. A partial view of the complete data workflow is very risky to scale development of demanding, complex or critical systems. In such a context, improving the trustworthiness of AI systems becomes the main challenge of this decade by thoroughly formalizing the data engineering process to be complete, repeatable and robust. In the guiding line of Confiance.ai program, this document aims to tackle the data lifecycle in the perspective of an end-to-end data engineering process, in line with the EC2's end-to-end engineering vision of trustworthy systems integrating AI.



The development of complex systems cannot be achieved by a simple succession of activities, such as data collection, training/validation/testing from a dataset, and deployment. It is a co-engineering of several disciplines building together a system, i.e. it could include for example AI and software/hardware engineering disciplines, and other transversal disciplines and engineering specialties, such as safety and security. The data lifecycle is a set of multiple flows, in interactions and transformed by functions of the system. In order to certify such a system, we have to respect all requirements, ensure the traceability and explainability, etc. Reaching this level of expectations implies a formalization of the data lifecycle during development and deployment. All this means that there is a paradigm shift on data when developing complex and critical systems embed AI, which introduces uncertainty. This can be seen as a moving from a code-centric development, with the associated tests, to a global and mastered data lifecycle, at development and runtime.

In the context of trustworthiness of systems built in co-engineering with AI, the objective is to introduce a breakthrough with AI with less changing of the traditional practices in Systems Engineering. To do that, we simply started from the traditional development lifecycle, and next customized it for data with IA/ML practices. The proposed workflow is divided into five phases, as presented in the figure below. These phases are described as follows:

1) *Orient Data* identifies the business and operational goals for data, and the expectations on data consisting of requirements, ODD (Operational Design Domain) borrowed from automotive [SAE, 2021], and operational scenarios.
2) *Specify & Architect Data* specifies how to develop data, identifies and manages the data impacts on the system, and ensures that decisions are relevant in the long-term.
3) 3) *Develop Data* produces the tangible data artifacts according to the expectations, in respect of the architecture and design decisions, and compatible with the components implemented and used by the system. The next two phases, which incorporate the data, fall under systems engineering (EC2) and are not discussed in this document.
4) *IVVQ* (Integration, Verification, Validation, and Qualification) (EC6) ensures, before deployment, that the data assets produced during the implementation phase meet the expectations. This phase, with a complete view at the system level, integrates and validates the multiple data flows of the system, crossing both IA and not-IA components.
5) System *deployment* groups all the activities that make a system (software, hardware) available for use in operational conditions.

The process presented above is a foundation step. However, to guarantee the trustworthiness of a system with AI, five transversal concerns are added to the global workflow to contribute and master the lifecycle of data.

1) *Evaluate Data Trustworthiness* guarantees, especially during data development, data quality by measurement.
2) *Assurance case*, thanks to a justified measure of confidence, ensures that a system will function as intended in the environment of use [Weinstock, 2008].
3) *Automation by AI/MLops*, i.e. continuous integration applied to AI/ML models, improves productivity, and avoids manual operations, possible sources of errors.
4) (Digital) *Documentation* remains an important artifact to keep trace of data development and is a mandatory activity for the certification,
5) *Reusability* for capitalization and reuse of common data assets.

During this development cycle, continuous improvement remains central. Each phase reports improvements to its previous phase, going back successively from deployment to implementation, and possibly to specification, what is shown in the figure below with the cycle Activities-Measure-Improvement. Measures are realized by the Evaluate Data Trustworthiness for evaluation of data quality and trustworthiness; they helps to identify weaknesses and the sets of decisions to make for data improvement.

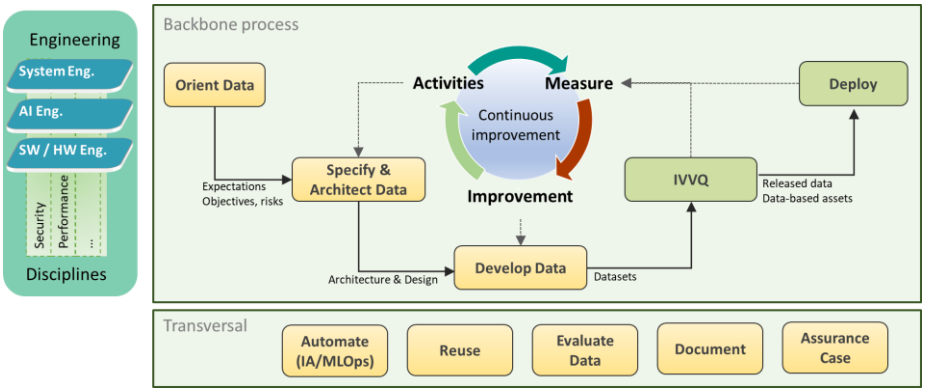


Figure 1: Overview of the data lifecycle by phase

C.2 The Data lifecycle

C.2.1 Activities

The diagram in Figure 2 (next page) is an overview of the main activities modeled in Capella. Data activities, in white, are grouped in the amber Produce data activity. They are led by the Data Engineer. Activities, in green, are activities external to Data Engineering. With an end-to-end vision at the system level, upstream activities, ensured by System Engineer, come from the operational and system levels. The Data Engineering domain provides data / datasets to the ML Model domain, which provides in reverse auxiliary models and improvement feedbacks for dataset. In downstream and in mirror of the Orient data, the IVVQ activity consumes released datasets for integration at the system level. For more details, refer to the End-to-end EC2 document [Robert et al., 2023].

The following figure is a big picture of the main data activities described in this document.

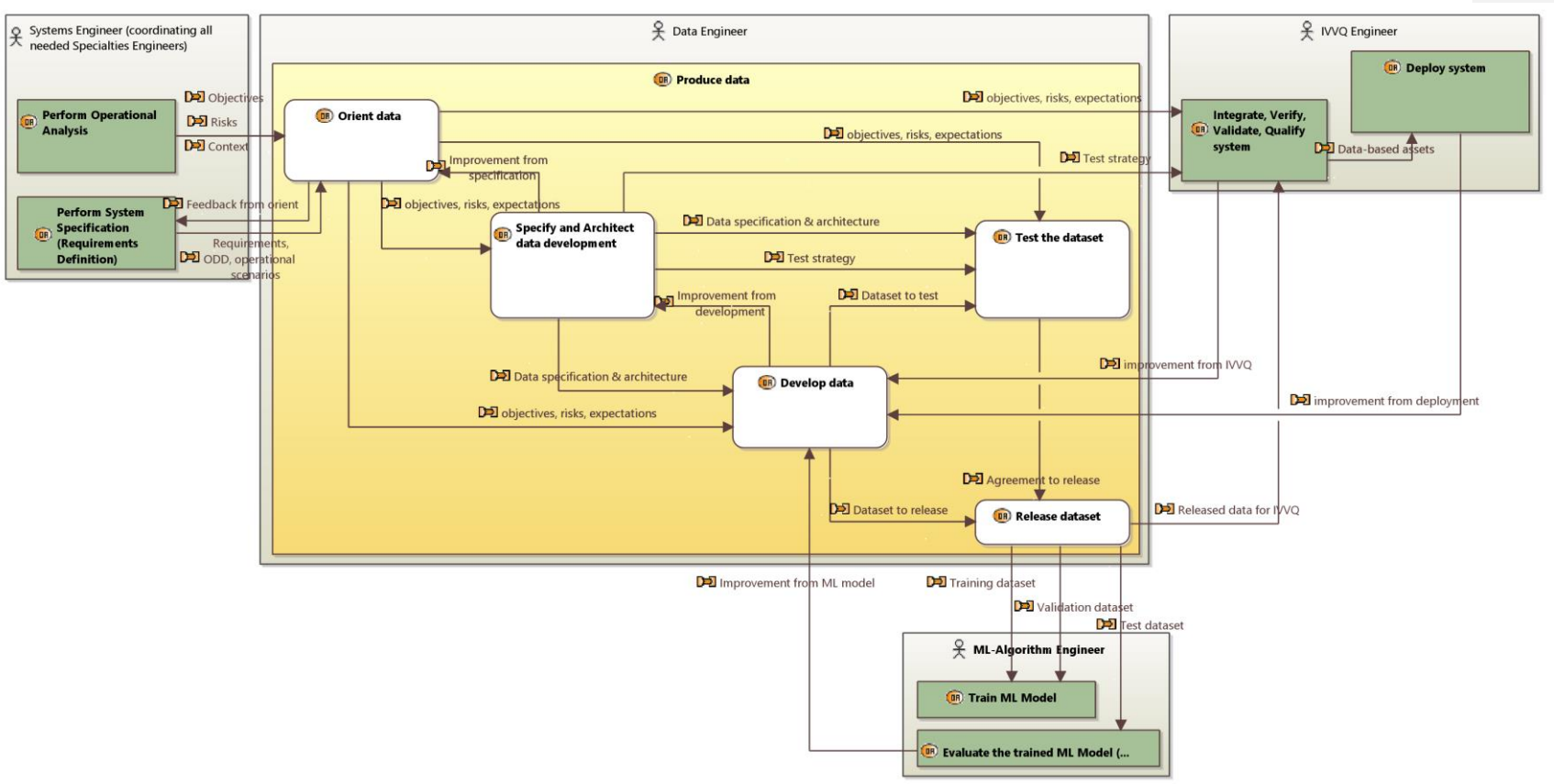


Figure 2: Main activities of data engineering

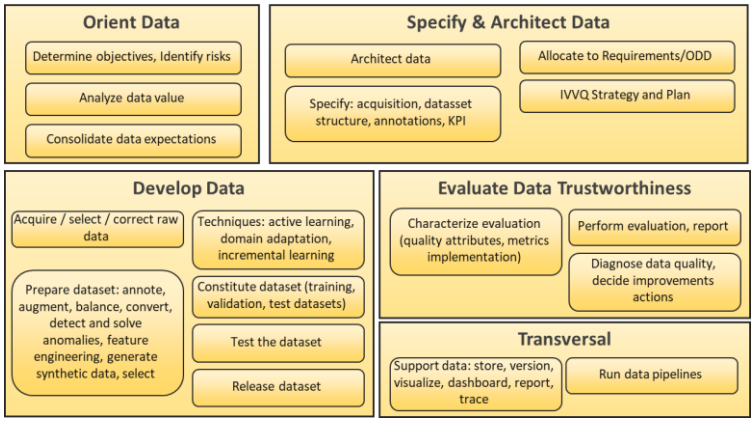


Figure 3: Overview of the main data activities

C.2.2 Data quality

Generally, and more specifically in ML, quality of the results depends on quality of the data. Poor, biased or unbalanced data skew the results.

But what does Data Quality mean? Data quality is related to the ability to meet the user requirements and ODD on data. More specifically:

1) data is the basic material to take care for further usages, and mainly for ML;
2) a quality attribute is a topic of interest to qualify data (e.g., consistency of data, respect of privacy on data), which requires a formula to compute a quality attribute;
3) after evaluation from the formula, it is asserted whether data meet the user requirements/ODD with quality.

Is the relationship so direct between requirements/ODD and data? Some quality attributes are obvious when a requirement is explicit (e.g., do not exceed 50 km/h), while others deserve analysis (e.g., representativeness of snow on an image), requires a deep analysis of data (e.g., for explainability of reasoning) or probabilistic analysis (e.g., data obsolescence). Usually, measure of data quality is not generic but project-specific for being in the correctness and accuracy of the project context.

Data quality does not apply at one or several times of the process, but all along from the early steps during Orient Data (e.g., with the definition of the data value) down to Deployment (e.g., with monitoring of data) On the other hand, data is regularly assessed to have regular reports and to measure progress, regression or drifts.

C.2.3 Systems engineering

Systems engineering describes a system at different levels of abstraction, often represented by a "V cycle" with descending and ascending branches. As depicted above, the data lifecycle also follows a V cycle. In practice, those cycles are not sequential, like in a waterfall cycle, but know regular iterations, for instance to release a system in several times. Considering those lifecycles, does the data lifecycle follow the one of systems engineering? Is it interwoven? To answer this question, remind the purpose the descending and ascending branches of system development. For more details, refer to [Robert et al., 2023].

The descending branch:

1. *Operational Analysis*, at the system context level, addresses the stakeholders expectations, usage conditions. It answers to the question: what the stakeholders involved in the system expect?
2. *System Specification* considers the description of the system from a black box point of view. It delineates clearly the limits of the system, identifies the interactions with stakeholders, and describes functions it supports. It answers to the question: what the system has to satisfy for the stakeholders?
3. *Architecture Definition and System Design* considers the description of the system from a white box point of view. Its structure, in system/hardware/software components, defines the logical architecture of the system, by searching for the best compromise against design drivers, (non-functional) constraints and viewpoints. It answers to the question: how the system will work so as to fulfil expectations?
4. *Component Specification* secures development and IVVQ through a physical architecture which deals with technical and development issues, favoring separation of concerns, efficient and safe component interaction (e.g. layered architecture, generic behavior and interaction patterns, component model, etc.). It answers to the question: how the system will be developed and built?
5. *Component Implementation* concerns the implementation of the components identified during the component specification.

The ascending branch consists in evaluating unitary the components, integrating the components, and next verifying and validating the system.

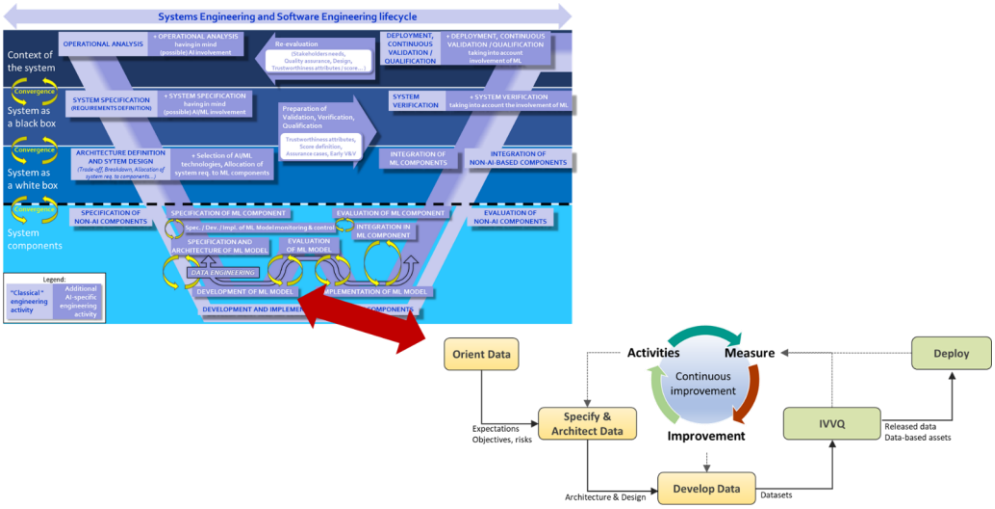


Figure 4: Relationship between system and data lifecycles

The exchanges of the Data engineering (DE) with the other domains are:

Domain	*Direction*	*Contents*
Operational and system level	Consumption	DE receives Requirements / ODD / operational scenarios defined at the operational level
Operational and system level	Production	DE provides functional and technical feedback for improvement
Machine Learning	Production	DE provides datasets to the Machine Learning engineering for training / validating / testing models
Machine Learning	Consumption	DE consumes auxiliary models in case of complex building of datasets
IVVQ	Production	DE provides the data test strategy, and datasets to IVVQ
IVVQ	Consumption	DE receives improvement feedback from IVVQ
Deployment	Consumption	DE receives improvement feedback from deployment

D. Phase: Orient Data

Objective:

- Data orientation defines the drivers for data to follow during its complete lifecycle

Roles in charge:

- Systems Engineer
- Data Engineer

Inputs:

- Data context elements
- All requirements: functional requirements and non-functional requirements
- Operational Design Domain (ODD)
- Operational scenarios

Outputs:

- Objectives to achieve for data
- Risks to reduce for data
- Value expected for data
- Data expectations, including data quality attributes

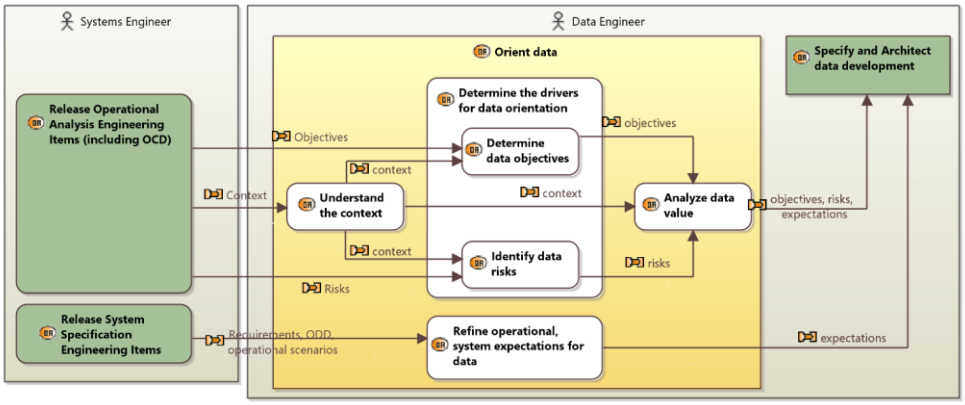


Figure 5: Orient data phase

The Orient Data phase identifies the expectations that concern the data and that impact the data lifecycle, from specification until deployment. This phase is then crucial to determine the reason of being for data, their levers and the risks to manage.

Related activities:

- Understanding of the context of the data, set at the operational level. The data context is the first input to understand the system in its environment, its history (e.g., volume, types of data) and evolutions, and its specificities to make the appropriate decisions during the data lifecycle.
- Consolidation of the expectations, coming from the system level, with:
  1) the functional (e.g., data domains), non-functional (e.g., safety, security) requirements,
  2) the characterization of the ODD,
  3) the operational scenarios.
- Determination of the objectives to reach and the risks to reduce. The objectives define the success criteria to maximize and achieve for data, while the risks define the failures and lack of trustworthiness to reduce. Objectives and risks are defined at the operational level with the OCD (Operational Concept Document).
- Explanation and expectation on the data value. The data value identifies the profitable value expected on data, with positive or negative impacts on the system.

At the end of this activity, the objectives, risks, value and expectations which apply all along the data lifecycle are known.

D.1 Understand the context

The purpose of this activity is to understand an application domain, its specificities in order to make the appropriate decisions during the various activities of the data development process.

Examples of common elements:

- The understanding of the market, its trends
- The operational elements (actors, interactions, expected services, constraints, etc.)
- The system as-it-is with its limitations and drawbacks to achieve the system as-to-be
- The meaning of the domain elements

Examples of initially collected data-specific information [IBM, CRISP-DM, p.13-16]:

- Existing data (databases, logs; format; performance, sources, metadata…), measures (size, throughput…)
- Usage
- Infrastructure (sizing)
- key attributes
- Quality of data: missing data, inconsistency, etc.
- Prospective on the data (percentage of growth).

The exchanges of this activity are the following:

- IN. Data of the operational context
- IN. Any valuable information on the existing system
- OUT. Data context

D.2 Determine data objectives

In the context of the system to be delivered, the objectives define the success criteria relating to data:

- Definition of the user-centric goals
- Definition of the major functional and non-function goals

The objectives are:

- Quantitative in order to be measurable during the development and deployment process
- Limited in terms of number

In this way, they can be assimilated to OKR (Objectives and Key Results), i.e. as major and high-level objectives to respect for the users, which can be refined in Technical Key Results, with a technical meaning for development teams (e.g., time to suggest proposals in the face of a critical situation < 1s). They define the criteria of achievements that must be followed at each step of a development to guarantee that the system, and here the data, remains on tracks.

The exchanges of this activity are the following:

- IN. Context
- OUT. Data Objectives

D.3 Identify data risks

Generally speaking, risk is the possibility of something bad happening. And data are everywhere in a system. Examples of risks on data: non-compliance with ethics because the data is not anonymized, loss of data between several cycles of learning, impossibility to compare performance improvement due to a wrong versioning of the first dataset, noncompliance of data format with a standard. The objective is to minimize the risks throughout the development and deployment process by clarifying the issues at the beginning and managing them during the execution of this process. Top-level risks are refined later in sub-risks, each significant at a step of the process.

A risk on the data is identified by its criticality, type, impacts (technical, financial, human, materiel, legal, social, ethic, etc.) and its mitigation means. The taxonomy of the quality attributes is an input to classify the types of the risks. Each risk is quantified.

The exchanges of this activity are the following:

- IN. Context
- OUT. Data Risks

D.4 Analyze data value

Data is an asset that has positive / negative / neutral impacts at different levels of an organization and a project, such as finance, marketing, social, IP, but also on development teams, or technical equipment. The goal with data is to generate profitable value.

The stake to know is:

- Considering all the costs, what are the benefits that can be obtained from data?
- How to value data in the ecosystem where it is involved?

Data value can be measured by:

- Its profitability in cost/benefit rate
- The creation of a knowledge which does not exist before or partially
- The improvement of the data quality of a system with the positive impacts on the related activities

Studies have to be led, such data monetarization, improvement of a complete value chain for instance to reduce the process time, or rate of errors / anomalies.

Data value is an upstream activity that must be followed during data specification and development, and in operational conditions.

The exchanges of this activity are the following:

- IN. Context
- IN. Objectives
- IN. Risks
- OUT. Data Value

D.5 Refine operational, system expectations for data

Operational requirements are refined in system requirements [Robert et al., 2023]. At those levels, some requirements are about data while others are not. Some explicit involve data while others must be deduced. Thus, it is first necessary to filter requirements that concern data and then reformulate them into data requirements.

However, requirements are not the only source. ODD and operational scenarios must also be considered. This is the reason why it is better to speak about expectations that encompass the three complementary notions of requirement, ODD and scenario. Another notion can be added, as a value of expectation, deduced from the three others: data trustworthiness, expressed with data quality attributes.

The purpose of this activity is to refine operational and system expectations into data expectations. It is crucial since it results in data specifications and development. Changing one expectation may be have

cascading impacts throughout the data lifecycle. Clarification of the user data expectations is therefore key to avoid additional costs, reductions in quality, delays, or, worse, damages in operational conditions.

1. *Requirements*

The operational level defines the high-level requirements of the system, which are refined in sub-levels (system/sub-systems, system components). This activity 1) identifies the system and component requirements that focus on data, and 2) analyzes their impacts on the data activities (specification, architecture, development). In output of this activity, it is ensured that data expectations are consistent, complete, and sufficient to perform the data activities.

2. *ODD*

This activity also considers ODD which precise the requirements (e.g., for autonomous driving, the color of the signal lines). Originally from the automotive, an ODD defines the limits within which the driving automation system is designed to operate, and as such, will only operate when the parameters described within the ODD are satisfied [1]. An ODD adds constraints to one or more requirements. The characterization of an ODD is a quantified description of an ODD that the implemented system must respect. The complete set of ODD determines the structure and content of the dataset and the constraints to meet on data.

[1] Taxonomy & Definitions for Operational Design Domain (ODD) for Driving Automation Systems J3259, https://www.sae.org/standards/content/j3259/

3. *Operational scenarios*

With the requirements and ODD characterization, the operational scenarios are a major input to specify and develop data. They describe the encountered situations in the application domain with standard, corner and edge cases. The set of test scenarios delineate what is expected on data. They constrain for instance the way to acquire data, or shape the content of the datasets.

4. *Data quality attributes*

This activity defines fourth main types of data quality attributes that the system must focus on and respect. [Delaborde et al., 2022] provides a list of quality attributes:

- *Technology*: dependability (reliability, security, maintainability), operability (effectiveness, robustness, efficiency, controllability), verifiability, portability, data quality, etc.
- *Interaction:* usability (learnability, accessibility, etc.), explainability
- *Ethics*: diversity, fairness, non-discrimination, benevolence, bias reduction, privacy, respect for the rule of law, system integrity, human agency and oversight, etc.
- *Intermediairies in trust*: audit, certification, compliance, labeling

Some data quality attributes apply at two levels:

- *Inherent data quality:* syntactic/semantic data accuracy, data completeness, data correctness, data diversity, data consistency, data usability, data representativeness, data reliability
- *System-dependent data quality*: data accessibility, data compliance, data confidentiality, data integrity, data efficiency, data precision, data traceability, data understandability, data availability, data portability, data recoverability, data timeliness, data governance

A data quality analysis must consider all the quality attributes as a whole, and not individually, with necessary tradeoffs. For instance, anonymization can reduce data representativeness but is priority to respect privacy and ethics constraints.

The exchanges of this activity are the following:

- IN. Functional requirements
- IN. Non-functional requirements
- IN. Data Quality requirements
- IN. Safety requirements
- IN. Security requirements
- IN. Ethics requirements
- IN. List of ODD
- IN. Operational scenarios
- OUT. Data expectations, including expectations on data quality attributes

E. Phase: Specify and Architect data development

Objective:

- Specifying and architecting the most relevant data with the best conditions of data development

Roles in charge:

- Data Engineer

Inputs:

- Objectives and risks
- Data expectations, particularly gathering the functional requirements, data requirements, safety requirements, ODD, and operational scenarios
- ML learning strategy

Outputs:

- Data specification document, made of:
  - Definition of the data KPI
  - Specification of the structuration of the datasets
  - Specification for the annotation rules with their annotation rules (gathered in an Annotation Guidelines")
  - Specification of the Data Acquisition system (i.e., identifying "Acquisition System" requirements)
  - Test strategy for data

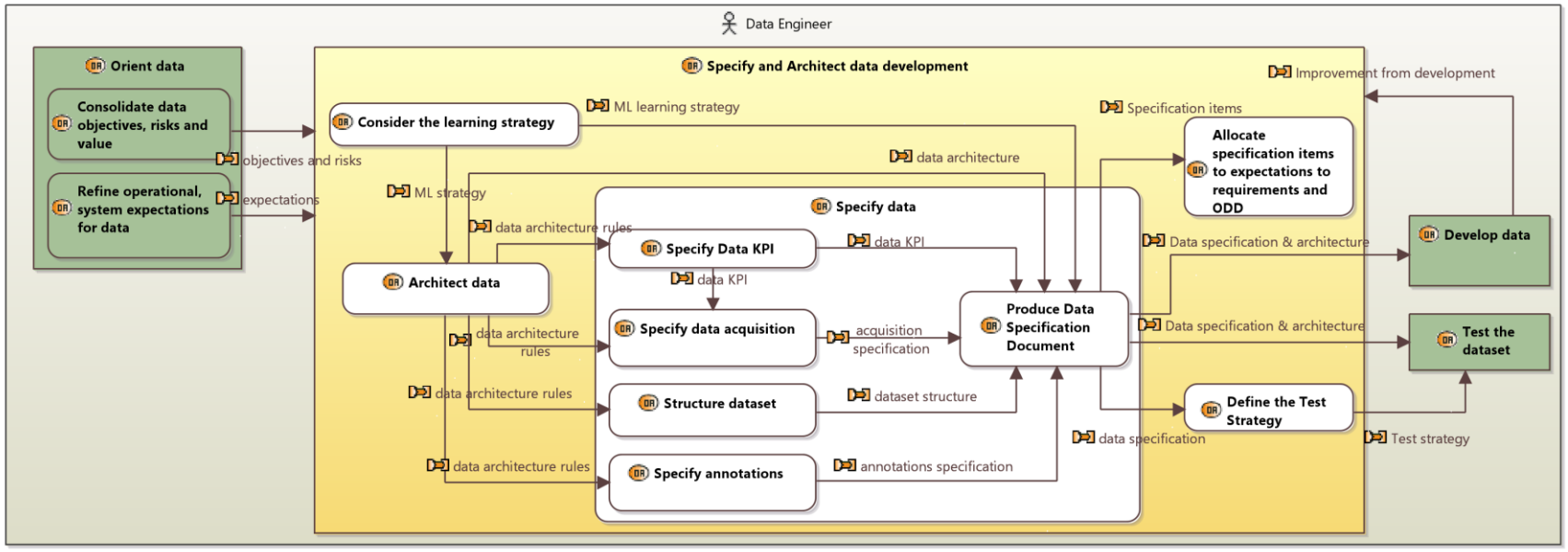


Figure 6: Specify and Architect Data

Data specification specifies how to structure a dataset and identify the key features in order that the developed data meet correctly the data expectations.

Data architecture generally refers to the overall design and organization of data within a system. It includes the structure, storage, management, and organization of data, as well as the policies, rules, and procedures for capturing, processing, and disseminating information. In the context of machine learning, architecture issues are similar. Data capture corresponds to data acquisition; data processing corresponds to data development: data dissemination corresponds to the use of dataset, especially by models. A data architect must have the complete view of a system where data is not reduced to the consideration of one dataset but the one of multiple flows of data between system components, where datasets are produced and consumed with respect of quality attributes.

Data specification and data architecture are driven from the objectives and expectations (functional requirements, model/data requirements, safety requirements, ODD, operational scenarios, etc.) identified during the Orient data phase, and the ML strategy defined at the operational and system levels and in the Machine Learning domain.

Data architecture considers the functional (e.g., structure, semantics, flow of data), non-functional (e.g., access, sensibility, size of data) and technical aspects to organize data and dimension all the allocated software and hardware resources.

Specify data

The internal data specification activity produces the required inputs for the data development phase, that are the data KPI for measure of the trustworthiness, the specification of the data acquisition, of the dataset structure for training / validation / testing, and the annotations to label the dataset. Those elements are gathered to produce the data specification document.

The specification of the dataset structure identifies a list of data characteristics (e.g. the speed of the vehicle, the distance between the vehicles…), and restrictions to respect, such as the range of possible values for each data characteristic (e.g. restriction of the speed range under which the function in question is operating, restriction on the range of distance between vehicles…). It consists of elaborating dataset specifications based on the input requirements. A first illustration of this activity and hence of the data specification process is performed on Renault Use Case named “Calibrating AI for Pedestrian Detection”.

Define the test strategy

Together, the data specification and architecture provide the elements to constitute the strategy to test data. This step is the input of the Test dataset activity (after the Develop data activity) to ensure that the dataset to release respects the expectations and the expected level of quality.

E.1 Consider the learning strategy

Complex cases in machine learning with large size of data or deep learning models, with an organization in multiple layers, imply a dimensioned data architecture. Examples of consideration: storing and retrieving data in intermediary layers, managing the life cycle of data during the learning phase, meeting performance constraints, reducing the risk of combinatorial explosion. Moreover, interfaces (API, structure of data and needed information by the algorithms, protocols, etc.) of data and algorithms must be compatible to work together, what it is an issue at the beginning when data structures and algorithms evolving quickly and are unstable.

The exchanges of this activity are the following:

- IN. Learning strategy
- OUT. Learning Strategy for structuration of dataset
- OUT. Learning Strategy for specification

E.2 Architect Data

Data architecture considers the expectations, the objectives and risks, and makes decisions that have major and long-term impacts on all activities related to the development and deployment of the data. The first specificity of data architecture is to be entangled in other architecture views, for instance, in business, technical, application architectures gathered in an enterprise architecture, and to depend on them. The second specificity here is data architecture comes up with IA/ML expectations, around the raw data, dataset, and all the technical considerations around them.

Examples of functional impacts:

- Organization of data by domains
- Data patterns, for instance with Domain-Driven [Evans, 2003]
- Data flow inside the systems, for instance to measure upstream/downstream data impacts
- Data interfaces (e.g. compatibility, conversion of formats)
- Formalization and representation of the data, and relationship with data governance
- Operations on the data lifecycle (CRUD), search, merge, etc.
- Manipulations in-memory vs. out-of-memory
- Optimization (e.g. number of features / rows)

Examples of non-functional impacts:

- Data access and usage
- Data sensibility
- Change management
- Ergonomics of visualization
- Performance, scalability, security, robustness, accuracy, etc.
- Chains of automation with data pipelines
- Optimization of storage

Examples of software/hardware impacts:

- Infrastructure to store and manipulate large volume of data
- Power consumption for green concerns

The exchanges of this activity are the following:

- IN. Expectations
- IN. Objectives, Risks
- OUT. Data Architecture

E.3 Specify data acquisition

The stake of specification and design of data acquisition is to ensure that the setting up of acquisition is the most representative and valuable for learning and knowledge management.

Basically, the acquired data meet the characteristics requested by the requirements, ODD, and operational scenarios. However, the conditions of acquisition can be delicate to honor. Examples regarding images and videos: respect of the lighting conditions, right angle / throughput / quality of the camera/sensor to capture a situation, correct sequence of actions and gestures in a context of a scene, the number and disposition of subjects/objects. For NLP, presence of valid/invalid words in a sentence, duplicates, semantic similarity, identification of topics. For time series, uniformity and well-chosen irregularities of events over time. In all cases, the data coverage must represent as much as possible all the valid cases and the cases of anomalies; the volumetry should not be too small (excess of frugality), too large (risk of combinatorial explosion), with a distribution conforming to the cases sought. The determination of contexts can impact the learning phases. It is then necessary to add sufficient annotations to characterize a situation. Moreover, in the perspective of sharing a dataset in three subsets of real, synthetic and simulated data, this implies to acquire the right data at the source.

Because it is costly to acquire raw data, the best strategy is to prepare and acquire data with the most fidelity in order to anticipate acquisition issues detected later during the phase of development with higher costs. This implies to well understand what data is, what is at the border and out of scope, how data will be managed, built and exploited. Data KPI are an input to stress on acquisition specificities.

Five questions can be asked before data acquisition and later during the data collection (in Develop Data): Is the data accessible, sizeable, useable (after preprocessing), understandable, and reliable? [Burkov, 2020, §3.1.1-3.1.5, 3.13]

Common problems with acquired data are: the high cost, the bad quality, the presence of noise, introduction of bias, a low predictive power, outdated examples, outliers, data leakage [Burkov, 2020, §3.2].

On the contrary, what is good data? It is informative, with a good coverage, reflects real inputs, is unbiased, not a result of a feedback loop, and is big enough [Burkov, 2020, §3.3].

For automation and reproductivity, a reference document is dedicated to the acquisition protocol. It specifies how the acquisition has to be performed. It is an input for the implementation of a data pipeline.

The exchanges of this activity are the following:

- IN. Expectations
- IN. Objectives, Risks
- IN. Data architecture rules
- IN. Data KPI
- IN. Data Design - Improvement - For Data Acquisition
- OUT. Acquisition specification

E.4 Structure dataset

In Machine Learning, and more largely in knowledge management, it is fundamental to correctly design a dataset, which structures the knowledge of an application domain, which is the pivot of efficient processing (e.g., extraction, transformation, visualization) during the development phase. By structure, we hear columns / attributes / dimensions of a data(set).

Some drivers help to define this structure:

- Applying the Occam's razor to keep the minimal and useful attributes
- Introducing attribute redundancy to speed up processing
- Encoding information for improvement of time and space

Regarding its content, the dataset can be enriched of contextual data, i.e. of metadata, in order to better particularized data (e.g. blur / rotation, day / night).

This activity to be related first to feature engineering during the dataset preparation, second to learning model that must be aligned to work together.

For complex project, data is not monolithic, has complex domain semantics, and comes with needs such adaptability, flexibility, or partitioning. Refer for instance to the Domain-Driven Design [Evans, 2003] for considerations that also applied to data in AI.

The exchanges of this activity are the following:

- IN. Expectations
- IN. Objectives, Risks
- IN. Data architecture rules
- IN. Learning Strategy
- IN. Improvement from data preparation and especially feature engineering
- OUT. Dataset Structure

E.5 Specify Data KPI

Data trustworthiness depends on data metrics to determine whether data provides the level of expected quality by the system. At this stage, the purpose is to define the KPI to reach. More precisely, it consists in setting targets on data quality attributes (Cf. Refine operational, system expectations for data activity) according the relevance given by the operational and system requirements, and eventually weighted by architecture rules and decisions.
The exchanges of this activity are the following: • IN. Expectations • IN. Objectives, Risks • IN. Architecture rules • OUT. Data KPI

E.6 Specify Annotations

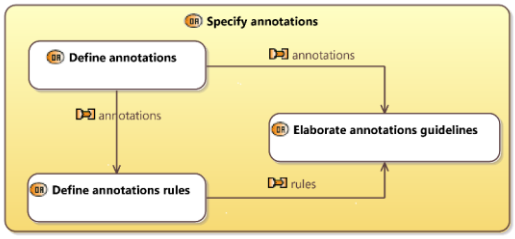


Figure 7: Specify annotations

E.6.1 Define annotations

Annotations are used to label a dataset in supervised or semi-supervised mode with the purpose to characterize and qualify data. An annotation enables to add semantic information on data not held by the data attributes or difficult to deduce. Examples: information to characterize weather conditions, a person, a car. A set of annotations defines a language for characterizing data, where annotations: • Have a business meaning • Have a set of attributes (e.g., numeric, string, percentage) • Are independent or related with other ones (e.g., composition, association, dependency) • Must respect or not constraints of use

- Are mapped on a representational notation, such as graphical (e.g., graphical form, color) for images or videos, or textual to annotate a text

An annotation instance is a label that is intended to be affixed to data (e.g., a blue rectangle to identify a person), manually or automatically, and to be used during the training phase.

An annotation language must be properly structured, respecting a clear taxonomy, for:

- Its understandability and its simplicity at the usage time
- Characterizing data without overlapping of meaning
- Its durability to cover new cases over time

Changing the organization of a set of annotations is costly (need of migration, update of guides, etc.).

Annotations are domain-specific or domain-independent, for example for reuse in case of cross-domain annotations or domain adaptation.

Annotations are accompanied by rules and conditions of application, and good practices.

The exchanges of this activity are the following:

- IN. Expectations
- IN. Objectives, Risks
- OUT. Annotations

E.6.2 Define annotations rules

Annotation labels cannot be affixed arbitrarily on data. They come with well form rules, a workflow and guidelines for their usage.

The exchanges of this activity are the following:

- IN. Expectations
- IN. Objectives, Risks
- OUT. Annotations rules

E.6.3 Elaborate annotations guidelines

This activity produces the annotations guidelines that will be used during the dataset annotation.

The exchanges of this activity are the following:

- IN. Annotations
- IN. Annotations rules
- OUT. Annotations Guidelines

E.7 Produce Data Specification Document

The Data Specification document contains all the elements from the Design stage that drive the Develop Data phase, such as the decisions of data architecture and acquisition, the elements to structure a dataset, the definition of the annotations.

The exchanges of this activity are the following:

- IN. Data Architecture
- IN. Acquisition setup
- IN. Dataset Structuration
- IN. Annotations
- IN. Learning Strategy
- OUT. Data Specification

E.8 Define the test strategy

Generally speaking, test strategy is a high-level plan that guides the definition and execution of the testing activities. It outlines the methodology and techniques that will be used to ensure that requirements are met with quality. It ensures that all necessary activities are planned, monitored and controlled throughout the testing lifecycle. Strategy answers the question: “what to do before doing?” Strategy helps to ask the right questions (and to solve them as best) on data testing before test implementation and testing during the development phase.

In the context of data testing, the data test strategy includes:

- Scope and objectives of data testing.
- Description of the methodology to test data
  - Focus on the types of testing to perform (functional, non-functional),
  - Used testing techniques, and the tools and frameworks to be implemented. Examples: identification of automation tools for data/MLOps, databases.
  - Logic of planning and executing tests: definition how testing activities are organized, scheduled, and executed.
- Test environment: required infrastructure resources (e.g., hardware, software) for testing.
- Identification of the test deliverables (e.g., test cases, test suites, scripts, reports).

Test strategy ensures that testing is an organized, planned and systematic activity for testing data with a high quality. Test strategy is created before the test plan and provides guidelines to create it. Refer to “Produce dataset test plan” activity.

The exchanges of this activity are the following:

- IN. Data Specification
- OUT. Data Test Strategy

E.9 Improve data architecture and design

Needs of improvement in this phase can come from:

- The phase itself:
  - For internal issues of completeness and consistency (e.g., for the definition of the annotations when specifying data acquisition from operational scenarios)
  - Due to evolution of the expectations, architecture rules or design
- The Develop data phase, especially during the data preparation, for instance with the use annotations, the change of structure of the dataset, first tests of performance, or with specific activities such as domain adaptation or incremental learning.

F.Phase: Develop Data

Objective:

- Implementation of the datasets that are consumed by ML Models

Roles in charge:

- Data Engineering
- ML-Algorithm Engineer

Inputs:

- Objectives and risks
- Data expectations, including the requirements, the ODD, and operational scenarios applicable to the data intended for the ML model
- Data architecture rules
- Data specification
- ML learning strategy
- Improvements from IVVQ, Deployment

Output:

- The training / validation / test datasets that will be used to train the ML model
- Test Plan
- Improvements for the Specify and Architect data phase

The following figure shows the main organization of the development activities. Test and release dataset are described in their respective section. The data development phase produces the datasets the most relevant with the expectations, data architecture and specification, to the destination of the ML-Algorithm engineer.

The four major steps to develop data are:

1. Acquisition of the data. It consists of acquiring real and raw data from one or several sources (e.g., sensors for image / video / time series, source text for NLP). Data is corrected when necessary (e.g., cleaning, data massage) to produce the initial dataset. In case of frugality, data acquisition is completed or replaced by generated synthetic data.
2. Preparation of the expected datasets that consists in transforming the initial dataset into datasets usable by models.
3. Jointly to acquisition and preparation, the dataset(s) is evaluated to verify the compliance with the expectations and the data quality rules.
4. Constitution of the dataset, which usually splits the dataset into training dataset, validation, and test datasets.

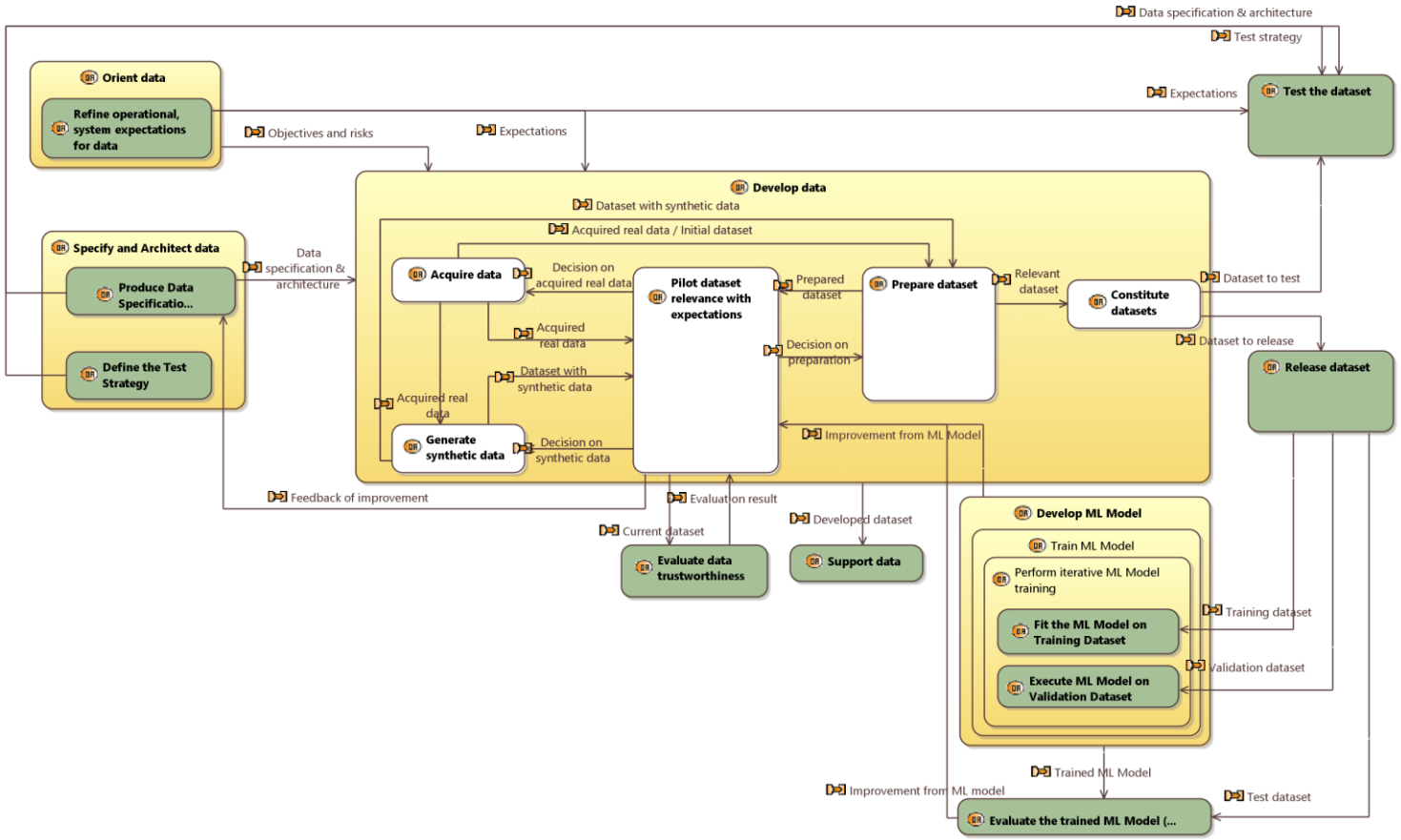


Figure 8: Develop Data phase

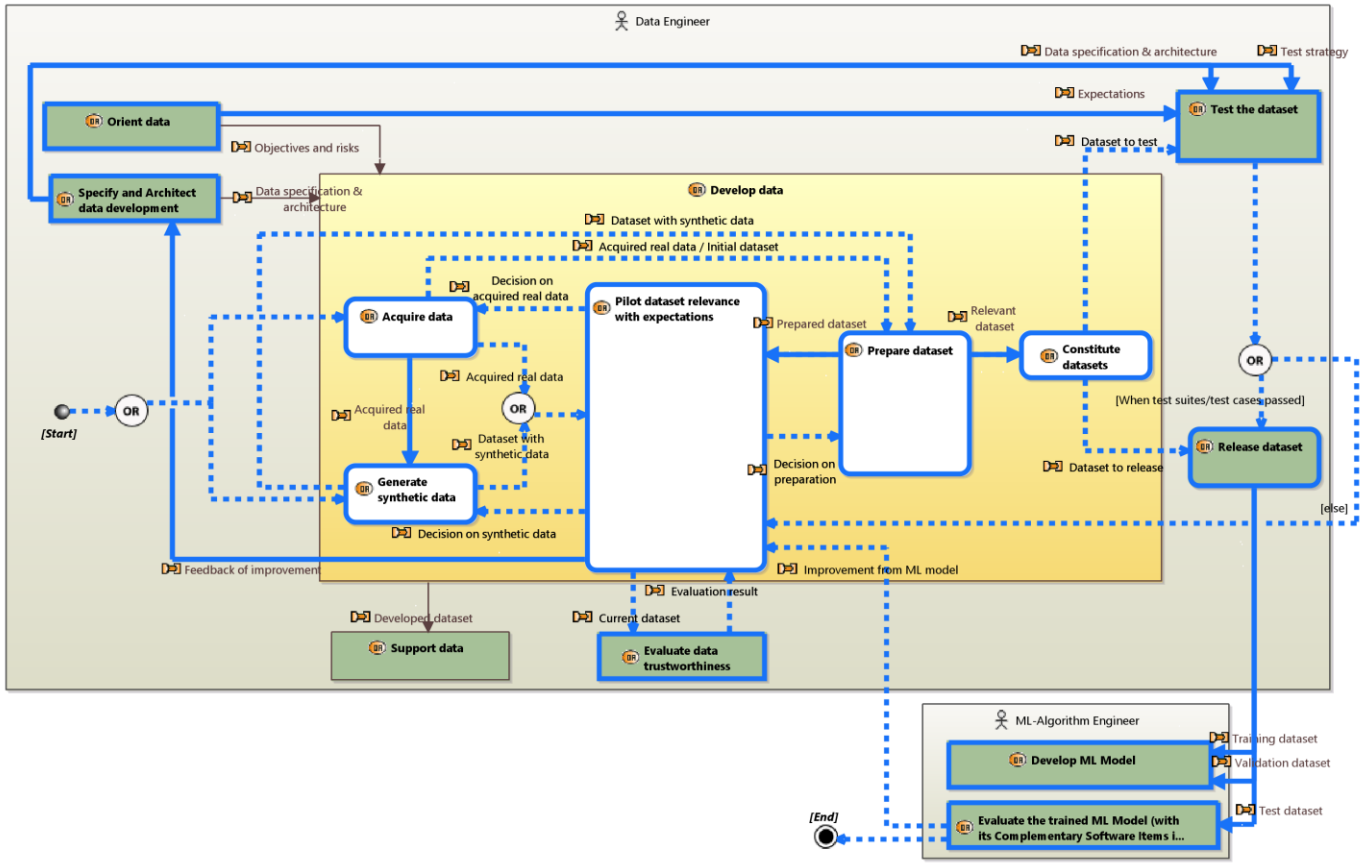


Figure 9: Develop Data phase with process

F.1 Pilot dataset relevance with expectations

The main output of the Data Development phase is a dataset that is most relevant to expectations, and according to the specification and architecture rules. This activity consists in sticking to this purpose by efficiently piloting the activities of data development: actions taken and to be taken, evaluation of progress made and to be continued.

- Acquisition and constitution of the initial dataset. It is ensured that data is correct and representative at the source. Otherwise, a new acquisition or improvement of the acquisition specification may be necessary.
- Generation of synthetic data. In case of frugality or difficulty to collect real data, data generation populates the dataset with synthetic data. The resulting dataset is either made of a mix of real and synthetic data or synthetic data only. This piloting activity compares the types of sources and mixes real and synthetic data acquisition for the best availability of required data.
- Preparation of the dataset. The piloting activity solicits any activity to shape and ensure that the dataset meets the expectations, specification and architecture rules (e.g., annotation, balance of dataset, anomaly management, application of techniques such as domain adaptation).

This activity leans one the Evaluate data trustworthiness to verify that the KPIs on data quality attributes for the considered dataset are met.

The exchanges of this activity are the following:

- IN. Objectives and risks
- IN. Expectations
- IN. Data specification and architecture
- IN. Dataset: acquired real data, generated synthetic data, being prepared
- OUT. Decision on data acquisition, generation or preparation

F.2 Acquire data

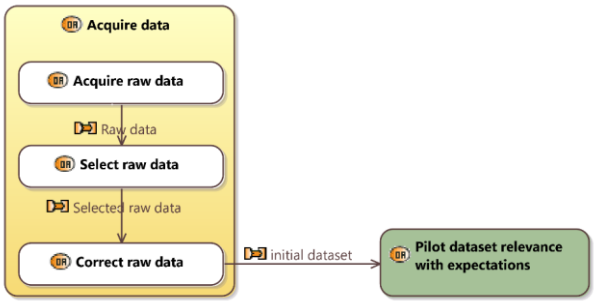


Figure 10: Acquire data

The constitution of the initial dataset in a given format (e.g., tabular, images, videos, text) consists in acquiring data from one or several data sources and to produce the initial dataset exploitable for ML processing.

The following process depicts the successive steps to acquire data, from raw data to a dataset meeting the expectations:

1. It starts with the data orientation by the definition of the objectives, risks, and the consolidation of the data expectations, to scope the data to acquire.
2. The data specification identifies the conditions to dimension correctly data acquisition (e.g., throughput, dimension of database for store images or videos, distributed infrastructure for differences of locations between acquisition and crowd-based labeling of datasets).
3. The purpose of data acquisition specification is to reduce time and costs by preventing bad conditions of acquisition (e.g., position of the camera, optimization of the dataset size).
4. The step of acquisition itself, followed by the selection of relevant data and correction of the raw data, is to constitute the initial dataset. This dataset is declared conformed after a successful compliance with the expectations. Then this dataset is versioned. The cases of incompliance come from either a bad acquisition or a problem of specification.
5. After acquisition, the next stage of dataset preparation continues before the model training.

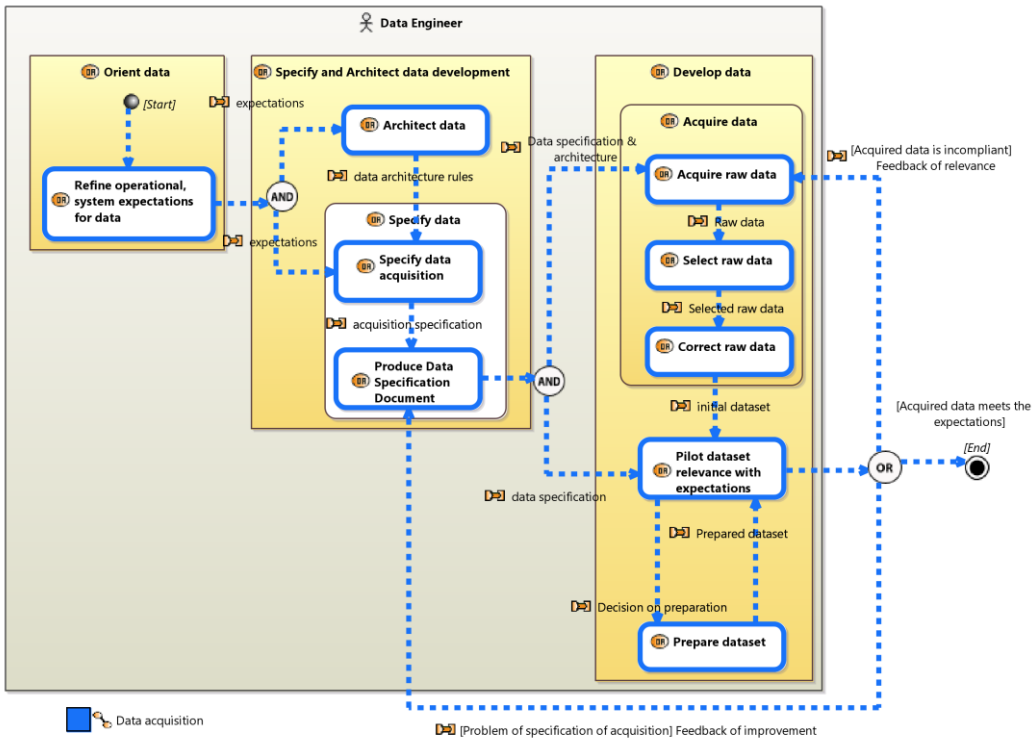


Figure 11: Process of Acquire data

F.2.1 Acquire data

The activity of data acquisition specification defines, at the Specify and Architect Data time, the strategy and protocol to acquire data optimally. This step consists in acquiring physically data according to this specification. The result is a set of raw data from one or multiple sources (e.g., sensors, database, Electronic Data Interchange), not necessary acquired at the same time. This can imply a management of different data formats and different modes of data persistence (e.g. files / sql / nosql databases). The campaign of acquisition is launched on demand or periodically. The data are erased each time or incrementally updated to keep the history.

Data acquisition is not a one-shot activity. Data refresh enables to keep data and models up-to-date while the environment evolves and to introduce the cases. Moreover, some acquisitions work in a streaming form that requires specific protocols (e.g., communication protocol, preprocessing data on the fly). A first assessment guarantees a preliminary level of quality of the acquired data.
The exchanges of this activity are the following: • IN. Raw data from data sources • OUT. Acquired data

F.2.2 Select raw data

Data acquisition is a data sink where the volume may become large and where all the data are not relevant for the given purpose of data or machine learning. Data selection is the operation to extract one or several subsets from the acquired raw data according to criteria given in the data specification.
The exchanges of this activity are the following: • IN. Raw data • OUT. Selected raw data

F.2.3 Correct raw data

The purpose of this activity is to provide an initial dataset from raw data gathered from relevant sources. It pre-processes the raw data to ensure a minimal quality before being enriched and exploited afterwards. The common operations of corrections to apply on the raw data are: • *Data massaging*: it removes in flow incorrectly formatted, duplicate, or incomplete data within a dataset, and handles the missing values. • *Data cleaning*: it ensures data is clean, comprehensive, error-free with accurate information, and helps in detecting outliers, and eliminating bias and obsolete information. • *Data anonymization*: it removes or encrypts personal information from the datasets to protect privacy with respect to data privacy regulations. • *Data correction* by type of dataset: depending on the type of data (e.g. image, text), operations are performed in order to improve the level of quality of data, such as blur reduction on images. • *Data correction* by application domain: depending on the application domain, corrections are applied according to domain-specific rules.
The exchanges of this activity are the following: • IN. Selected raw data

- OUT. Initial Dataset

F.3 Generate synthetic data

Generation of synthetic data compensates for situations marked by a lack of (labelled) data, rare, unexpected or very expensive situations in the real life, to reduce costs, or due to legal conditions to be respected. Examples: application of GDPR for data privacy, corner/edge cases, or feared events such as a careless pedestrian crossing a street, or attacks. Synthetic data, or artificially generated data, has the potential to overcome these issues and become a valuable resource for machine learning [Bosca, 2023].

Some general characteristics of generation of synthetic data:

- All types of data are potentially concerned by generation of synthetic data. Examples: insertion of persons or objects in an image or video; text generation or insertion of in an existing text; generation of information in tabular representation.
- Generation can start from an existing dataset available by acquisition, or from a scratch dataset.
- Inserted objects can be real data or synthetic data.
- Generation can be iterative, for instance adding persons in a street first, and adding snow in the next iteration.
- Different techniques of generative AI can be used and combined, such as diffusion models and GAN. For instance, persons are added by diffusion model and snow by GAN.

The following sections are organized as follows: 1) the first activity is to enrich the data specification with specific elements to generation of synthetic data; 2) a generic activity synthetizes the main steps of generation for synthetic data; 3) this generic activity is applied to diffusion model; 4) it is also applied to 3D-model.

F.3.1 Specification

The Data Specification activity produces the main document of specification for data engineering. After measuring the gap between this document and the expectations for generation of synthetic data (requirements, ODD, operational scenarios), this activity completes this specification document, either in the same or in a separate document, with new specificities:

- The objects to generate with their characteristics (e.g., a person, car),
- The objects classes,
- The context,
- The specification of the chosen generation technique, such as a diffusion model, GAN, with the expected generations,
- The KPIs to reach.

This specification describes the standard and corner/edge cases.

The exchanges of this activity are the following:

- IN. Data specification
- IN. Objects to generate with their classes
- IN. Context
- IN. Specification of ML model
- IN. KPI
- OUT. Data specification for generation of synthetic data

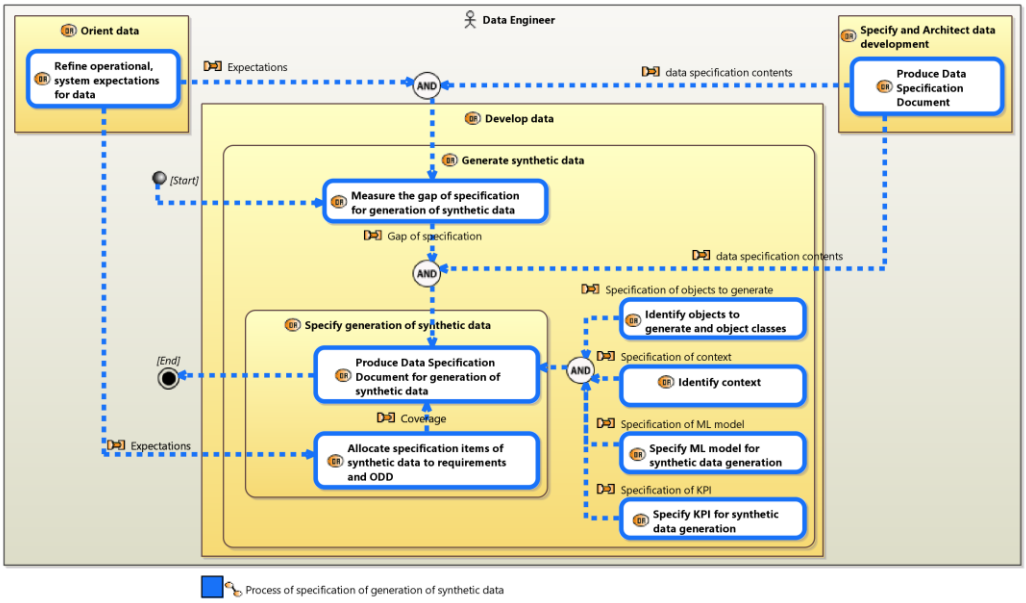


Figure 12: Specify generation of synthetic data

F.3.2 Generate synthetic data – Common process

This activity describes common behavior to generating synthetic data. The following two activities specialize it for image generation based on diffusion model (image insertion) and for 3D models.

Common activity of Generation of synthetic data

The specification contains all the elements to prepare and generate the synthetic data.

1) The first step is to start from an initial dataset. It comes either a) from real data by data acquisition, or b) from other sources, such as a 3D-model, which is created and initialized from scratch, or reused. In case of reuse, to continue to the next step, a validation checks that there is no issue with the dataset, such as the level of performance.
2) The dataset is then prepared: labels identify the objects to generate, contextual and required objects to insert (refer to the data generation activity with 3D-model below). The dataset at this stage is the dataset that will be used to train the ML model for generation of synthetic data. It is released for this purpose.
3) In parallel, the ML model is specified (e.g., identification of the type of generation algorithm, parameters, architecture). The ML model for data generation is considered as an auxiliary model. Indeed, it follows the traditional process of building the ML model algorithm, training the ML model with the released dataset, and releasing the ML model when validated. This process may be complex (e.g., reused of foundational model) and this model can be viewed as a black-box model. The purpose of ML specifications is to formulate what is expected from this model (refer to the generation of synthetic data by diffusion model, or application of style transfer for 3D-model).
4) Between dataset preparation and application of the released ML model, the dataset can be enriched of additional elements. Refer to the generation of synthetic data by diffusion model.
5) The inference to generate objects starts when the dataset is complete and the dedicated ML model is released.
6) The last step is validation. When the generated objects corresponds to the expected objects by comparison of the validation dataset, the dataset with the generated objects is released. Else, either a gap comes either from the trained ML model or from the prepared dataset.
7) It is possible to apply several inferences and validations in a row. For instance, 1) first cars are generated by diffusion model, followed by a validation, and 2) then night or snow effects are applied GAN, followed by a new validation.

The exchanges of this activity are the following:

- IN. Data specification for generation of synthetic data
- IN. Initial dataset
- OUT. Dataset with generated data

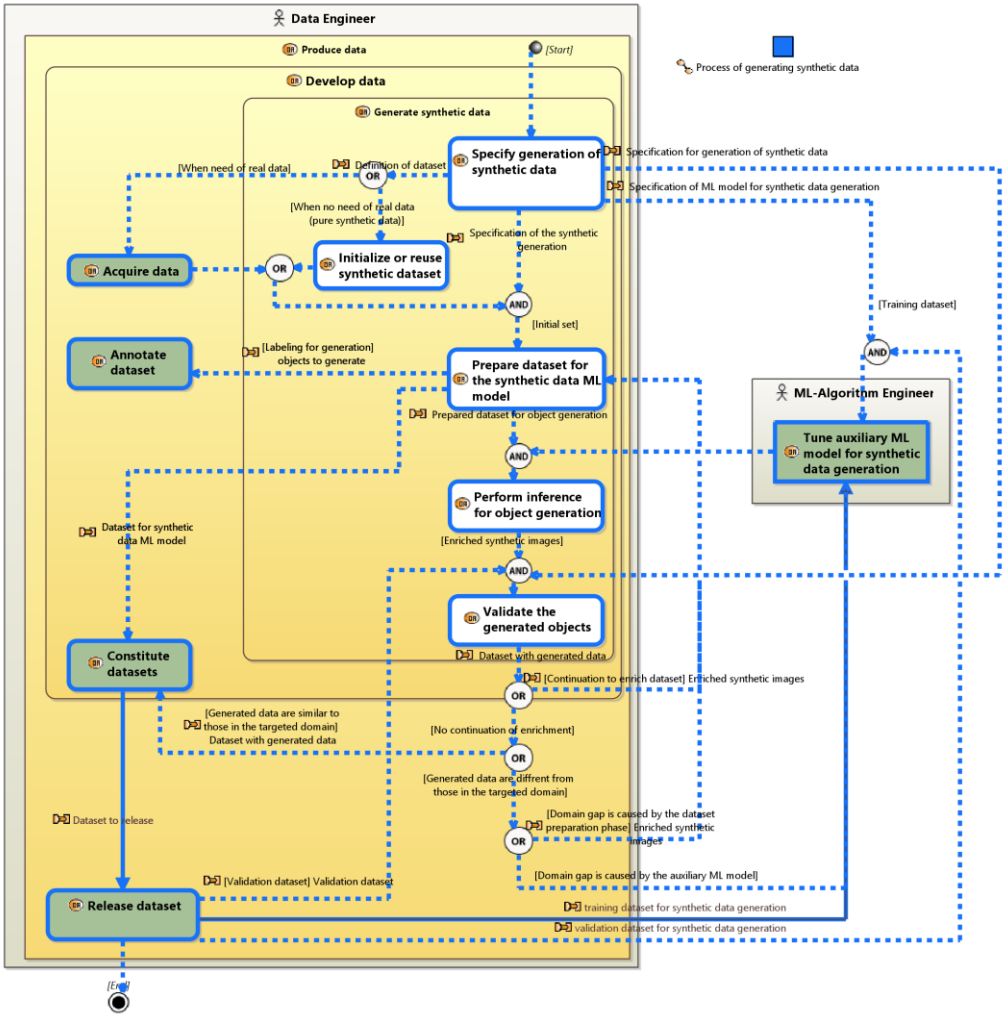


Figure 13: Generate synthetic data

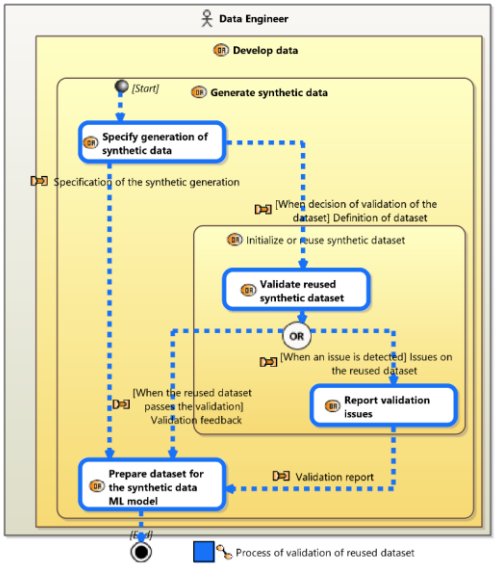


Figure 14: Process of validation of reused dataset

F.3.3 Generate synthetic data – Diffusion model

This activity is about generation of synthetic data by diffusion model. Differences with the generic activity are the following:

- The dataset is initialized by the data acquisition activity.
- There are two types of generation by diffusion model for images:
  - Inpainting: noisy zones identify the places where to generate objects; the inference consists in replacing those noisy zones by generated objects.
  - Image generation: there is no noisy zone, the inference consists in generated the full image.
- Validation verifies that the generated objects have the expected characteristics.

The exchanges of this activity are the following:

- IN. Data specification
- IN. Initial dataset from data acquisition
- OUT. Dataset with generated data

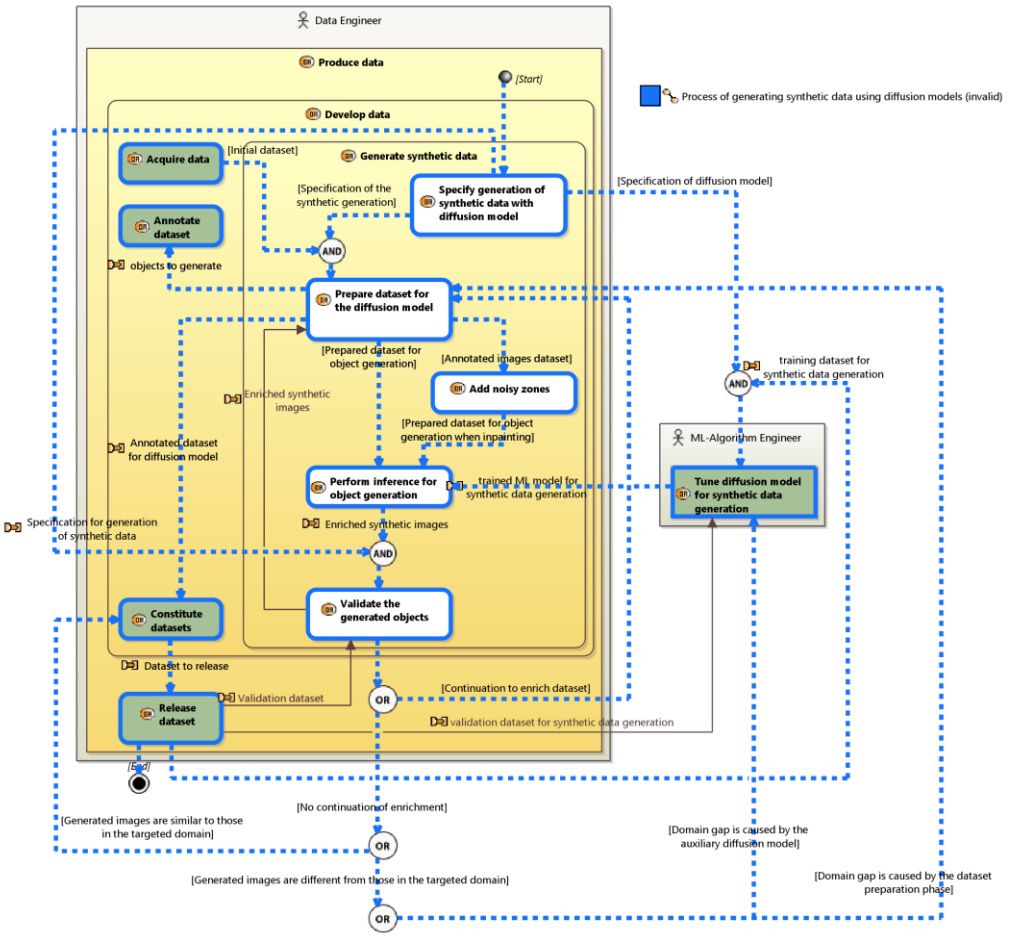


Figure 15: Generate synthetic data using diffusion model

The two processes below present the generation of synthetic data by diffusion model: 1) when generation of a full image, 2) when inpainting with noisy zones.

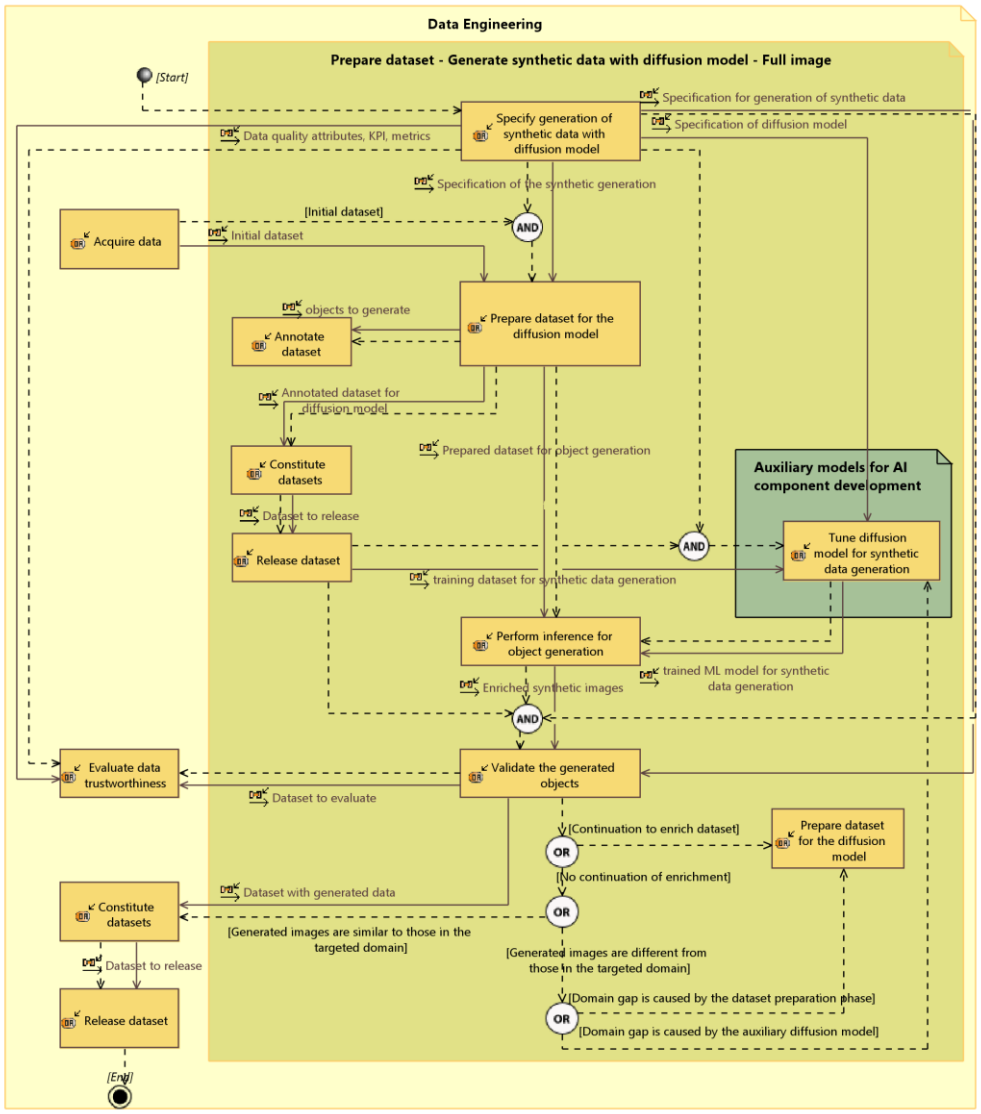


Figure 16: Process of generating synthetic data using diffusion model – Full image

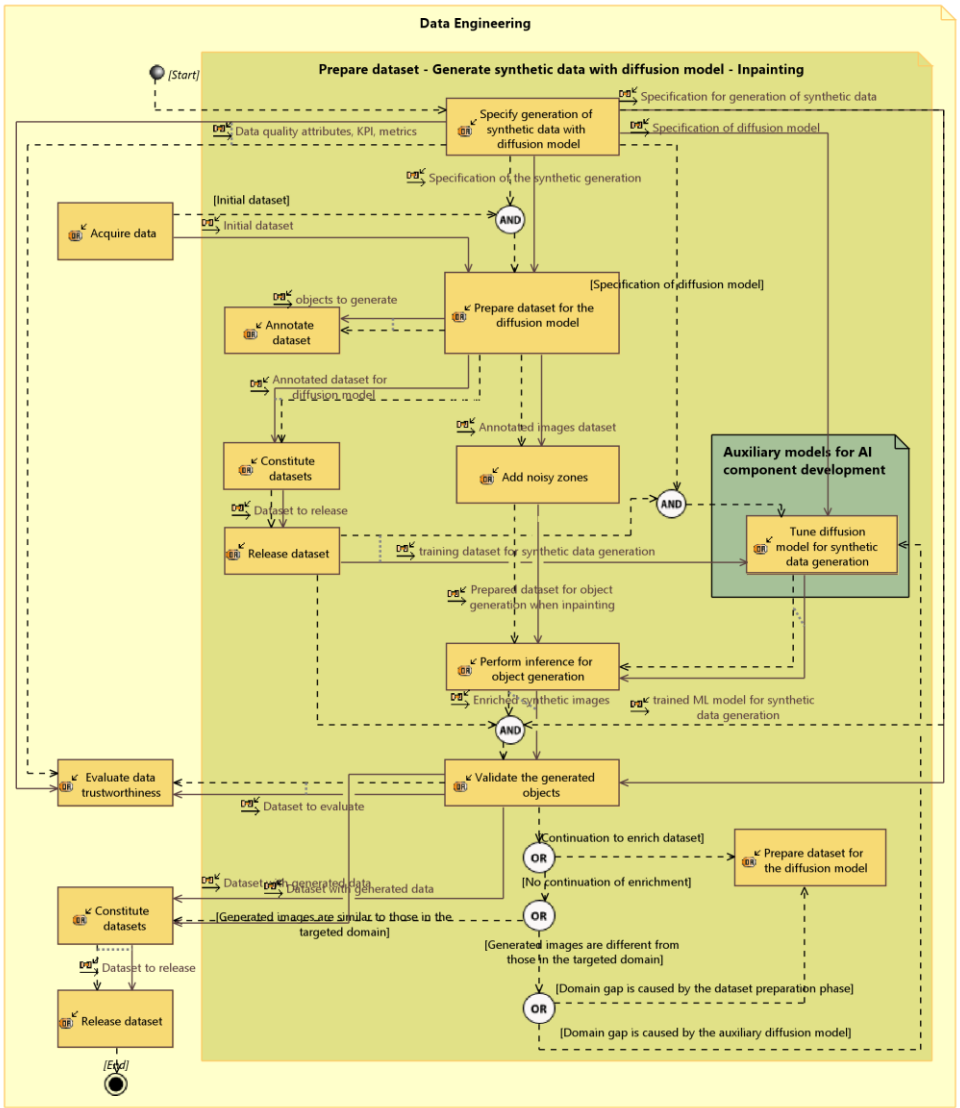


Figure 17: Process of generating synthetic data using diffusion model – Inpainting

F.3.4 Generate synthetic data – 3D model

This activity is about generation of synthetic data with 3D-model. Differences with the generic activity are the following:

- The dataset is created from scratch or from an existing 3D-model. In case of reuse, the 3D-scene is validated, for instance to check the performance.
- A catalog offers a list of off-the-shelf objects.
- The initial dataset is enriched of 3D scenes and scene scenarios.
- The inference is realized by the 3D engine which generates objects available from the catalog.
- Very often, a domain adaptation step is required to reduce the domain gap (using ML).
- Validation verifies that the scenarios are correct.

The exchanges of this activity are the following:

- IN. Data specification
- IN. Initial dataset
- OUT. Dataset with generated data

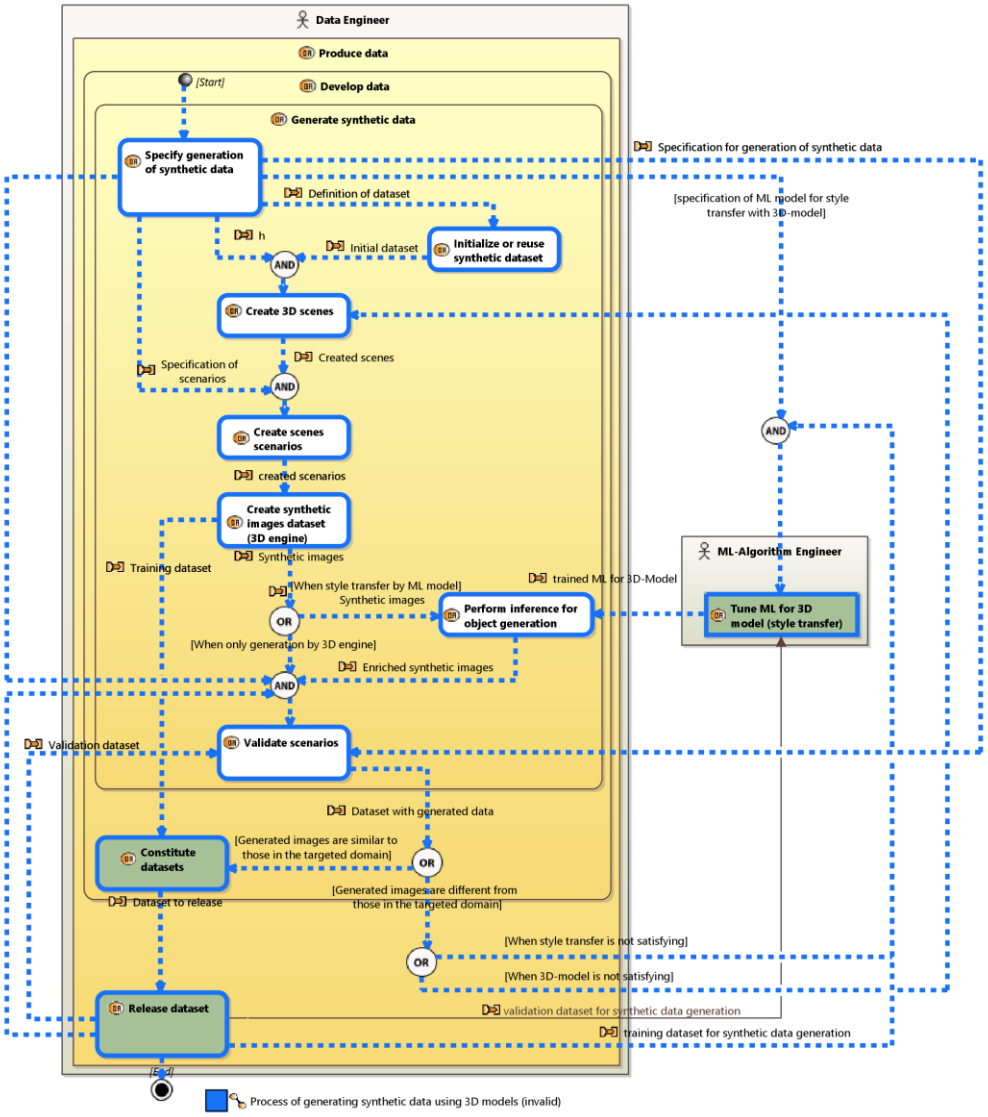


Figure 18: Generate synthetic data using 3D model

F.4 Identify missing data

On a dataset initialized by data acquisition and/or enriched with generated synthetic data, some data may be missing. To prevent releasing a dataset with missing data, a three-step process is proposed, as depicted in the following diagram:

1. *Specification steps*. Requirements, ODD and operational scenarios delimit what is a well-formed dataset (e.g., for autonomous car, there is a scene with one child running alone across the street). Those expectations are formalized and translated in formal features for objective description and measures (e.g., minimum number of representative data items in a set). Those features are expressed in the data specification document by KPI to meet. Regarding the evaluation concern, the KPI enables to characterize data evaluation, specifically here, of missing data. The KPI are then translated into metrics, for instance in terms of data completeness, diversity and representativeness with thresholds to respect. Refer to the section on evaluation of data trustworthiness.
2. *Acquisition and measure during data development*. During data development, the dataset is populated with data coming from data acquisition and/or generated synthetic data. The piloting activity solicits the data evaluation which objectively measures relevance of the dataset against the expectations, based on the metrics implemented previously (e.g., there is a scene with *one* child running alone across the street). When the fact of missing data is raised, subjective decision is taken with actions to do for improvement.
3. *Improvement when missing data.* When missing data is detected, depending on the reason, the actions of improvement are applied. Examples: 1) simply annotating, or augmenting some data items in the dataset, 2) more complex, generating new synthetic data, 3) launching a new phase of data acquisition to acquire new data, or 4) reviewing the specification of the data acquisition and the starting a new acquisition, 5) reviewing the specification of the generation of synthetic data, updating the 3-D model.

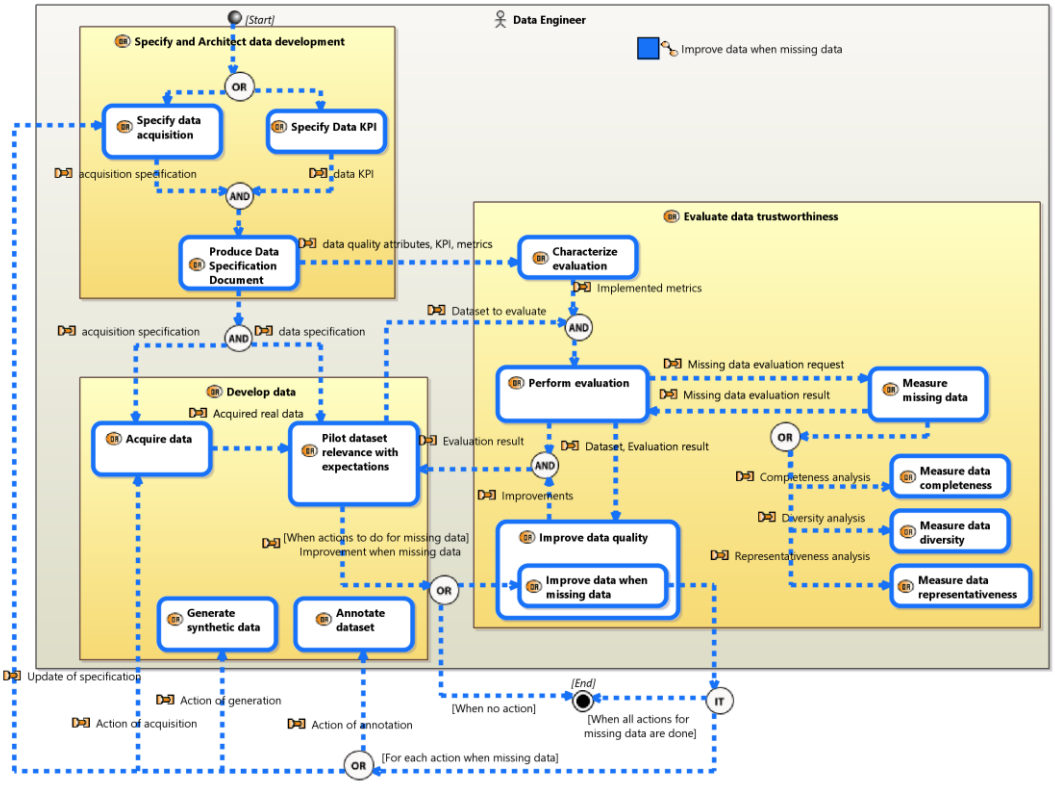


Figure 19: Process of missing data

F.5 Prepare data – Basic activities

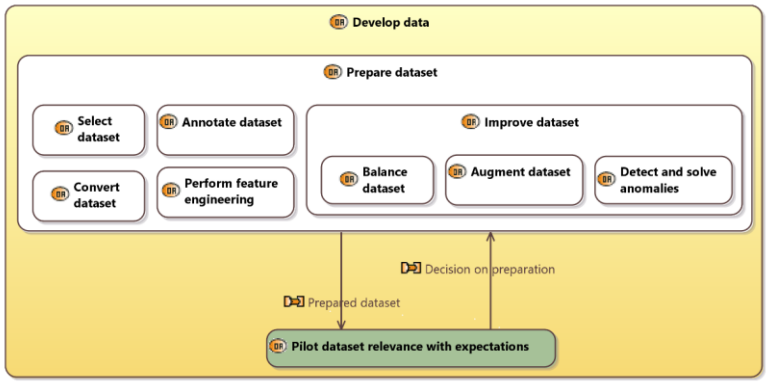


Figure 20: Prepare data activity

Obtaining the initial dataset from data sources is the preliminary but insufficient step to train and apply AI techniques. This phase consists in preparing this dataset to reach a data quality to use it in Machine Learning to train the model. The major preparation activities are: selection, conversion, annotation, feature engineering, balance / augmentation of dataset, and solve of anomalies.

At the end of data preparation, and before starting working on a model, some conditions must be satisfied [Burkov, 2020, §5.1.6]:

- The dataset is correctly labeled;
- All examples are converted into numerical feature vectors;
- Engineered features and filled missed values were engineered using only the training data;
- The dataset is baselined.

Moreover, the measurement of the dataset concerning the expectations must verify conformance with main metrics [Adjed et al., 2022]:

- Diversity: data contains information fitting various environments and provides enough information to train models.
- Representativeness: for a given population, data provides enough samples for evaluation. The issue of unrepresentativeness occurs for instance when, at the source, some data are under- or over-represented, or when, in case of frugality, it is very difficult or impossible to acquire the whished data.
- Completeness: it is expressed by the proportion of missing values, absent values and sparseness of values in the dataset.
- Coverage: it represents an evaluation of model-knowledge on the dataset, like for code-coverage.

- Corner cases: it is a situation in a dataset that occurs only when two or more operating parameters are at extreme values. It represents ambiguous examples where the results is erroneous and 'strange' for the AI/ML model.

Measurement must also ensure:

- Unbiasness in order to avoid biases introduced for instance during data specifications, acquisition, or implementation with annotations.
- Data quality, specific to the data itself, such as data consistency or reliability, or related such as ethics.

This section gives a list of the basic activities of data preparation. The next ones focus on specific topics of data preparation.

F.5.1 Select dataset

At acquisition time, raw data can be large and only a subset needs to be selected to provide the initial dataset of interest. For instance, from a 20-minute video, only 5 are interesting. At preparation time, for a given purpose, only 2 minutes are relevant to train an ML model. Then, a new selection activity removes and keep only a subset of the initial dataset. Examples of selection usage: • Keeping relevant records. • Cleaning incorrect records. • Removing corner/edge case situations in order to consider only standard scenarios, or the contrary to work on corner/edge cases.
The exchanges of this activity are the following: • IN. Dataset • IN. Selection criteria • OUT. Selected dataset

F.5.2 Convert dataset

This activity encompasses all the operations of transformation on a dataset. Examples: • Standardization, normalization of the dataset values. • Data massage can be considered as such in order to correct and standardize values (e.g., trim extra-words or characters, respect of zip code encoding) in series.

The exchanges of this activity are the following:

- IN. Dataset
- IN. Convert instructions
- OUT. Converted dataset

F.5.3 Balance dataset

In classification, a dataset is said imbalanced when the number of samples representing a class is much smaller than the ones of other classes. Imbalancing have generally a detrimental impact on the performance of classification learning algorithm. Training such algorithms with an imbalanced dataset results in biased classification towards the most represented classes since the training process tends to treat under-represented class samples as noises and thus give them lower weights. This is especially bad when trying to detect low occurrence events.

Methods to alleviate an imbalanced dataset at:

- The data level by undersampling, oversampling, data augmentation or synthetic oversampling methods;
- The algorithm level by weighted losses, or negative and hard mining methods.

The exchanges of this activity are the following:

- IN. Dataset
- OUT. Balanced dataset

F.5.4 Augment dataset

The need for data augmentation comes from several reasons: some data are under-represented, missing, non-existent in reality or very rare, or simply to scale data figures according forecast. The causes are many. For instance, some information is sensitive, some data is not or badly captured by humans and sensors, lost, frugality, or must be simulated (e.g. in the defense or nuclear sectors to create feared events, or for domain adaptation). The interest of data augmentation is: 1) from the operational point of view, to improve the model prediction accuracy and to reduce the costs of collecting and labeling data, 2) to improve the quality and relevancy of the dataset.

The data augmentation increases artificially the size of the original dataset by adding an existing dataset with appropriate variations. Different techniques exist and according to the types of data and transformations. Examples on image: changing the color, applying blur, flip, rotation, brightness. On text: introduction of synonyms, word insertion or deletion, change of punctuation. Data augmentation concerns as much the creation of new records, such as new sentences in a text, as the enrichment of existing data, such as the embedding of characters in a photo. Another technique is to add data with synthetic data with

new samples for the under-represented classes. Refer to the specific section on generation of synthetic data.
The exchanges of this activity are the following: • IN. Dataset • OUT. Augmented dataset

F.6 Annotate dataset

Labeled data are data for which each object has an identified target value, the label. Labeled data are used in supervised learning. They stand in contrast to *unlabeled data* that are used in unsupervised learning [Sammut and Webb, 2017][1] A *label* is a target value that is associated with each object in training data. In classification learning, labels are classes. In regression, labels are numeric [Sammut and Webb, 2017][2]. Data labeling, or tagging, is the process to associate descriptive labels, a.k.a tags, to data in order to take advantage of the supervised learning setting. Labels correspond to expected AI model outputs for a given input data. These labels mostly correspond to annotations with their attributes defined during the data specification. Their help to monitor, browse and filter the data in order to analyze and extract specific situations, context and conditions such as location, weather. Some additional labels, such as sensor calibration, can be mandatory. Some labels can be automatically generated (e.g. time, geolocalization). Usual techniques to label data: 1. *Use existing labels*: An early idea of data labeling is to exploit any labels that already exist. Cf. semi-supervised learning where the aim is to learn from the labels to predict the rest of the labels. 2. *Crowd-based* i.e., a simple approach is to label individual examples. A more advanced technique is to use active learning where questions to ask are more carefully selected. More recently, many crowd sourcing techniques have been proposed to help workers become more effective in labeling. 3. *Weak labels*: While it is desirable to generate correct labels all the time, this process may be too expensive. [1] [Sammut and Webb, 2017] *Labeled Data*. https://link.springer.com/referenceworkentry/10.1007/978-1-4899-7687-1_439 [2] [Sammut and Webb, 2017] *Label*. https://link.springer.com/referenceworkentry/10.1007/978-1-4899-7687-1_438
The exchanges of this activity are the following: • IN. Dataset • IN. Annotations • OUT. Annotated (labeled) dataset

F.7 Perform feature engineering

Feature engineering is a means to increase Machine Learning efficiency. The representation of information through features contributes to shape the structure of a dataset.

A feature is a numeric representation of raw data. There are many ways to turn raw data into numeric measurements [..]. Feature engineering is the process of formulating the most appropriate features given the data, the model, and the task [Zheng and Casari, 2018]. This process involves the initial discovery of features and their stepwise improvement based on domain knowledge and the observed performance of a given Machine Learning algorithm over specific training data [Burkov, 2020]. It is the process of adjusting the representation of the data to improve the efficacy of the Machine Learning algorithms. [..] Feature engineering is difficult, expensive and time consuming [Burkov, 2020].

Feature engineering implies [Géron, 2019]:

- Feature selection, by selecting the most useful features to train on among existing features
- Feature extraction, by combining existing features to produce a more useful one-as we saw earlier, dimensionality reduction algorithms can help
- Creating new features, by gathering new data

Good properties of the features are high predictive power, fast computability, reliability, uncorrelatedness [Burkov, 2020].

To scale features, features must share the same scale. Common techniques are normalization (by rescaling the features to a range of [0, 1]) and standardization (by centering the feature columns at mean 0 with standard deviation 1) [Burkov, 2020].

The exchanges of this activity are the following:

- IN. Dataset
- OUT. Annotated (labeled) dataset

F.8 Detect and solve anomalies

This activity consists in detecting anomalies and possibly to solve them. Manual anomaly detection can be tedious and time-consuming, in case of either complexity or scalability. To help the data engineer, an anomaly detection model is a means to assist him/her. Coupled with the active learning, the data engineer is driven to easier correct anomalies. This is exemplified here on time series.

The exchanges of this activity are the following:

- IN. Dataset
- OUT. List of detected anomalies (without correction)
- OUT. Warning when an anomaly occurs

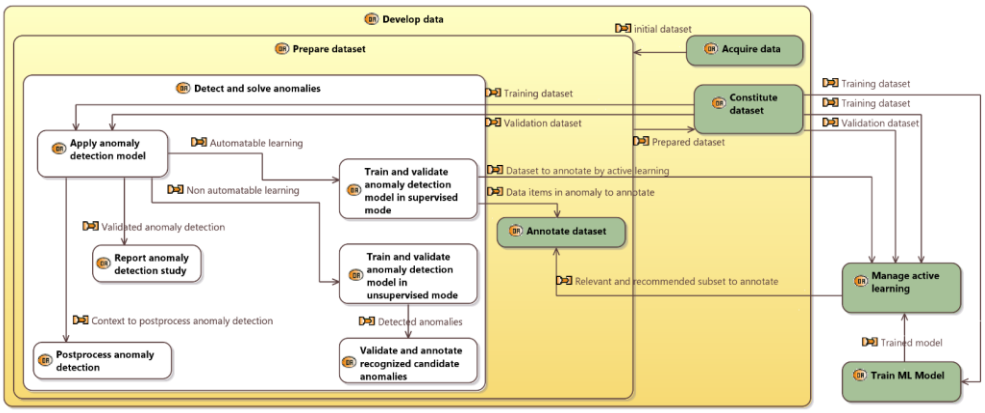


Figure 21: Process of anomaly detection

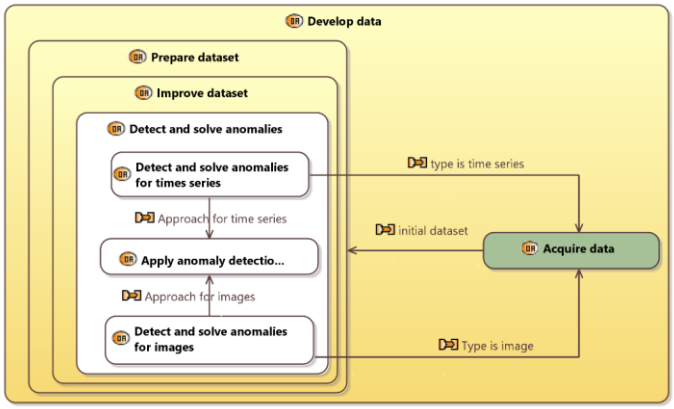


Figure 22: Process of anomaly detection for an object type

F.8.1 Detect anomaly in time series

For a given use case (e.g. on time series with sensors), an anomaly is an abnormal behavior in a time series (TS). Following data collection, the origin may be due, for example, to a change of context, to a

defective sensor, to a malfunction in data feedback to the server. A detected anomaly is either corrected or removed. According to a standard model of forms of behavior, an anomaly corresponds to an irregularity of form.

Typology of anomalies: an anomaly relates to the collection process, from the ground truth, to a drift over time.

Learning mode:

- *Supervised*: ideally, error management is automatable (this depends on the maturity of the UC), with the possibility of applying Active Learning (AL).
- *Unsupervised*: an expert validates, on a given case, whether an anomaly is really one; if so, it categorizes it by annotating it. This knowledge is acquired to iterate over the dataset.

This process makes it possible: 1) to switch from unsupervised mode to semi-supervised mode, 2) to apply AL.

Learning about the treatment of anomalies:

- Anomalies are annotated with new categories
- The model is improved by optimization of calibration (better precision at the level of thresholds by evaluation on the basis of scores).

Models are based on probabilities or ML for the purposes of 1) anomaly detection, or 2) monitoring.

Types of detection approaches:

- Change-point detection
- Contextual deviation analysis for anomaly detection
- Anomaly detection using 1D-CNN
- Anomaly diagnostic based on counterfactual analysis

Method maturity:

The expert measures the maturity of the method on a use case. The approach is more widely validated when it is successfully applied to several use cases using labeled public datasets. In this second case, the expert is no longer involved.

Constitution of the datasets:

For the constitution of the training/validation/test datasets, the method is rather to limit each dataset to a past period. The difficulty lies in the fact that the collection regimes can vary over time and that the distribution of anomalies between datasets is homogeneous.

Pipeline:

The processing is carried out according to three phases: pre-processing (preparation), application of the model, post-processing dedicated in particular to the construction of anomaly scores for the detection of thresholds, and to the calibration of the model.

The exchanges of this activity are the following:

- IN. Dataset

- OUT. Annotated dataset
- OUT. Augmented list of annotations

F.8.2 Detect anomaly in time series with active learning

The principle consists in merging 1) the detection of anomaly in time series in supervised mode, described in the previous section, 2) and AL described page 57, that is in using an acquisition function, providing a relevant subset to annotate by the domain expert, applying a new training on the newly annotated dataset, and evaluating whether performance with AL is improved to stop or continue AL.

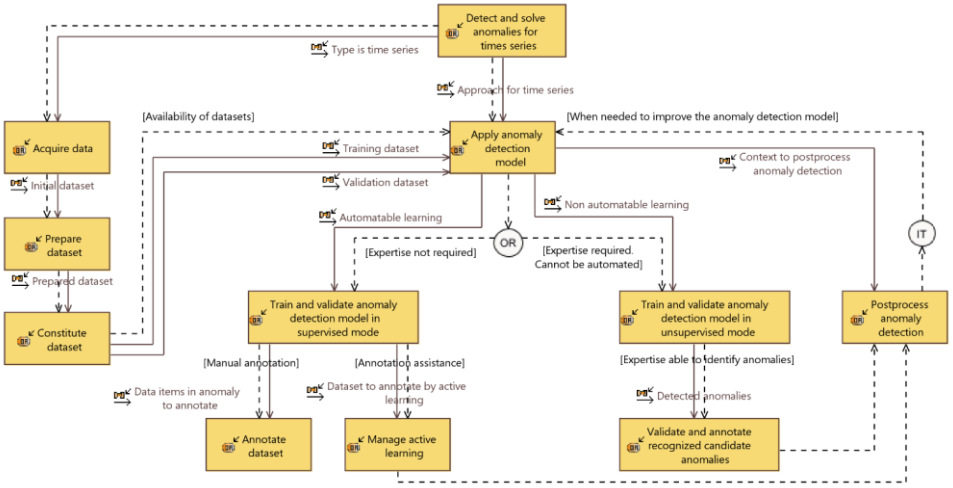


Figure 23: Process of anomaly detection for time series with active learning

F.9 Prepare data – Techniques

This section presents techniques usable during the data preparation.

F.9.1 Apply active learning

The objective of an active learning (AL) algorithm is to select the most useful data to be annotated in order to maximize the performances of a given model whilst keeping annotation costs to a minimum [Granger et al., 2022].

The term *Active Learning* [1] is generally used to refer to a learning problem or system where the learner has some role in determining on what data it will be trained. This is in contrast to *Passive Learning*, where the learner is simply presented with a training set over which it has no control. Active learning is often used in settings where obtaining labeled data is expensive or time-consuming; by sequentially identifying which examples are most likely to be useful, an active learner can sometimes achieve good performance, using far less training data than would otherwise be required [Sammut and Webb, 2017].

A smart selection for a type of data (e.g. image, text, video) is assisted, for instance with a uncertainty or trustworthiness score to select and label data elements, in order to cover as much as possible the application domain.

[1] [Sammut and Webb, 2017] *Active Learning*. https://link.springer.com/referenceworkentry/10.1007/978-3-030-10576-1_300008

In more details, active learning (AL) aims to facilitate the task of the dataset annotator (e.g. a domain expert) by offering him/her a relevant set of dataset elements to annotate. A major problem is the constitution of this set of elements. For this, a heuristic is to be determined in order to group/distinguish data for data annotation. This step is carried out through an acquisition function:

- The naive AL strategy (simplest heuristic) consists of randomly drawing data and applying the AL until satisfied.
- Another preferred strategy is to rely on criteria such as diversity (e.g., uniform distribution), model confidence, or consistency. A weakness is that the dataset is not always well balanced.

The imperfect model (to give the relevant information on the information to be annotated) is improved until satisfaction. The criterion for stopping annotations corresponds to the level of performance confirmed by the annotator. A validation dataset, manually annotated, makes it possible to compare the dataset annotated by AL to check whether the dataset is correctly annotated automatically.

To apply the AL strategy, the architecture of the ML model is known as well as its behavior, for example by attaching to a sub-layer a deep learning model.

Steps Applied:

1. Acquisition (AL)

Input for acquisition function:

- The non-annotated training dataset

- The annotated training dataset so far (not used at the SotA level)
- The ML model for the considered task (e.g., detection, classification)

Treatment :

- Apply the model to non-annotated data (even annotated for experiment)
- Retrieve predictions and internal representations
- Apply the chosen heuristic

Output:

- Dataset to annotate for recommendation (= dataset with a filter applied on)

2. Annotation of data selected by the acquisition function (AL)

Cf. Annotate Dataset Activity, Page 52.

Input:

- Set of data to annotate (identified previously)

Treatment :

- Apply annotations (=Annotate Dataset Activity)

Output:

- Data set to annotate

3. New training with newly annotated data

4. Approval

Input:

- Stopping criterion for AL
- The annotated dataset validation
- The ML model (re-trained by the previous task) for the considered task

Treatment:

- Inference (application of the model on the dataset)
- Calculation of the performance metric

Output:

- Decision according to the result of the metric: if satisfactory, then stop, otherwise go back to step 1

Notes for improvement:

- At the design stage: update of the labels, with impacts at the implementation level to automate the update.

- At the implementation stage: automation to update the labels, update of the training / validation / test models.

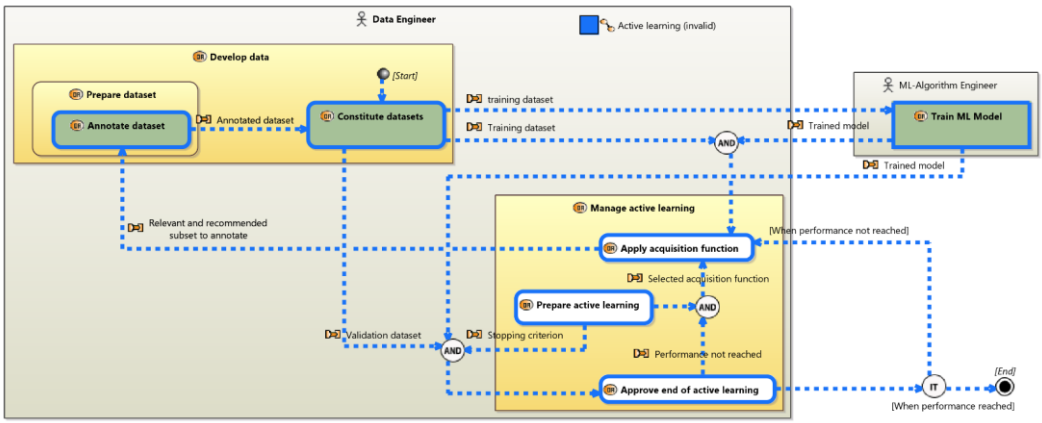


Figure 24: Active learning activity

F.9.2 Apply domain adaptation

The goal of Domain Adaptation (DA) is to reuse a model for a new task not initially planned, with the stake to keep a good level of performance when a trained domain is applied to a new one, or/and when the context changes (e.g. adapting to driving a vehicle in a new country). Impacts are at the level of the Develop Data, for instance with a revision of the labels, or even at the specification level, for instance with the introduction of new annotations. The challenge is to minimize the manual cost when moving to a new domain.

There is no general method for DA. The choice of a method depends on the considered scenario (example for images: day-night transition, change of location). The level of maturity is still empirical enough to proceed with data characterization (e.g. recognition of objects and their characteristics for an image) and consequently to measure the differences between source and target domains. Quantification is left to the intuition of the data scientist. However, there are perfectly known methods for carrying out DA. Ex: GAN, image style (ex: Fourier transformation), adversarial methods, pseudo-labeling followed by relearn.

The general method for DA is as follows:

- First phase: Domain gap analysis:
  - Identification/analysis of the scenario:

- Prerequisite: the source domain is known
- Data characterization:
  - Quantity of data, quantity of annotations (from many to none) on the target domain
  - Nature of the inputs (e.g., sensor, its sensor characteristics, content of the image which changes), nature of the outputs (e.g., new classes of objects, transition from segmentation to detection)
- Qualification of the nature of the gap (scientific intuition today, absence of methodology) and choice of a resolution method

- Second phase: Application of the domain adaptation (run)

DA can be associated to Active Learning to ease adaptation of the dataset.

Examples of tackled topics in Confiance.ai [Loesch et al., 2023]:

- Continual Unsupervised Domain Adaptation for Semantic Segmentation. Used methods: RPS Net, Training strategy.
- Domain Adaptation in Object Detection. Used methods: FDA-based image translation, Adversarial training, Pseudo-labeling.
- Domain Adaptation for label shift. Used methods: Regularized Learning under Label Shift (RLLS), Importance Weighted DA under Generalized Label Shift (GLS), Label matching deep domain adaptation (Lamda).
- Source dataset construction for Person Re-Identification. Used methods: Features extraction, Dimensionality reduction, Source selection as a function of target distribution.
  Domain adaptation in autonomous driving (Valeo), with fish-eye image understanding. Study at two levels: 1) knowledge transferring from synthetic data to real data, and 2) robustifying detection model for domain-aware active learning.

The exchanges of this activity are the following:

- IN. Model before domain adaptation
- IN. Dataset on a domain different from the training dataset
- OUT. Model after domain adaptation

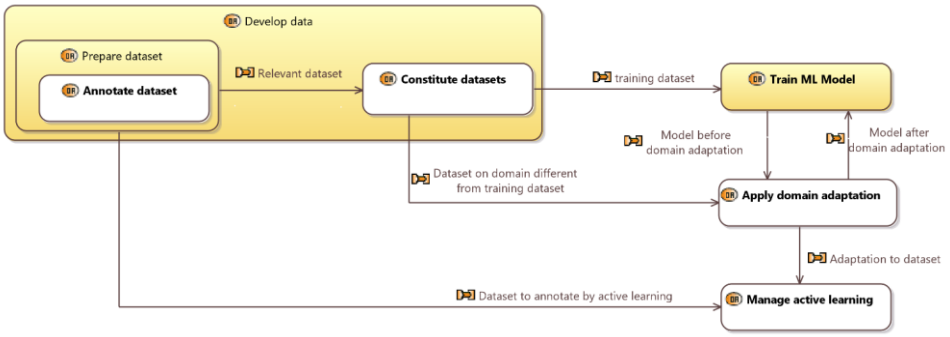


Figure 25: Domain adaptation activity

F.9.3 Apply incremental learning

Positioning of Incremental Learning (IL)

A model must be able to adapt to new datasets according to a learning scenario (e.g. new examples, new classes, new domain). Three approaches are possible:

1. Method: joint / offline at T0. For a first or completely new training (from scratch), training or retraining is expensive but it offers the best learning results. This is the preferred method. The training is carried out in one time on a large dataset (baseline). The term Joint means once for all tasks.
2. A second possibility is to specifically adapt an existing model by fine-tuning. However, it is possible to face the problem of *catastrophic forgetting*, that is the tendency to abruptly and drastically forget previously learned information upon learning new information [McCloskey and Cohen, 1989].
3. IL is characterized by the fact that all the training data is no longer necessarily accessible (e.g. by application of the GDPR law) and it is not possible to carry out new training (e.g., impossibility to annotate a dataset on new classes of data when previous data are inaccessible due to losses). This is the case for instance for embedded systems where the processing capacities are limited. Retraining is then carried out incrementally from the previous set of reduced size. The problem is to keep the same performance, both on the new data and on the previous data.

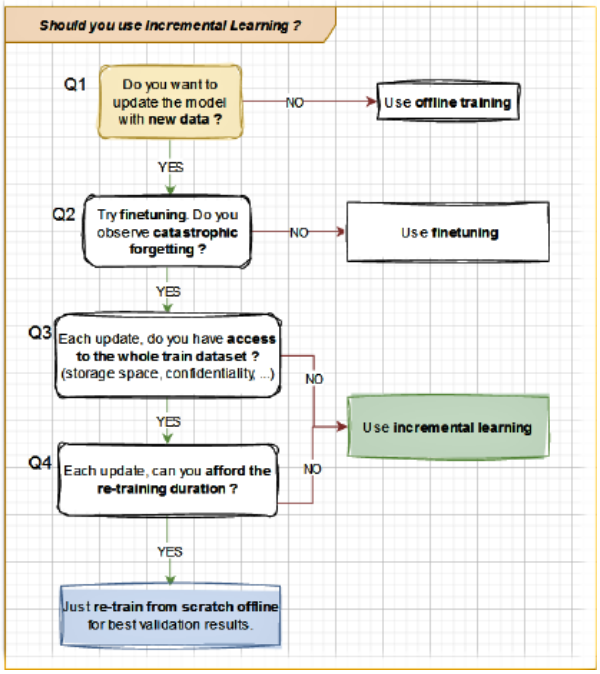


Figure 26: Decision tree to select the incremental learning technique

Picture from [Lerbourg, 2023]

It is possible to classify IL into three categories:

- Replay (a.k.a. rehearsal): only a small number of previous data is kept, for example in a buffer.
- Regularization: when calculating the loss, the goal is to reduce the learning speed on new data in order to less forget the old ones.
- The architecture of the network is modified by neurogenesis (parameter isolation).

IT can be used to push the limits of classic ML learning or fine-tuning. Its interest is to be combined with other techniques. For example :

1. IL and active learning (AL). AL makes it possible to detect what needs to be relearned. IL avoids complete relearning.

2. IL and Domain Adaptation. In the case of adaptation from one domain to another, fine-tuning can be sufficient by learning new characteristics, but with the risk of catastrophic forgetting.

The exchanges of this activity are the following:

- IN. New dataset items
- IN. Trained model for incremental learning
- OUT. New model by incremental learning

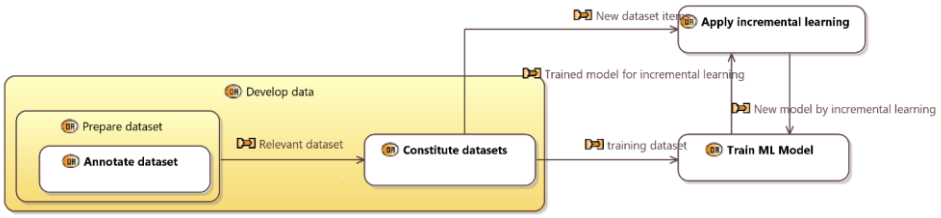


Figure 27: Incremental learning

F.10 Constitute datasets

The prepared and finalized dataset, before test and release, is the base to constitute several datasets, each with its own role. The dedicated term in Machine Learning is to “split”, or segregate, it into three new datasets:

1) The training dataset is used to initially train the algorithm and teach it how to process information. This set defines model classifications through parameters, establishing the behavior of the Machine Learning model.
2) The validation dataset is used to evaluate the capacity of generalization of the model.
3) The test dataset is used to assess the accuracy and performance of the models, independent of the training dataset. This set is meant to expose any issues or mistrainings in the model. The test dataset is used during the evaluation of the model before its release.

Usually, the training set is the biggest one, while the validation and test sets are roughly the same size, much smaller than the size of the training set [Burkov, 2020, p.15]. Indeed, this common split practice presupposes that data is homogeneously scattered, what is not always the case. Moreover, in case of cross-validation [Sammut and Webb, 2017, p.306], the dataset can be split in more than three sets.

The problem is to distribute correctly the data in three or more sets, how to distribute wisely and with balance the data of the regular / anomaly cases. There are many strategies to do that, four of the most common ones are:

- Use a default or custom ratio to split it into the two subsets, sequentially i.e. in the order it appears in the source, making sure there is no overlapping. For instance, use the first 50% of data for training, the subsequent 30% of data for testing and the remaining 20% for testing.
- Use either of the methods above (sequential vs. random) but also shuffle the records within each dataset.
- Use a custom injected strategy to split the data, when an explicit control over the separation is needed.

To obtain a good partition of the entire dataset into training, validation and test sets, the process of partitioning has to satisfy several conditions: 1) data was randomized before the split, 2) split was applied to raw data [and synthetic data], 3) validation and test sets follow the same distribution, and 4) leakage was avoided [Burkov, 2020, §3.13]. If this is not the case, an algorithm of distribution must ensure the right balance of data between the numbers of datasets.

The exchanges of this activity are the following:

- IN. Dataset to segregate
- OUT. Homogeneous distribution of data between training, validation and test dataset

F.11 Execute development data pipeline

A pipeline is a sequence of transformations with the purpose to automate tedious and recurrent actions, to replay them safety as many as whished in the same manner, and to detect regression in an automated chain as soon as possible. In this perspective, DataOps automates tasks to improve data quality, but also speed, and collaboration between teams in the data area with the culture of continuous integration and improvement. DataOps is related to MLOps, which focus on ML models.

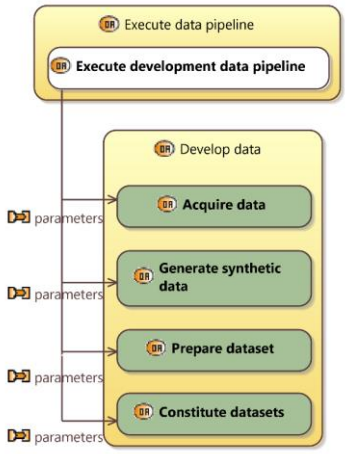


Figure 28: Execution of data pipelines

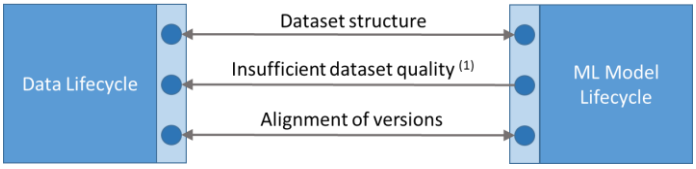
The development data pipeline automates recurrent development data activities. All data activities, from Acquire data / Generate synthetic data / Prepare dataset / Constitute datasets, are potentially candidate.
The exchanges of this activity are the following: • IN. Pipeline parameters • IN. Data, datasets • OUT. Execution report

F.12 Improve developed data

A need of improvement during the Develop data phase can come from: • The phase itself from anomalies detected during regression tests, or during the different activities of Develop Data such as the data augmentation or balancing. • Metrics results report that some KPI of data quality attributes are not achieved. Metrics on data are a means to detect anomalies on a dataset: diversity, representativeness, completeness, coverage, corner cases, etc. • The acquired data / generated synthetic data is incomplete or too poor. • Developed data does not respect architecture rules or specification.

From model development. Data and algorithms are entangled, and quality of the models depends on well-formed data. A correct model assessment, such as robustness, depends on the data quality.

- Incompatibility or evolution of data and model, or vice versa, in terms of dataset structure and expected features by the model requires a synchronization between data and model, in one or the other side.
- Insufficient dataset quality:
  - Metrics not met: diversity, representativeness, completeness, coverage, corner cases.
  - Data quality attributes not satisfied (e.g., data privacy, data consistency).
- Alignment of versions.



From downstream phases:

- A feedback of correction or improvement was raised during IVVQ or deployment.

To upstream phases:

- Specify and Architect data. Data specification is incomplete or has a lack of design (e.g., problem of data acquisition specification).
- Specify and Architect data. Functional data architecture rules are unsuitable (e.g., data governance, definition of KPI).
- Specify and Architect data. Non-functional data architecture rules are unsuitable (e.g., undersized communication infrastructure for data flow, data visualization tool).
- Orient data. There is a lack in the expectations (e.g., inconsistent, incomplete requirements, imprecise operational scenarios).

The exchanges of this activity are the following:

- IN. Improvements from Develop Data itself
- IN. Feedback from model development
- IN. Feedback from IVVQ, deployment
- OUT. Improvement for Develop data
- OUT. Improvement for Specify and Architect data
- OUT. Improvement for Orient data

G. Evaluate Data Trustworthiness

Objective:

- Data measurement ensures that data expectations, data quality, and finally data trustworthiness are met

Roles in charge:

- Data Engineer

Inputs:

- For definition of the data evaluation:
  - Data quality attributes fitting use case/project needs
  - Data KPI
- For evaluation:
  - Dataset

Output:

- For evaluation:
  - Evaluation report
- After evaluation:
  - Actions of improvement

For a good understanding of trustworthiness and quality attributes, refer to [Sohier et al., 2023].

Data Quality is defined as:

- [ISO/IEC 25024:2015, 2015] Degree to which the characteristics of data satisfy stated and implied needs when used under specified conditions.
- [Mamalet et al., 2021] The extent to which data are free of defects and possess desired features.

Evaluation of the data trustworthiness indicates the degree to which data and data items satisfy expectations. Evaluation is applicable at different times of the process, usually during data development (e.g. raw data, dataset preparation), but also during IVVQ and deployment (e.g., for data drift).

Three activity groups mark out evaluation of the data trustworthiness, as depicted in the diagram below (on the right side of the diagram):

1. *Characterization of the evaluation*. Data orientation defines the objectives and expectations to achieve; Data specification and architecture set the KPIs to achieve. From those inputs, characterization defines quality attributes, their organization and how to assess them with metrics. At the end, metrics are implemented.

2. *Data evaluation*. In an evaluation context (e.g., data preparation), data evaluation consists in applying evaluation with metrics on a given dataset and reporting the evaluation results. The interest and focus of evaluation varies with the context and objectives to achieve. For instance, data representativeness is evaluated during development with the concern to prepare a dataset before training, while data drift is a monitoring indicator during deployment.
3. *Data quality improvement*. Metrics provide an objective result. A diagnosis analyses this result in its context (e.g., technical issues, progress over the time and previous improvements, user priorities) and provides a conclusion that is, in comparison, a subjective result. Then, all the elements are available to decide on improvement actions to be taken, such as tuning the dataset with parameter adjustment (e.g., balancing data), or cascading updates at an upstream level (e.g., updating the conditions of acquisition during the Data architecture and design stage when some important objects are missing in a video/image). This step contributes to continuous data improvement throughout the data lifecycle.

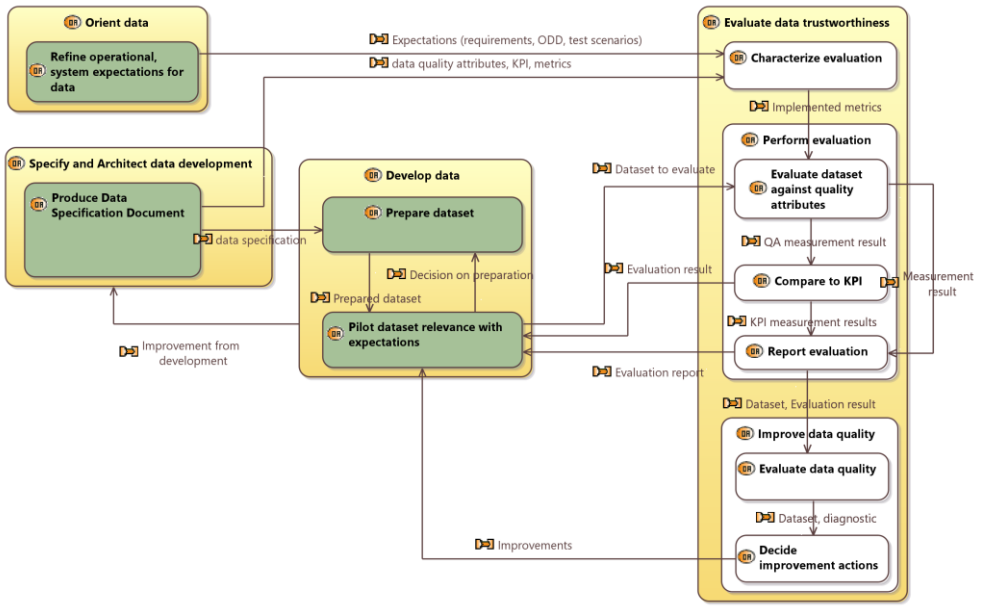


Figure 29: Evaluate data trustworthiness

Operationally, those activity groups are broken down in activities organized in six periods as shown in the following process, illustrated in the context of data preparation, which are:

For preparation of the evaluation:

1) Characterization of the quality attributes for the evaluation,

2) Implementation of the characterization with metrics,

For evaluation:

3) Execution of an activity which solicits the data evaluation (e.g., data preparation),

4) Data evaluation,

For actions after the evaluation:

5) Diagnosis, and decisions of improvement actions thanks to the diagnosis.

6) Application of the actions in respond to the evaluation result and the made decisions.

The following diagram mixes two processes: the first one, in blue, concerns the preparation of the evaluation; the second one, in red, the evaluation and the actions of improvement for convergence of the dataset to meet the KPIs that translate the satisfaction of the quality attributes.

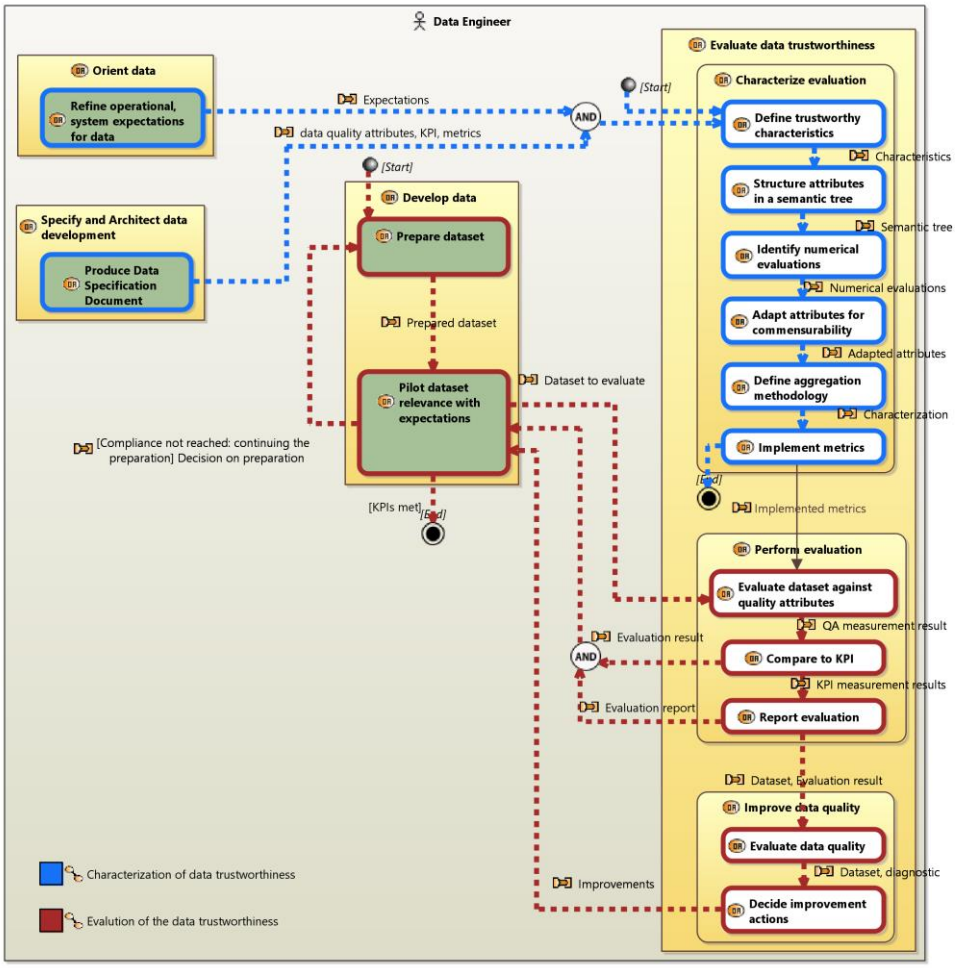


Figure 30: Evaluate data trustworthiness

G.1 Characterize evaluation

G.1.1 Characterize quality attributes

KPIs, formalized during the Specify and Architect data phase, define major indicators to meet for a system. Technically, they are translated and decomposed in quality attributes to measure the trustworthiness of a system with different scores. In the context of what is to be measured and who is using it (e.g., customer, manager, developer), a quality attribute has different purpose and meaning:

- The business quality attributes ensure to keep under control development, IVVQ and deployment phases of a system (e.g., for risk management). Refer to the Orientation phase.
- The operational quality attributes validate that operational requirements and ODD are met (e.g., safety measures). Refer to the Orientation phase.
- The technical quality attributes assess technical objectives, for instance identified during the Specify and Architect data (e.g., quality of acquisition) and and Develop data phases (e.g., threshold to declare that a dataset meets expectations).

Characterizing data trustworthiness with data quality attributes is not trivial. Quality attributes must be identified and organized in order to compute and compare them. For foundation, we reuse MCDA (Multi-Criteria Decision Aiding) introduced in [Delaborde et al., 2022]. MCDA is a generic term of systematic approaches for decision makers to assess or compare alternatives on the basis of criteria. It is compatible with data evaluation when the notion of criterion stands for quality attribute. Characterization of data evaluation is divided in five steps according to [Delaborde et al., 2022], reused and detailed below.

Quality attribute values are quantitative or qualitative to measure them and to assess that the targeted objectives are reached or not. In case of complex system and complex measures, a quality attribute is broken down in sub-attributes.

1. Define trustworthy characteristics

The characterization and evaluation of trust attributes focus on the definition and structuring of the attributes that constitute trust in the context of AI-based critical systems [Delaborde et al., 2022]. In the data context, data quality attributes are for instance data accuracy, completeness, integrity, lineage, timeliness, or traceability.

2. Structure attributes in a semantic tree

In a project context, the number of criteria may be large. This phase consists in structuring assessment as a tree, where:

- The root is the overall evaluation;
- The leaves are the elementary attributes.

A classification makes sense to the stakeholders. For instance, according to the focus of a project, a node Ethics can hold data privacy, fairness, and benevolence as sub-nodes.
3. Identify numerical evaluations All nodes return a numerical evaluation. Specific KPI, metrics or evaluation methods are used to qualify the leaves of the tree according to the use cases [Delaborde et al., 2022]: • KPI, identified in Specify and architect data, define targets to meet; • Metrics identify: 1) the formulas to compute a metric, 2) the consumed (data attributes) nodes by the formulas.
4. Adapt attributes for commensurability Quality attributes are quantitative or qualitative (with a numeric equivalence), respect or not the same value domains and scales, with different weights of importance. One work consists in making them comparable, with necessary transformations, to compute (e.g. with min/max, arithmetic mean, weighted sum, etc.) and validity on each node of the hierarchy.
5. Define aggregation methodology This step is optional. Indeed, MCDA comes with the fact that the global assessment would be made on the basis of several trustworthiness attributes [Delaborde et al., 2022]. The interest is to make comparison and compromises (i.e., tradeoffs) between different sets of data evaluation and to opt out the best solution. This requires a final aggregation method to apply computations (e.g., with application of weight, normalization), comparison, and representations (e.g., chart, table).
The exchanges of this activity are the following: • IN. Data quality attributes • IN. Data KPI • OUT. Characterization of the data quality attributes • OUT. Data metrics ready for implementation

G.1.2 Implement metrics

This activity is the implementation of the characterization defined previously with the purpose to evaluate data(set) by quality attributes. For reliability on measures, some issues to prevent data are [Géron, 2019][Mattioli et al., 2021]:

• Size too small; • Poor-quality of the data (e.g. noise, pollution), either at the source during the acquisition, the preprocessing or due to the implementation (e.g. with wrong annotations); • For Machine Learning: unrepresentative, underfitting (model too simple) and overfitting (model too complex) of the dataset; irrelevant features; • Misalignment between available data and requirements.
The exchanges of this activity are the following: • IN. Data metrics ready for implementation • OUT. Implemented data metrics

G.2 Perform evaluation

G.2.1 Evaluate dataset against quality attributes

During development or at runtime, implemented metrics evaluate data quality on data(sets).
The exchanges of this activity are the following: • IN. Implemented data metrics • IN. Dataset • OUT. Measurement results

G.2.2 Compare to KPI

Measurement results are compared to the KPI.
The exchanges of this activity are the following: • IN. Measurement results • OUT. KPI measurement results

G.2.3 Report

Measurement results are reported in different ways, for instance graphically, in tables, flat files, or sophisticated documents.

The exchanges of this activity are the following:

- IN. Measurement results
- IN. KPI measurement results
- OUT. Report

G.3 Improve data quality

G.3.1 Diagnose data quality

Data evaluation is the means to objectively know the level of data quality. Before making decisions, a diagnosis is necessary. A diagnosis is a subjective evaluation to understand the causes of unsatisfied quality attributes according to evaluation results, technical elements, KPI, and expectations. Decisions depends on the context.

The exchanges of this activity are the following:

- IN. Dataset
- IN. Evaluation result
- IN. KPI
- IN. Expectations
- OUT. Diagnostic on dataset

G.3.2 Decide improvement actions

The diagnosis allows actions to be taken to improve the evaluated data(sets).

Decisions of improvement actions depend on:

- The context of evaluation (e.g., IVVQ purposes are different from dataset development; availability of all elements of evaluation; right respect of evaluation conditions; technical complexity; completeness of impact analysis),
- The objective of evaluation (e.g., evaluation of a precise dataset property such representativeness vs. evaluation of a high-level quality attribute; respect of KPI which reflect requirements or ODD),
- The objectives and risks to respect (e.g., in a project some quality attributes are priority while others are secondary).

In a work context, improvement actions depend on a problem-resolution mapping, generally informal. For instance, what to do for underrepresented objects in a video/image? When it is just a problem of balance, data augmentation in the dataset can fix the issue. However, for a video, this may require a new acquisition. Here, underrepresented objects is the problem. Data augmentation and data acquisition are two solutions of resolution.

Improvements actions can cascade in upstream stages (e.g., the specification of acquisition to augment the number of a kind of objects in a video).
The exchanges of this activity are the following: • IN. Dataset • IN. Diagnostic • IN. Actions of improvement • IN. Expectations • OUT. Decisions

H. Phase: Test the dataset

Objective:

- Testing, before release, ensures that the dataset is of good quality and reliable enough to be consumed by ML models

Roles in charge:

- Data Engineer

Inputs:

- Data expectations, including the requirements, the ODD, and operational scenarios applicable to the data intended for the ML model
- Data specification including the test strategy
- Dataset to test

Output:

- Test Plan
- Definition of the test plan
- Test results

The purpose of testing is to ensure validation of the datasets. This is the functional point of view of testing, and later, at the system level, the concern of the IVVQ engineer. However, to be complete, tests also involve validating the means of developing datasets down to the data platform. The rest of this section must be read at those two levels. In the following figure, which depicts the testing process, this is particularly visible at the test case level that solicits both the activity of preparing the dataset, with the “Dataset to test” label, and the data development activities, with the “Data platform to test” label, such as generation of synthetic data, data augmentation / balancing, labeling, or visualization (i.e., a data support activity).

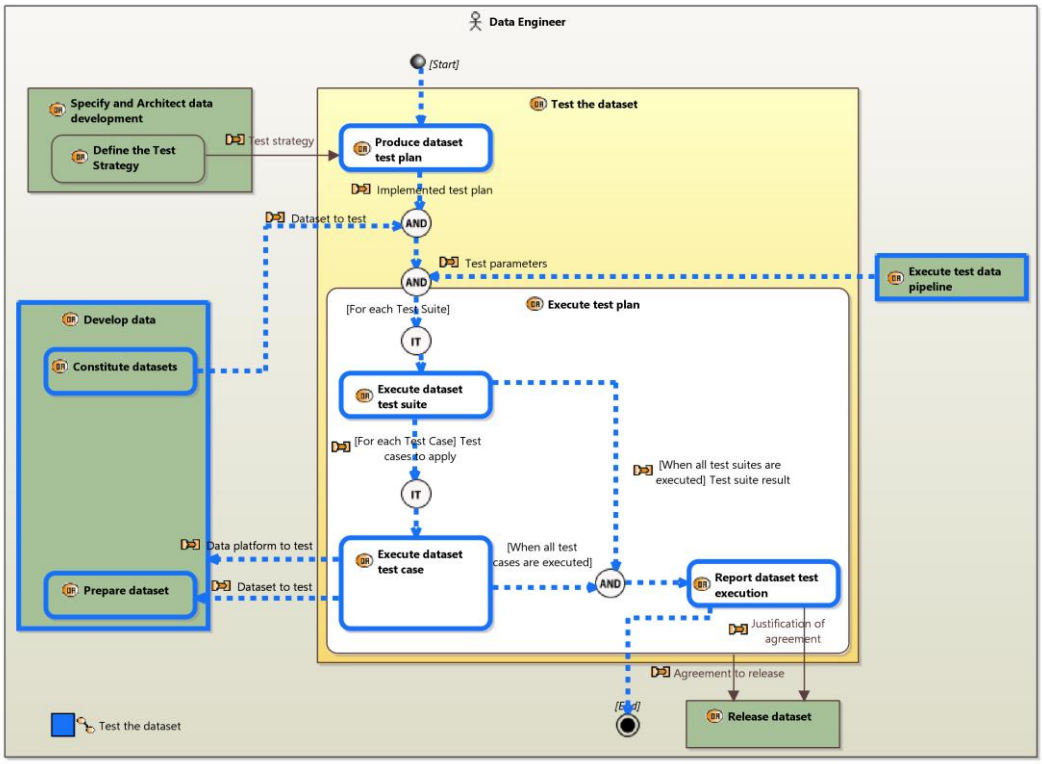


Figure 31: Test the dataset

Organization of the testing activities

For quality, a dataset is to be testing before delivery. In this purpose, some steps must be respected:

1. *Producing the test plan*. The test plan is to the test strategy what implementation is to specification.
    a. *Writing the test plan*. The test plan is written from the Test strategy defined during the data specification. It describes the details needed to implement and running the test. It is constituted of test suites, composed of test cases. Those test suites ensured that the developed dataset to be tested (available after the constitution of the dataset) is valid. A Test case can solicit functions (e.g., common operations on images), available in the context of preparation of the dataset, for realization of tests, and evaluate the level of data trustworthiness reached by the dataset.

   b. *Implementing the test plan*. The test suites/test cases can be implemented using a test framework, with assertions to valid.
2. *Executing the test plan*. When all the suites are executed, a report indicates whether the dataset is valid and is ready for release. For automation, a dedicated data pipeline executes all the test suites and creates the report.

Note that dataset, available after the Constitute datasets activity, is actually a set of training / validation / test datasets.

Three axis to lead testing

There are three main axes to lead testing:

1) the types of activities to test,
2) the level of data quality to reach,
3) the domain specificities.

1. Types of activities

Usually in software, tests verify that an output corresponds to an input according to a transformation function. However, what is the equivalence for data? What is a relevant test on data? For data, relevance is shown on the following table.

Type of activity	*Relevance*	*Reason*
Data acquisition	Maybe	The result of data acquisition is the initial set. If there is no or few transformation, then tests can be applied.
Generation of synthetic data	Yes	A complex activity may require use of auxiliary models, complex data, and sometimes data scalability or on the contrary a need of accuracy on frugal data.
Basic function of data preparation	Maybe	Dataset selection/conversion is useless. It could be in case of dataset balancing or augmentation, for instance to test a change of distribution on complex dataset such as time series.
Dataset annotation / Feature engineering	No	Only if deemed necessary.
Detection and solve of anomalies	Yes	Anomaly detection is useless on basic or hidden cases. However, it is necessary on complex structure. It is also useful on solve of anomalies (equivalent to a function of correction in traditional software).
Active learning	Yes	The proposition of classification must be asserted.
Domain adaptation	Yes	Tests are specific to the context of adaptation.

Incremental learning	Yes	Tests are specific to the context of adaptation.

2. Data quality

Generally, data testing is led to ensure validity of activities, but not the level of data quality to reach. For instance, in the initial dataset, all the expected items can be present with the expected features, but with a bad level of accuracy or availability or timeliness. The Orient data / Refine operational and system expectations section, identifies quality attributes. They are translated in KPI in the data specification. They are characterized and implemented by the evaluation framework. During testing, a test can ensure the level of reached accuracy with an error margin.

3. Domain

Testing depend on the application domain. For instance, tests for image, video, or text processing by batch differ from tests for supervision or surveillance. More, system criticality can require a more severe level of acceptance and stacks of explainability. The test framework / environment must be adapted to the domain context.

Testing and assurance case

Tests validate that a dataset meets expectations. A cross-matrix ensures that each expectation is checked by (at least) a test, and that each test checks all the related expectations. This traceability is a proof which can be exploited by assurance case.

Testing and operational scenarios

An operational scenario is an input to define a test suite. In case of complex scenarios, the same scenario can be broken down in several test suites.

H.1 Produce dataset test plan

Testing data guarantees that the implemented data meets the expectations:

1) During the development, to reach and maintain convergence towards the target, with regular regression tests,
2) Before the release, to ensure data quality, for instance with test campaigns.

During the specification phase, the test strategy outlines the objectives and approach of data testing, and answer to the questions: "What will be tested?" and "How will it be tested?" The test plan on its side

makes concrete test strategy. It identifies the testing scope, approach, tasks, roles and responsibilities, and test schedule. Here are the main differences:

Topic	*Test strategy*	*Test plan*
Scope	High-level document that outlines the overall data testing.	Detailed document that outlines the specific details of data testing.
Overview	Overview of the testing approach.	Detailed information about the data testing process.
Types of contained Information	Testing methodologies, test environment, test deliverables	Description of the tests (e.g., test suites, test cases). Schedule of the tests. Examples: architecture of the pipelines, objective of each one, orchestration. Environment. Identification of major technical elements to perform testing. Ex: deployed package, nodes.

The test suites, test cases are described. They are related to expectations (requirements, ODD, or operational scenarios) to meet. A traceability matrix identifies the level of tested expectations, and the expectations which are not implemented.

Test suites are implemented and executed by data pipelines to guarantee, by automation, a reliable reference of validation, and to raise as soon as possible issues in the dataset to release. A pipeline performs one or several test suites. Each one is scheduled at periodic times or triggered by events.

At this stage, this activity only consists in: 1) writing the test plan, including the test suites, test case, 2) implementing the test suites, test cases, 3) implementing the data pipelines.

Examples of regression tests to apply on data:

- Right acquisition of the initial dataset to operate tests
- Right structure of the dataset
- Respect of thresholds in requirements and ODD
- Respect of the expressed data quality attributes
- Success of the operational scenarios with test suites

The exchanges of this activity are the following:

- IN. Expectations
- IN. Data specification including the test strategy
- IN. Dataset information
- OUT. Test plan: definition and implementation

H.2 Execute test plan

During the development and essentially before a release, executing a test plan, as a set of test suites, ensures that data to release is correctly implemented and respect the expected level of data quality attributes. The test result reports:

- The result of each test unit and test case
- The synthesis of tests passed and not
- Scores on the data quality attributes
- The coverage matrix, with 1) for each test, the satisfied requirements / ODD, 2) for each requirements / ODD, the tests passed or not, 3) the percentage of coverage

Data pipelines automate execution of tests suites, except for those involving manual interventions.

The exchanges of this activity are the following:

- IN. Expectations
- IN. Data specification including the test strategy
- IN. Datasets
- IN. Test plan
- OUT. Test results

H.3 Execute test data pipeline

Test data pipelines automate the test to execute on data.

The exchanges of this activity are the following:

- IN. Pipeline parameters
- IN. Data, datasets
- OUT. Execution report

I. Phase: Release dataset

Objective:

- Making available datasets to consumer activities

Roles in charge:

- Data Engineer

Inputs:

- Data to release
- Agreement to release

Output:

- Released data

In an industrial context, the practice is to release a data(set) before using it by a model. The released is supposed to be tested with a justification (e.g., matrix coverage, description of the test environment), versioned, and available in a shared environment. Moreover, to be usable, the structure and the conditions of use must be described.

The consumer activities are generally:

- ML activities, for instance to train or validate a model
- IVVQ activities for validation at the system before deployment

J. Support data

Objective:

- Tools provide, during the development, or later during IVVQ and deployment, common data services.

Roles in charge:

- Data Engineer

Inputs:

- Data, datasets

Output:

- Elements provided by the tool (e.g., reports, logs for trace)

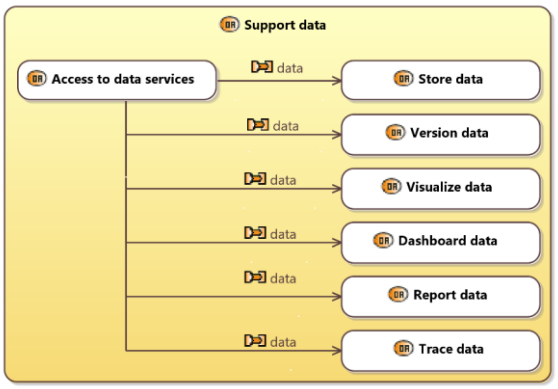


Figure 32: Support data phase

Data development and deployment need common services: versioning for configuration management of data (e.g., versioning of the initial dataset and of the successive versions of the dataset), visualization of the dataset (e.g., box-plot, histogram, Pareto chart, violin plot), dashboard to measure the progress of data construction, reporting (e.g., visual or on document), logs.

J.1 Store data

This activity encompasses all the services for storing data, that are:

- The infrastructure: cloud vs. on premise
- Storage by tools and associated format: SQL vs. NoSQL (e.g. OpenSearch), S3 (e.g., Minio), NFS
- The physical organization of data: path of files (e.g., images, files)

The exchanges of this activity are the following:

- IN. Data or datasets to store
- OUT. Stored data/datasets

J.2 Version data

Change management of data keeps track of versions of the dataset. Major steps need versioning, such as when the initial dataset or first dataset are available, for major reworks, or for each release. A baseline provides a reference point, either to start a new branch of work, for instance for improvements, or for learning exploration, or comparison over the time.

Depending on the size of the dataset, a version can become very space consuming, requiring a space management of versions.

The exchanges of this activity are the following:

- IN. Datasets
- OUT. Versioned datasets

J.3 Visualize data

Visualization is a presentation of data according to a viewpoint which defines rules, such as conditions of filter, aggregation, synthesis, convention of representation, methods of computation. Examples: graphical representation of the data, statistics, trends.

There are two levels of details:

- At the dataset level
- At the data element to explore the dataset

A visualization is relevant if it provides a pertinent representation to understand and analyze data, at a glance or in details.

The exchanges of this activity are the following:

- IN. Datasets
- OUT. Visual representations

J.4 Dashboard data

A dashboard does not visualize data. It is a representation to follow-up progress on data. Examples: progress towards reaching a threshold for a data quality attribute, curve of observed drifts on deployed data, global score of trustworthiness on data, health state of data pipelines. A data dashboard presents, according to a viewpoint, synthetic and analytical information about data in order to make decisions, in the form of actions, or to organize work.

The exchanges of this activity are the following:

- IN. Data, datasets
- OUT. Results on dedicated dashboard

J.5 Report data

A report, which aggregates and formats data, is generated:

- With a given purpose (e.g., to list anomalies)
- To a dedicated audience (e.g., for a development team member or manager)
- According to a format (e.g., a flat or sophisticated document)

The exchanges of this activity are the following:

- IN. Data, datasets
- OUT. Report

J.6 Trace data

Trace on data comes up at two levels:

- Edition: traceability enables to set relationships between elements, for instance between a dataset item and a requirement/ODD
- Reporting: log facts on data, and relationships with other elements, in a report

The exchanges of this activity are the following:

- IN. Datasets
- OUT. Traceability, reports

K. Combination of techniques

This section explores how to combine data techniques introduced in this guide and ML model techniques for efficient results or to provide new perspectives. Five data techniques were retained (i.e., active learning, incremental learning, domain adaptation / transfer learning, data augmentation, labeling) and only one on model (auxiliary model). This list is not closed.

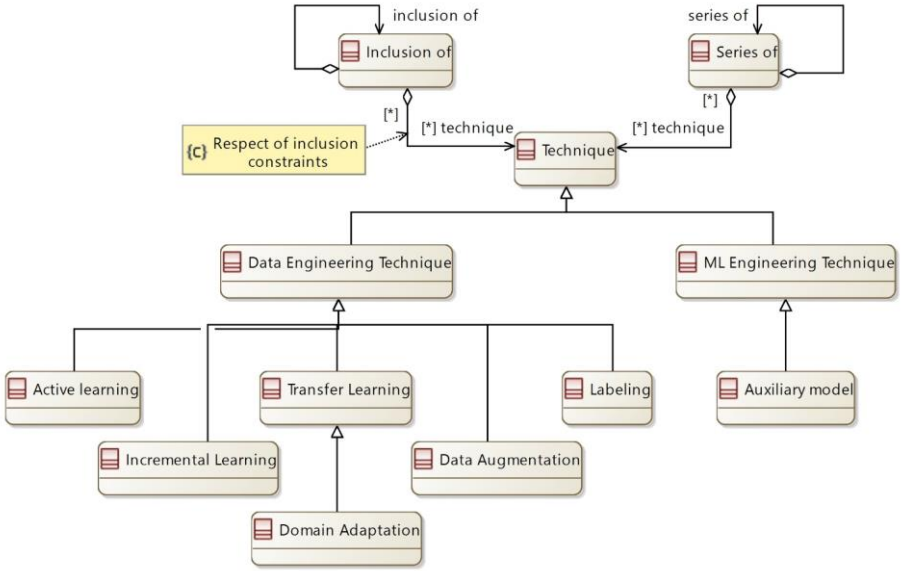


Figure 33: Relationships of techniques used by the data engineering

In the next pages, a Cartesian table combines those six techniques. Read this table like this: a one-row technique uses a one-column technique. The possibilities of combination are either 1) acceptable, expressed by Yes, 2) uncertain, expressed by Maybe, 3) impossible, expressed by No. This table is exploratory and with maturity, some answers may vary.

Uses ↱	ACTIVE LEARNING	AUXILIARY MODEL	DATA AUGMENTATION	DOM. ADAPTATION / TL	INCREMENTAL LEARNING	LABELING
ACTIVE LEARNING	No	Yes	Maybe	Maybe	Yes	Yes
AUXILIARY MODEL	No	No	No	No	No	No
DATA AUGMENTATION	No	Yes	No	Maybe	Maybe	No
DOMAIN ADAPTATION TRANSFER LEARNING	Yes	Yes	Yes	No	No	Yes
INCREMENTAL LEARNING	Maybe	Maybe	No	Yes	No	Yes
LABELING	Yes	Yes	Yes	Maybe	No	No

Technique 1	Technique 2	Possible	Description	Example / Application
Active learning	Active learning	No	Double-reference of activity.	
	Auxiliary model	Yes	An auxiliary model enables to detect candidate data items for active learning.	Use of model for detection of anomalies in time series, such as the Air Liquide use case, or SAM for image annotation.
	Data augmentation	Maybe	This would mean exploring and getting, by the technique of generation, all the data items for active learning.	Use case to find.
	Domain adaptation / TL	Yes	Active learning is performed by the use of domain adaptation/transfer learning to bridge domains.	For road signs, proposition of candidate data signs to migrate when adaptation from one country to another one.
	Incremental learning	Maybe	Active learning leans on incremental learning to identify and propose candidate dataset items for annotations.	Use case to find.

Technique 1	Technique 2	Possible	Description	Example / Application
	Labeling	Yes	Traditional practices of annotations for active learning. This is by definition the part of the activity of active learning.	
Auxiliary model	Active learning	No		
	Auxiliary model	No		
	Data augmentation	No		
	Domain adaptation / TL	No		
	Incremental learning	No		
	Labeling	No		
Data augmentation	Active learning	No		
	Auxiliary model	Yes	Use of a generative model to augment data.	Generation of synthetic data by diffusion model.
	Data augmentation	No	Double-reference of activity.	
	Domain adaptation / TL	Maybe	Data augmentation may use the technique of transfer learning / domain adaptation for a targeted augmentation.	Use case to find.
	Incremental learning	Maybe	Usage not identified yet.	
	Labeling	No	Data are already labeled.	
Dom. adaptation / Transfer learning	Active learning	Yes	Active learning eases to perform domain adaptation / transfer learning.	Identification of signs to annotate in a traffic lane in order to optimize domain adaptation from one country to another one.

Technique 1	Technique 2	Possible	Description	Example / Application
	Auxiliary model	Yes	An auxiliary model helps to domain adaptation / transfer learning in case of complex case and when reusing an auxiliary model.	Generation of synthetic data adapted to a target domain, such as look of objects for a culture, fashion, etc.
	Data augmentation	Yes	Generation of data to target a new domain.	Synthetic to real data generation.
	Domain adaptation / TL	No	Double-reference of activity.	
	Incremental learning	No	This would mean a continuous adaptive domain adaptation / transfer learning.	
	Labeling	Yes	Domain adaptation / transfer learning uses annotations to label data according to the target domain. It is a usual practice.	
Inc. learning	Active learning	Maybe	Active learning to optimize the selection of incremental data.	
	Auxiliary model	Maybe	Application of an auxiliary model during incremental learning for optimization.	
	Data augmentation	No		
	Domain adaptation / TL	Yes	Incremental learning aims to adapt the current model to the current data domain.	
	Incremental learning	No	Double-reference of activity.	
	Labeling	Yes	New data are annotated.	
Labeling	Active learning	Yes	First objective the active learning, easing the labeling.	
	Auxiliary model	Yes	Use of auxiliary model to label data.	Model for automatic pre-labeling.

Technique 1	Technique 2	Possible	Description	Example / Application
	Data augmentation	Yes	Generation of labeled data.	Labels which do not exist in the reality.
	Domain adaptation / TL	Maybe	Target domain without label but available in another one.	
	Incremental learning	No		
	Labeling	No	No double-labelling.	

L. Conclusion

This guide is a digestion of the data engineering practices in the context of machine learning and critical/complex systems. It is open to new activities as practices evolve.

After assimilation of the different activities introduced here, a real mastering consists in:

1) Managing efficiently the data lifecycle and optimally soliciting the convenient activities;
2) Interacting and composing complementarily with the machine learning to create data / ML model synergy.

The ways to improve this guide are:

1) Experimentation on operational projects for feedback and elicitation of practices and strategies of optimization;
2) Identifying the activities for assurance case;
3) For adaptability and in case of product / project families, the way to introduce reusability (i.e., equivalence of fine-tuning for ML models).

M. Bibliography

[Achepsah and Marwala, 2019] Achepsah Leke, C., Marwala, T. *Deep Learning and Missing Data in Engineering Systems*. Springer.

[Adjed et al., 2022] Adjed, F., Feuilleaubois, E., *Data evaluation metrics*. Confiance.ai program, Batch 2.

[ARCADIA] ARCADIA method. [Online]. Available: https://www.eclipse.org/capella/arcadia.html

[Bosca, 2023] Bosca, A., Leroy, B., Randon, Y., Winckler, N., Devèze, L. (2023). *Guidelines on synthetic data usage in machine learning context*. Confiance.ai program, Batch 2.

[Burkov, 2020] Burkov Andriy (2020). *Machine Learning Engineering*. Draft version.

[Capella] Eclipse Capella. [Online]. Available: https://www.eclipse.org/capella/

[Duboue, 2020] Duboue P. (2020). The Art of Feature Engineering. Essentials for Machine Learning. Cambridge University Press.

[Evans, 2003] Evans, E. (2003) Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison Wesley.

[Géron, 2019] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly. 2nd Edition.

[Hutchinson et al., 2021] Hutchinson B., Smart A., Hanna A., Denton E., Greer C., Kjartansson O., Barnes P., Mitchell M. (2021). *Towards Accountability for Machine Learning Datasets: Practices from Software Engineering and Infrastructure*.

[IBM, CRISP-DM] IBM SPSS Modeler CRISP-DM Guide. English version. Version 18.

[Lerbourg, 2023] Lerbourg, L. *[TBD – Document name]*, Batch 3.

[Leslie and Van Otten, 2020] Leslie, J., Van Otten, N. (2020). *Designing and Building Data Science Solutions*.

[Loesch et al., 2023], Loesch, A., Berjaoui, A., Troya-Galvis, A., Sanchez, E.H., Winckler, N., Vu, T.-H., (2023). *Domain Adaptation for object detection, label shift, person reID and continual semantic segmentation*, Batch 2.

[McCloskey and Cohen, 1989] McCloskey, M., Cohen, N. *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*. Vol. 24. pp. 109–165. doi:10.1016/S0079-7421(08)60536-8. ISBN 978-0-12-543324-2.

[Mattioli et al., 2021] Mattioli, J., Adedjouma, M., Delaborde, A., Jurie, F., Khalfaoui, S., Lecue, F., Leroy, B., Sohier, H. (2021). Confiance.ai, *Project EC2: Process and Methods, Trustworthy AI Algorithm Engineering Guideline*. Confiance.ai program, Batch 2.

[Robert et al., 2023] Robert, B, Awadid, A., Langlois, B., Le Roux, X., Proum, C-M. Confiance.ai, *Project EC2: Methodological guide for engineering trustable AI-based systems.* Confiance.ai program, Batch 2.

[Sammut and Webb, 2017] Sammut, Claude and Webb, Geoffrey, I. (2017). *Encyclopedia of Machine Learning and Data Mining*. Springer. 2nd edition.

[SAE, 2021] S. J. 325, “Taxonomy & definitions for operational design domain (odd) for driving automation systems,” 2021. [Online]. Available: https: //www.sae.org/standards/content/j3259/

[Sohier et al., 2023] Sohier, H., Mattioli, J., Amokrane-Ferka, K., Awadid, K., Botella, B., Delaborde, A., Gonzales, M., Khalfaoui, S. *Methodological Guideline for Trustworthy AI Assessment*. Confiance.ai, Batch 3.

[Weinstock, 2008] Weinstock, C.B. (2008). Assurance Cases. Software Engineering Institute.

[Zheng and Casari, 2018] Zheng A., Casari, A. (2018). *Feature Engineering for Machine Learning*. O’Reilly.

Confiance ai

Title : Formalization of the Data lifecycle for critical systems development

Keywords : Critical System, Data, Trustworthiness

Data is an essential part of systems integrating machine learning. Machine learning quality depends on data quality. However, today, developing data is often informal, without a clear process, while a data life cycle is not linear in a complex system and must be trustful for critical systems. This methodological guide describes how to develop data with trustworthiness for such systems.

Our partners

Air Liquide · AIRBUS · Atos · cea · Inria · NAVAL GROUP · GROUPE RENAULT · SAINT EXUPÉRY · SAFRAN · sopra steria · SystemX · THALES · Valeo · FRANCE 2030