Rev. 1.0.1.0037-e19 (2021/03/09)

Machine Learning Quality Management Guideline

1st English Edition

February 12, 2021
(Japanese: June 30, 2020)

Minor Update 1: March 9, 2021

Technical Report CPSEC-TR-2020002
Cyber Physical Security Research Center
National Institute of Advanced Industrial Science and Technology (AIST)

Technical Report
Artificial Intelligence Research Center
National Institute of Advanced Industrial Science and Technology (AIST)

Foreword and Disclaimer

This English version of the Machine Learning Quality Management Guideline is a translation of「機械学習品質マネジメントガイドライン 第 1 版」(Machine Learning Quality Management Guideline 1st edition), published by AIST on June 30, 2020 in Japanese.

This guideline is assembled by the "Committee for Machine Learning Quality Management" in the National Institute of Advanced Industrial Science and Technology (AIST).

This guideline is non-binding in relation to laws and regulations/official guidelines. Provisions described as normative in this guideline have normative meaning only if the guideline is adopted voluntarily. This document is distributed on an as is basis, without warranties of conditions of any kind, either express or implied.

This guideline is developed under support from the New Energy and Industrial Technology Development Organization (NEDO).

Table of Contents

1 Summary of the Guideline

The content of this Chapter is informative. Contents included in this chapter that constitute the norms of the Guideline are described again in following chapters.

The overall structure of this summary is as follows. The chapters and sections in the list below show the location of the corresponding content in the main part.

- Section 1.1 explains the background and purpose of the Guideline.
- Section 1.2 presents expected ways of using the Guideline (Chapter 2).
- Section 1.3 analyzes the reasons why quality management of AI is difficult.
- Section 1.4 mentions an overall concept of quality management process which the Guideline is based on.
- Section 1.5 presents three viewpoints of “external quality” (quality viewpoints that do not depend on implementation methods and can be evaluated only through use) set in the Guideline as goals (Chapter 3).
- Section 1.6 complements the previous section and explains why some of components generally discussed as “quality of AI” were not adopted in the previous section and views in the Guideline.
- Section 1.7 presents eight aspects of “internal quality” (quality aspect that depend on implementation method and can be managed by measurement or processes) set in the Guideline as “means of the quality management” (Chapters 6 and 7).
- Section 1.8 presents an overall image of development process on which the Guideline depends (Chapters 4 and 5).
- Section 1.9 clarifies the relationship between this guideline and various external normative documents (Chapter 8).
- Section 1.10 explains the structure of the remaining parts of the Guideline.

1.1 Purpose and Background

The effectiveness of artificial Intelligence (AI), especially machine learning technology, has been accepted in broad fields of applications such as manufacturing, automated driving, robots, health care, finance and retail business, and its social implementation seems to start to blossom. On the other hand, it is difficult to identify the cause when any accident occurs or to explain advantages of AI-based products to the amount of investments due to the lack of technologies to measure and demonstrate the quality of AI-based products or services. Consequently, wider acceptance of AI in the society is lagging, causing a big obstacle to the expansion of AI development business.

This document establishes a basis for quality goals for machine learning-based products/services, and provides procedural guidance for realizing quality through development process management and system evaluations.

This document aims to enable providers of products and services to evaluate and improve the quality of their systems so as to reduce accidents and/or losses caused by AI malfunctions in the society. Furthermore, it enables stakeholders to express their product quality using provided norms, which can be used for both commercial purposes (e.g. quoting prices of their AI-based products) and social purposes (e.g. to express their responsibility to the society).

1.2 Expected target of the Guideline

The primarily expected users of the Guideline is providers of products and services which are constructed using machine learning (hereinafter referred to as "service developers"[1]) and system developers that actually implement products and services as software. For each product, the service provider and the system developer may be either a single entity that develops products or services and provides them to end users (referred to as "self-development entity") or two separate entities depending on the sharing of responsibilities based on contract (sub-contracting or quasi-entrustment). Moreover, a service developer may sell implementation of machine learning components as a separate product. We primarily expect the Guideline to be used as a reference for these entities to share clear goals on required quality in accordance with situations in which products or services are used and to realize said quality throughout the system development process.

Furthermore, as a secondary usage, we expect the quality levels set by this Guideline can be referred to by end users for judging whether a particular system is safe to use. Also, the Guideline is expected to serve as a technical starting point for specifications and evaluation/certification the quality.

The Guideline is described in a generic way that it can be applied to as broad use cases of machine learning technology as possible. For each of specific application area, one can pick required parts of this documentation to make a specialized guideline. In addition, we expect each developer may choose to make their own specific version of development guideline to use, based on the document. The research project currently plans to publish some examples of such document as reference. The following list shows some possible cases of use of the Guideline as an example.

- When an organization for AI-based system development is established (contract, etc.)
  - An entity requested to develop a machine learning-based system or a machine learning component which constitutes that system (referred to as "development entruster"[2]) concretizes the Guideline in line with an application and designates

1 In cases where software developers implement systems in advance and sell these as packaged or customized products, such developers are treated as service providers in this guideline.

2 The "Guideline for Contract on Use of AI and Data" compiled by the Ministry of Economy, Trade and

them for an entity to which development is entrusted (referred to as "development entrustee") as specifications that serve as contract requirements.

- An entity that is to be a development entrustee refers to the Guideline as a standard for process management to guarantee the quality for a development entruster and use them as a ground or reference to calculate man-hours and the price for order.

– Design and development
  - An entity that designs and develops a machine-learning based system or a machine-learning component (development entruster or development entrustee) uses the Guideline as a standard for process design, system design and quality management.
– Society use
  - A self-development entity or development entruster bases its self-compatibility declaration on the Guideline with respect to quality requirement as a social norm when a system is provided to the society or used for its own business.
  - The Guideline serves as standards for social consensus on the quality of machine-learning based systems in the future.
  - The Guideline serves as standards for the design and operation of a third-party assessment system on the quality of machine-learning based system.

This Guideline is currently applicable for machine learning based systems that implemented with supervised learning. Although the basic concept can apply to other implementation methods such as unsupervised learning, semi-supervised learning and reinforcement learning, a specific way of treating them will be added in future revisions.

1.3 Challenges behind machine learning quality management

The implementation of artificial intelligence by machine learning is, if simply said, a sort of software. However, a conventional concept of software quality management is not technically feasible to improve and maintain the quality of machine-learning based systems. This section sorts out challenges related to differences between AIs and conventional software from several viewpoints.

Industry of Japan defines "development entruster" in this Guideline as "user", and "development entrustee" as "vendor" or "SIer" (system integrator), as it is written mainly for B2B (business to business) development agreements. This guideline, however, uses the term "user" only for end users of products and services, who are finally affected by the quality of the products.

Importance of environmental analysis

Machine-learning based products and services are often used under environmental conditions whose complexity is difficult to be understood by humans. Compared to regular programs for which humans have to analyze and identify complexity of all environmental conditions and implement every program codes as rules for judgment, there are high expectations on machine learning technology capable of automatically establishing rules for judgment based on data without having to identify detailed environmental conditions through analytical works by humans as an efficient means for implementing systems that operate under highly-complex environmental conditions.

On the other hand, from the viewpoint of ideal quality management, it is desirable to analyze environmental conditions as precisely as possible and understand risks at the early stage of system design especially from the viewpoint of quality management of systems whose compatibility to rare environmental conditions is called into question. If a machine-learning technology is used for these systems, analyses on environmental conditions used to be carried out at the time of implementing program codes in the past would be omitted. This is the reason why environment analyses at the early stage are important.

Therefore, it is important to consider how deep the status of use and environmental conditions are analyzed in order to balance between two characteristics (efficiency of implementation and environmental compatibility) at an early stage of system design including differences from the development of conventional software.

Lifetime-long requirement for risk assessment

Generally speaking, a system that operates in real world and constitutes so-called cyber physical systems, hereinafter referred to as “CPS”) has risks originating from external environmental changes in addition to risks that exist in regular software systems. Different from the implementation of regular software capable of theoretically analyzing and simulating unfamiliar situations to a certain degree and taking countermeasures, a machine-learning based system established based on real data may not be capable of dealing with significant changes in data trends originating from changes in our surroundings (although we have some anticipations on its generalization capability).

Moreover, a machine-learning based system may be designed in a way that it can attain practical quality only after additional learning using real data at the operation stage.

From this viewpoint, it is often necessary to introduce a continuous process lifecycle which integrates development and operation (DevOps) so that it can respond to situational changes after the system’s initial operation. We need to plan a lifecycle that includes operation-time updates from the early stage of the development planning and to reflect its requirements into both the operation plan and the contracts between stakeholders.

However, ensuring a certain level of quality at the starting time of operation is often

necessary, even if quality management will be carried out continuously. It is true particularly for systems with risks that compromise safety. From this viewpoint, the Guideline focus on quality tests from the implementation stage to the initial commencement of operation. Moreover, an effect of risks and a required quality level could be changed by varying a form of system operation at the early stage of operation and the full operation stage (for example, a possibility of avoidance by monitoring or intervention of the system by humans). In this case, it is required to synchronize the time when the form of operation is changed and the time when quality management goals are changed.

Furthermore, if the system implementation is updated at the time of operation, there is a risk in some cases that the quality deteriorates (called regression in software engineering) even if that update aims to improve the quality. It is supposed, therefore, that we need a certain form of quality monitoring and countermeasures at the time of operation, but they vary depending on forms of development and operation. Section 4.3 sorts out some possibility for such operation models.

Quality assurance depending on data

A request for “quality of data” used for building machine learning is often mentioned in data-oriented development. The Guideline assumes that the quality of data itself is not the ultimate goal of quality management but rather the cause of quality deterioration or means for ensuring quality. Of course, it is almost indispensable to ensure a certain level of data quality to actually ensure the machine learning quality through the lifecycle process management such as the Guideline. In addition, when a machine-learning based system is developed by sharing works, learning data may be sold/bought or shared for development. In such a case, there is a room for discussion by treating “data quality” as an independent character. Moreover, there has been pointed out at recent academic meetings that there are security risks such as contamination of learning results due to intentional injection of improper data. Effects of such improper data cannot be detected by simple numerical evaluations but require checks on data sources from broader perspectives.

The Guideline is supportive of ensuring the quality by means of numerical evaluations to the extent as possible, but (at least now) we need to secure the quality through qualitative data quality management as well.

1.4 Basic concept of quality management

(note) The Guideline defines “quality in use” as quality to be provided to end users in the whole system. On the other hand, “external quality” and “internal quality” are examined at component level[3] which will be combined into a single system.

“External quality” of each component is quality required as a part of system from an objective perspective. It includes, for example, security, reliability, and consistency. This external quality includes both qualitative and quantitative qualities and cannot always be expressed in single measurable indicator.

On the other hand, “internal quality” of each component refers to quality as a unique characteristic which is specifically measured or evaluated through acts of development including design when the component is created.

Based on this concept, Figure 1 explains hierarchical quality model in which external quality of each component is realized through its improved internal quality and is required for achieving internal quality of the element in one layer outside. “External quality” of the whole system is realized and provided by a provider of products/services for the sake of “quality in use” viewed from their end users.

[3] The term “internal quality” roughly coincides with the “internal measure of the quality” in ISO 25000 series. The concept behind the “external quality” is not directly associated with “external measure of the quality”, because the guideline assumes that external quality is not generally satisfactorily measurable as a numeric quality indicator. Instead, this document ensures satisfaction of an external quality “level” through assessments of internal qualities.

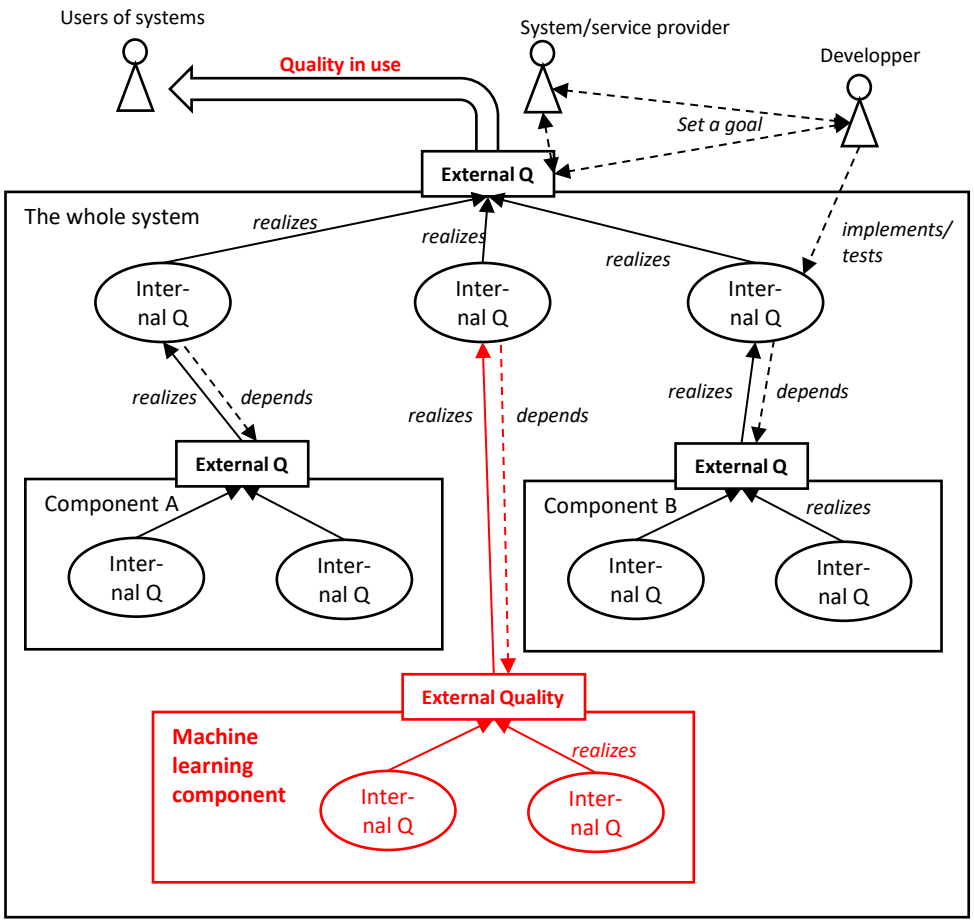

Figure 1: Hierarchical quality model

In the Guideline, “quality” of a machine-learning based system itself is understood by 1) “quality in use” expected to be satisfied in the whole system in use, 2) “external quality” expected to be satisfied in components of the system built by means of machine learning, and 3) “internal quality” which constituent components built by means of machine learning uniquely have. “Quality” is understood as what satisfies a required level of “external quality” through improving “internal quality” of machine-learning components and realizes “quality in use” of a final system (Figure 2).

Three viewpoints listed in Section 1.5 were set as external qualities of machine-learning components (target of quality management). We focused viewpoints unique to machine-learning components as internal qualities to achieve said external qualities and extracted eight viewpoints listed in Section 1.7 at this stage. The quality goals were categorized by level in relation to the three viewpoints for external qualities, and the performance goals were set for each of the eight viewpoints for internal qualities in accordance with said categorized levels in order to achieve those goals through various technologies and development process management. This is a basic concept of quality management in the Guideline.

Since qualities in use of the whole system to be realized ultimately have different focuses

depending on its application, it is not specified specifically in the Guideline (See the following supplement).

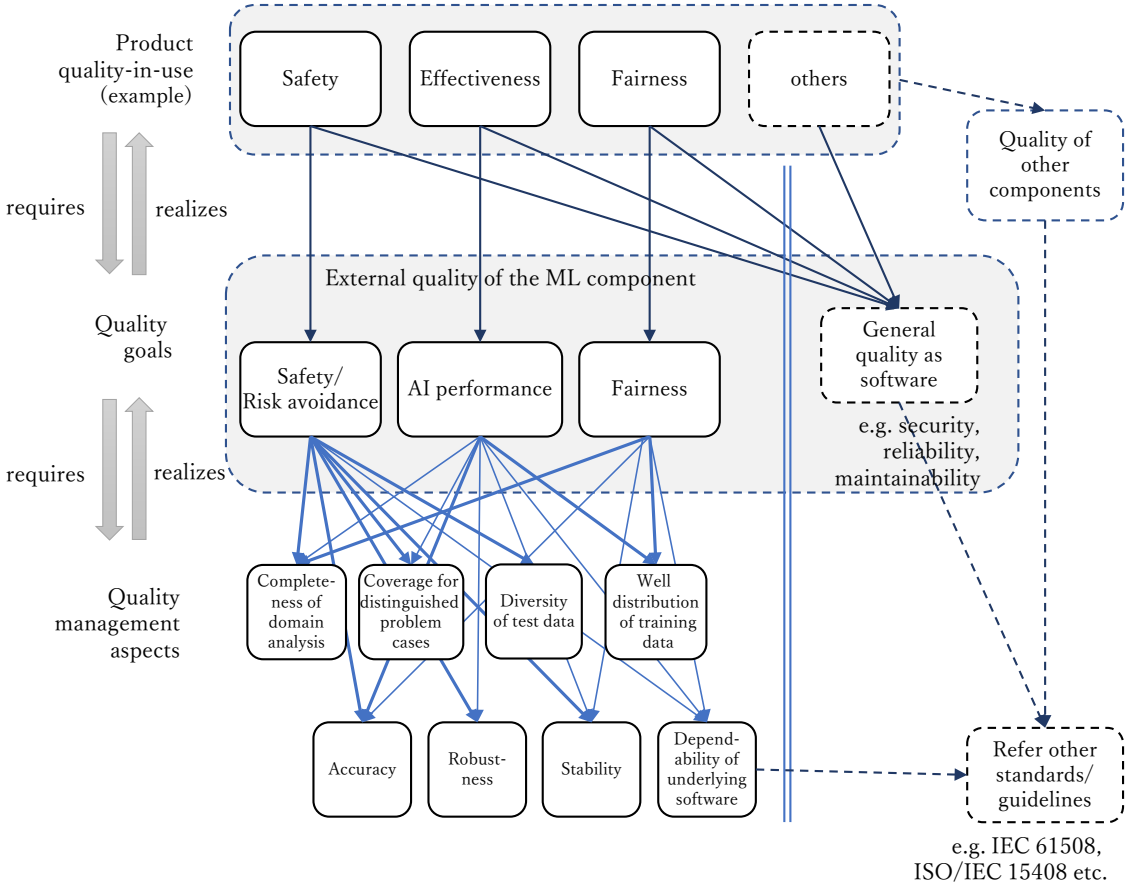

Figure 2: Overall structure of realization of product quality

Example 1) A module to recognize objects based on front images installed on a self-driving car is envisioned. One of qualities in use of automobiles is “safety for avoiding collisions with obstacles under all environmental conditions under which automobiles can be driven”. In order to realize this, the object recognition module must have a characteristic that “it can correctly recognize obstacles in all possible climate conditions and time zones” as an external quality. Internal qualities required to realize this include completeness of learning conditions. This is realized by the process of configuring machine learning training data.

Example 2) AI to estimate stock prices included in automated stock trading services is envisioned. One of qualities in use of the whole service is “maximization of profits”. A module to estimate stock prices must have a characteristic that “minimizes errors in stock price predictions and maximizes the total of envisioned trading results as an external quality. Internal qualities required to realize this include, for example, the precision of reasoning of machine learning This is realized by optimizing machine learning training parameters.

(Supplement) When a product or service is developed using strict processes to evaluate safety and reliability of the whole industrial systems such as IEC 61508[4] or IEC 62278[5], a requirement for external qualities is of course identified in relation to machine learning components in the design process of whole conventional systems.

On the other hand, if products or services are used for IT services that have strong request for machine learning technologies, an expected level of risk is not as high as that of industrial systems in many cases. Therefore, it may not be necessary to apply a strict risk management process to the whole system development.

Moreover, if we expect machine-learning components to "make a wise decision" in any way as an overall use of AI technology, the three viewpoints listed in this Section may often require somewhat direct response to external qualities and qualities in use of the whole system. The above two cases illustrate specific cases.

Based on such observations, Figure 2 is described in a way that qualities in use of the whole system and external qualities of machine-learning components are paralleled. Moreover, three qualities in use (safety, effectiveness, and fairness) are exemplified.

1.5 External quality characteristics to be achieved

In conventional systems, quality requirements on most software components are "correctness in terms of given design specifications", whatever the quality-in-use required to the whole system is. It is because most of the quality-in-use requirements are considered at the system design stage, not at the implementation stage.

On the contrary, in many cases, implementors of machine learning components are required to consider some quality aspects closely related to the quality-in-use.

This document identifies the following three properties as the quality goals specific to machine learning components.

Safety / Risk Avoidance

The safety, on the machine learning component level, is a property to reduce possibilities of generating undesirable, probably harmful outputs from a machine learning component. The following are some specific examples related to the safety property:

– Oversight of obstacles in object recognition for automated driving and false

4 IEC 61508-1:2010: Functional safety of electrical/electronic/programmable electronic safety-related systems - Parts 1

5 IEC 62278:2002: Railway applications - Specification and demonstration of reliability, availability, maintainability and safety (RAMS).

recognition of type thereof;
- Oversight of foreign objects in the detection of contaminants in a food factory's production line; and
- An order exceeding the acceptable level in automated trading of securities due to illegal input (spoofing)

We have set seven levels of safety, varying from the one that affects many human lives and continuity of businesses to the one that only causes minor losses of opportunities for profit (Section 3.1). Since the characteristics of safety are closely related to the properties handled in safety specifications for conventional systems, we have set four levels (Levels 4–1) corresponding to the existing specifications (e.g. SIL of IEC 61508) in response to strong industrial requests taking the combination and affinity with those specifications into consideration. On the other hand, typically for the application of machine learning to IT services and smart devices, we added three levels from the practical point of view, because only minor damages that are not considered as risks in conventional industrial products are envisioned in many cases so that these requests for quality are categorized into the same level (Not Applicable) in the existing safety specifications.

Generally, in machine learning systems, it is not practical to strictly guarantee a characteristic that "they always operate safety at all times" and machine-learning components themselves may not achieve qualities in use required for the whole systems. When a system is actually built, its realization depends in many cases on "safety assurance valves" by implementing peripheral software not on machine-learning components. The Guideline recognize requests for qualities in use of the whole systems and requests for external qualities of machine-learning components separately in the same way as conventional specifications for safety and set levels required for external qualities of machine-learning components based on risk assessments on the whole system and its configuration (Section 5.1.1.5).

AI performance (usefulness)

As the second axis of characteristics, we focus on applications to fields in which the usefulness of machine learning functions is emphasized and categorize it as the axis of characteristics of "AI performance". A typical application in which priority is given to AI performance rather than to risk avoidance includes a scenario in which higher average performance is required than negative effects caused by individual misjudgments such as request forecasting by retail stores and forecasts of investment decisions.

In some cases, both "AI performance" and "safety" are required. For example, although the top priority of automated driving is risk avoidance to prevent accidents, improved ride comfort and the optimization of arrival time at a destination can simultaneously be the second and third goals. As another example, on a price forecasting application, an excessive purchase could cause more indirect negative effects on management environment than expected,

resulting in big economic losses that cannot be assessed based only on the average value of profits. In this case, it is necessary to clarify an optimal middle ground of two axes of characteristic, "AI performance" and "risk avoidance".

Since specific targets to be achieved of AI performance (Key Performance Indicator (KPI)) differ from one application to another, three levels were set from the viewpoint of how much the achievement of those goals is required.

Fairness

A kind of social norms or ethics is often required for AI. From the engineering (not humanities) viewpoint, many of these social norms boil down to various existing characteristics of qualities in use and the process of their setting. Under these circumstances, we can extract "fairness" as a quality viewpoint specific to AI, which has not been taken into consideration in the past, by using a recent statistical technique.

Although conventional software has to make a "fair judgment" in many cases, its realization completes mainly when examinations are made in the design stage. Therefore, other quality characteristics such as "reliability" (software operates correctly as designed) were emphasized in the implementation stage. However, in machine learning, probabilistic and statistical behaviors are included in the implementation process and learning results so that a prior examination is not enough to ensure fairness, and machine-learning components (software elements) have to directly realize "fairness" as a quality.

From this perspective, the Guideline pick up "fairness" as the third external quality characteristic. "Fairness" is defined in the Guideline as a request for certain statistical characteristics regarding the distribution of overall output such as the non-existence of undesirable biases from the users' point of view in output from machine-learning based systems. As an example, we envision cases where it is required to explicitly or implicitly prove that there is no bias due to differences between races or sex in health insurance quotes and scorings for recruitment.

Four elements (FAST (fairness, accountability, sustainability, transparency)) are generally pointed out as social requests for machine-learning based systems. The Guideline focus firstly on fairness which can be directly analyzed as a statistical characteristic.

There is much room for discussion in how to set and realize goals for fairness. Currently, we provisionally categorized fairness into three levels depending on the level of social request.

The fact that certain levels of risk avoidance and AI performance are required for all attributes. is not considered as fairness here but sorted out as risk avoidance and AI performance. For example, a request for quality concerning the effect of differences in sex or physical constitutions in protective devices for safety is sorted out as a "request for risk avoidance so that they are safe for all sexes and physical constitutions" not as a "request for fairness without any differences in safety due to sex". This is because a technical way of realizing fairness can be sufficiently balanced out with other performance indicators and it is

not desirable to solve the problem by improving fairness at the cost of safety.

Furthermore, as regards this external quality characteristic, we examined “which aspects of fairness are measurable and can be subject to quality management” and “how can those aspects be realized in machine-learning based components” in the Guideline. “What kind of fairness realizes social justice in a specific system” should be examined in advance outside the scope of the Guideline. These social norms and ethics will be discussed further in Section 1.6.3.

1.6 Possible other aspects for AI quality

The Guideline set forth that the quality is managed with a focus on three viewpoints (risk avoidance, AI performance and fairness) as described above. Generally, various viewpoints are under discussion concerning the “AI quality”. This section sorts out the relation of some viewpoints other than the three which we have already sorted out with the Guideline.

Security/privacy

AI security including machine-learning based systems has two viewpoints; “security that affects external environment” and “security as information processing system”.

The former includes malfunction due to intentional input equivalent to adversarial example. This point will be examined in detail in the following section as “attack resistance”.

As for the latter, consistency and availability among three so-called security components (secrecy, consistency and availability) can be discussed as pure software security. The inference calculation using a trained machine-learning model is a simple numerical calculation in which the time calculation does not hardly change based on input values, and it is considered enough if reliability as conventional software is assured. These elements should be managed by means of conventional specifications as necessary, for example, by setting the evaluation assurance level (EAL) set forth in ISO/IEC 15408. Therefore, they are not examined in the Guideline.

As for “secrecy”, it is pointed out in academic research results that a part of training data can be estimated based on the content of trained machine learning models under certain conditions, and this could pose a future challenge from the viewpoint of protection of personal information and privacy. However, at this stage, we will keep an eye on the progress of academic research and, in the Guideline, we examine “secrecy” within the conventional framework of the protection of privacy by reference to the handling of anonymized data under the Act on the Protection of Personal Information as necessary.

Resilience to attacks (security)

In recent years, it has been pointed out repeatedly that the current implementation of AI in machine learning is vulnerable to intentional alterations of input data. Moreover, it is pointed out that such vulnerability may be exploited through altering external situations of

the system. On the other hand, risks associated with actual operations have not been evaluated clearly. Furthermore, countermeasures against such attacks are pointed out to have negative impacts on performance of machine learning components in regular operation.

At this point, we consider that it is more practical to accurately evaluate a possibility that an attacker intentionally alters a surrounding environment when the system is in operation and its consequence for each of the axes of the quality characteristics listed in the previous section than evaluating attack resistance as an independent quality requirement. The necessity of countermeasures against intentional alterations is examined in the quality management process in according with whether such alterations in actual operation may be excluded. If it is necessary to take such countermeasures, their effects are basically evaluated through "Stability of Machine Learning Model" described in Section 1.7.6.

Social aspects such as ethicalness

As qualities required for machine-learning based systems, ethicalness and social validity of results and judgments made by machine-learning components may be included. For example, in fields closely related to personal information (prior scorings for hiring, crime forecast, etc.), it may be required to guarantee that differences in sex or races do not affect judgments legally or socially. Moreover, in fields that are likely to cause human and economic damages, there may be a case where the validity of possible judgments with different risks is called into question (a judgment made by a self-driving system under a situation where some human damage is inevitable).

The Guideline do not directly examine what kind of judgment machine-learning components should make to have social validity in those cases, since it should be sorted out by humans in advance as a part of required definitions at the beginning of system development. Then, a machine-learning based system draws out processing results with a high probability to the extent possible toward "correct" output sorted out by humans, and this is considered as a request for qualities in use.

Section 1.5.3 lists "fairness" which has not been treated as direct external qualities especially in conventional software engineering, as one of external quality axes. The international efforts for ethical aspects such as fairness and transparency are summarized in Section 8.2 as reference.

Limit of response to complex external environments

As a general problem of AI, discussions on the limit of complexity of environments where AI can be utilized draw attention. Machine-learning based systems are often incorporated into a part of so-called cyber-physical systems in open environments such as streets and public spaces. Therefore, it is impossible for AI to make expected judgments under all environmental conditions, if ultimate conditions are included. This issue is inherently common not only to

machine learning but also to devices and software that operate in open environments. Conventional specifications for reliability engineering such as IEC 62998[6] seem to intend to tackle this issue somehow.

The Guideline follow the overall concept of past specifications. The adequacy of risk analysis and system design required for realizing the quality in complex external environments becomes subject to examination of quality management methods with regard only to differences in characteristics from conventional software and points to remember for unique analysis and design originating from those differences.

1.7 Internal quality characteristics subject to quality management

In the Guideline, the following eight characteristics were extracted as characteristic axes of quality management in response to the achievement of two external qualities (“risk avoidance” and “AI performance”) as specific characteristics managed by internal qualities of machine-learning components. Section 9.1 describes the outlie of analyses leading to this extraction.

“Fairness” will further be analyzed to add necessary characteristic axes of quality management. Please refer also to Section 9.3.

[6] IEC TS 62998-1:2019: Safety of machinery - Safety-related sensors used for the protection of persons.

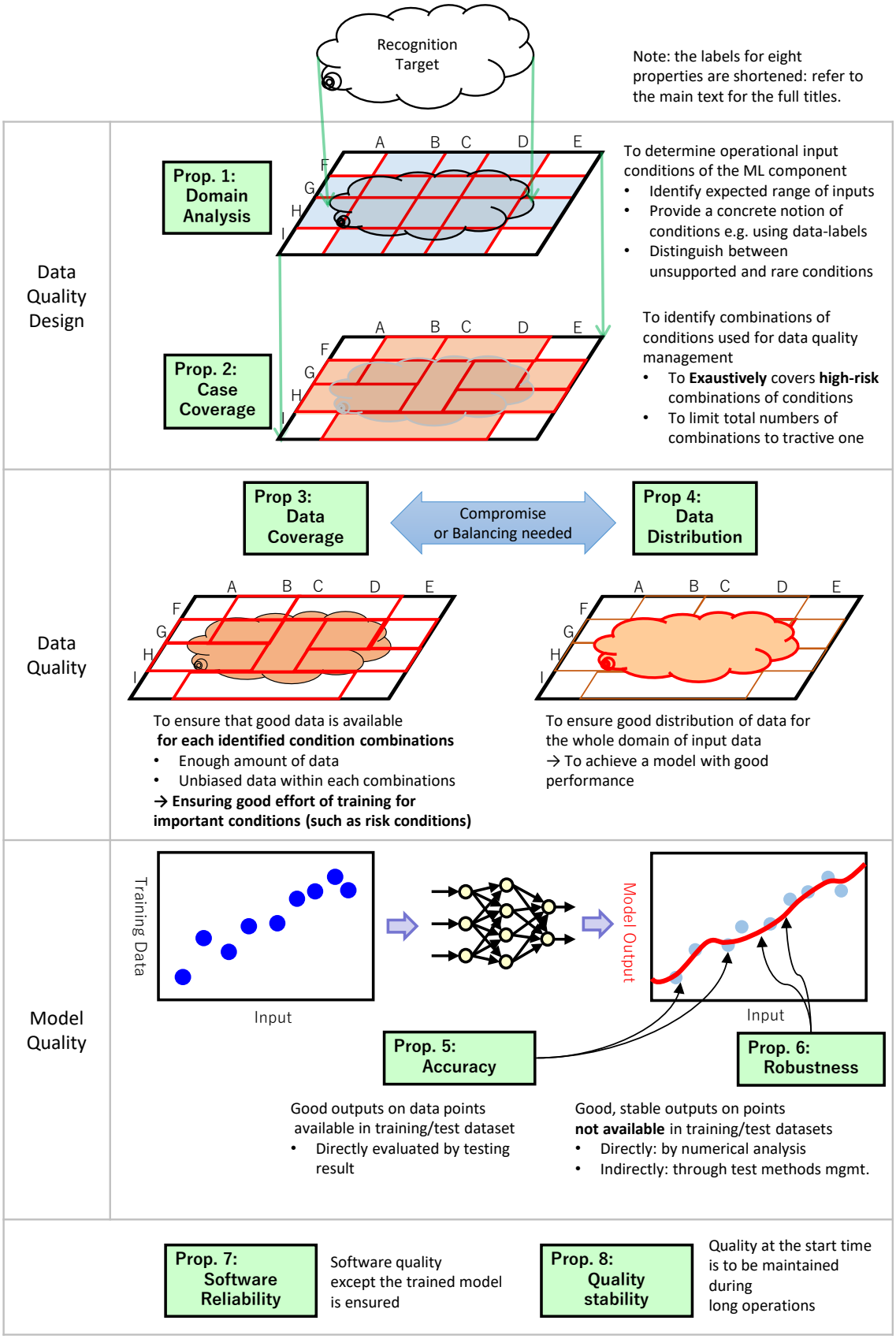

Figure 3: Internal quality characteristics focused

Sufficiency of requirement analysis

First, we have to analyze the characteristics of real data in operation which are envisioned to be input to machine-learning components corresponding to the usage of machine-learning based systems in real world. We call a case where the analysis results cover all possible uses as “sufficiency of requirement analysis”.

The fundamental requirement analysis is usually thought to have completed as a part of early-stage design. At this stage, we aim to write down the analyzed requirements with specific words to a level tagged to individual data as well as to set an “axis” for analyzing the quality based on data, so that they can be used to sort out training data and check if there is necessary test data in later stages.

There are different ways of setting a specific axis for data analysis. The basic concept of the Guideline envisions a concept of feature tree in conventional software product line engineering and a more simplified method of classifying and organizing “the axis” as itemized independent conditions and capturing specific uses as their combination (See the following examples). Moreover, we conduct at this stage both top-down analysis from the request side such as risk analysis/failure mode analysis and bottom-up analysis consisting of preliminary data analysis in the PoC (Proof of Concept) stage to sufficiently sort out possible situations in which external qualities change (deteriorate).

(Example 1)

We can write down the following situations which a self-driving vehicle running outdoor might encounter in machine-learning components that recognize traffic lights based on images.

(Please be reminded that this example is not sufficiently comprehensive for actual application)

Significance of display: Green, yellow, red

Time zones: Day, night

Weather conditions: Sunny, cloudy, light rain, heavy rain, snow, fog, etc.

Others:

For example, the situation in which “the traffic light is red on a rainy night” is one “combination of situations”.

(Example 2)

We can write down how a machine-learning component that predicts the sales trend of a retail store is used as follows (this example is not comprehensive, either).

Day: Monday, weekday, Friday, Saturday, Holidays

Weather conditions: Sunny, cloudy, light rain, heavy rain, snow
Time zones: Morning, before noon, noon, afternoon, evening, night, midnight
Season: Spring, summer, fall, winter
Neighboring events: Yes/No

In this example, one combination could be “there is a neighboring event on a holiday morning under winter clear sky”.

The sufficiency of this requirement analysis deals with analysis of risk factors in conventional software and test designs to include those risk factors when a black-box test is conducted. It is also an important characteristic that establishes a unit to grasp and check the quality. On the other hand, when machine learning is implemented, minor characteristics above a certain level may be left to learning processing in the training stage, and a person who implements the system may not give detailed instructions on how to judge individual conditions. We can say that these are the biggest benefits. Moreover, in some cases, the implementation by machine learning shows better performance than that by humans. From this viewpoint, the configuration of “details to be included in requirement analysis” becomes a very important point in examining an implementation strategy of machine-learning components taking the quality into consideration.

Furthermore, when this type of analysis is made, we may recognize a situation which we do not encounter in real life and an extremely rare situation in which it is not practical (it is not required to respond) to guarantee operation (e.g. snowfall in summer) or a rare situation that does not appear in training data to be collected but it is required to respond to (e.g. snowfall in spring in the Tokyo area). Specifically, it is very important to distinguish those two situations in any system for which risk avoidance is required and the distinction is directly connected to design of safety/robustness of the whole system. At this stage of considering the “sufficiency of problem domain analysis”, it is also important to identify situations that cannot happen in real life and eliminate them from examinations in later stages, in the process of identifying rare situations (cases) difficult to be found based only on data collected from real life that require response and designing corresponding systems.

A specific concept of configuring these situations will be explained in Section 6.1 in more detail.

Coverage for distinguished problem cases

On the premise of the sufficiency of problem domain analysis, sufficient examinations of data design to collect and sort out sufficient training data and test data are required as “Coverage for distinguished problem cases” in response to various situations which systems have to respond to.

In an extremely simple system, it is enough to mention that corresponding data to all

"combinations of situations" identified in the problem domain analysis described in the previous section included in training datasets and test datasets. However, if a situation in which a system is envisioned to be used is complex, the number of possible combinations becomes enormous. Therefore, it is not practical to cover all combinations with datasets (for example, only in the simple case mentioned in the above Example 2, the total number of combinations is 700. In a real case, this number is envisioned to reach 10,000). In this case, it is required to check completeness with a rough granularity level that covers several situations to sufficiently deal with combinations of detailed situations in which any danger or reduction in performance is likely to occur. There is a concept of "standard for completeness" applied to design of tests in the field of software engineering which aims to achieve practical and sufficient operational completeness by selecting appropriate means for each application.

Coverage of test datasets

We call a property where a sufficient amount of data (especially, test data) is given to each "combination of situations that require response" designed in the previous paragraph without any missing situation as "coverage of datasets".

When regular software is developed, the details of all features in real world which software operation depends on is grasped in any point from problem domain analysis to implementation and reflected ultimately as conditional branching or calculation formula in programs. However, when machine-learning components are built, more minute situations than a certain degree are not grasped explicitly as "feature quantity" or "ground true labels" of training datasets and are only included implicitly in training datasets. They will be reflected in final operations throughout the training stage of machine learning. The purpose of configurating this axis of characteristic is to guarantee that the shortage of learning due to the shortage of data or any oversight of learning in specific conditions due to biased data does not occur in any situation or case identified in requirement analysis or data design.

(Example 1)

In the case of recognition of images of traffic lights mentioned earlier, this characteristic means that all forms of traffic lights in each prefecture, their height and distance between them, and road conditions around them are included as data without any bias and that training is not carried out based only on limited data of a specific city.

(Example 2)

When image recognition AI that aims to recognize "cats" from small animals around town is to be built and individual characteristics of cats such as breed and size are excluded from recognition, this characteristic means that sufficiently diverse images of "cats" and "other small

animals such as dogs" are made available as data and that training does not only target specific breeds (e.g. calico cats) in environments where the product is expected to be used.

Uniformity of training datasets

A concept contrary to "coverage" mentioned earlier is "uniformity" of data in relation to the overall assumed input data. When each situation or case in datasets is extracted in accordance with the frequency of its occurrence in whole data to be input, data is considered as "uniform" (Figure 4). The balance between this uniformity and coverage is evaluated. The prediction accuracy of machine-learning technologies is generally supposed to improve by using samples extracted uniformly in relation to input environments as training datasets. However, the coverage of situations explained in the previous paragraph may be emphasized depending on actual application and quality characteristics required therefor. It is necessary to consider to which we give priority, coverage or overall uniformity, and how to strike a balance between them.

When risk avoidance is strongly requested, sufficient training data is required for high-risk situations that have to be avoided by making correct judgments. If such a rare case occurs and you intend to train data by ensuring a sufficient amount of data in relation to that rare case and maintaining the uniformity to all other situations, the necessary amount of data might be enormous. In this case, priority may be given to training of rare, high-risk situations.

On the other hand, when overall performance (AI performance) is requested, certainty of inference results in other cases might deteriorate by giving more priority to the training of rare cases than the actual probability of occurrence, thereby resulting in the deteriorated average performance as a whole. In this case, standards for coverage for each situation mentioned in the previous section are not appropriate.

Moreover, when fairness is strongly requested, "what kind of fairness" is requested may change a decision. For example, you may choose to give the cases artificially-equal training by artificially processing data (selection or addition of data) or let the cases learn at random in line with the distribution of extracted training data.

Since the two viewpoints mentioned in Section 1.7.3 and this Section may be compatible or incompatible, you may need to adjust appropriate training data to strike a balance at an adequate level. Moreover, different characteristics may be sought in the training stage and the quality test stage.

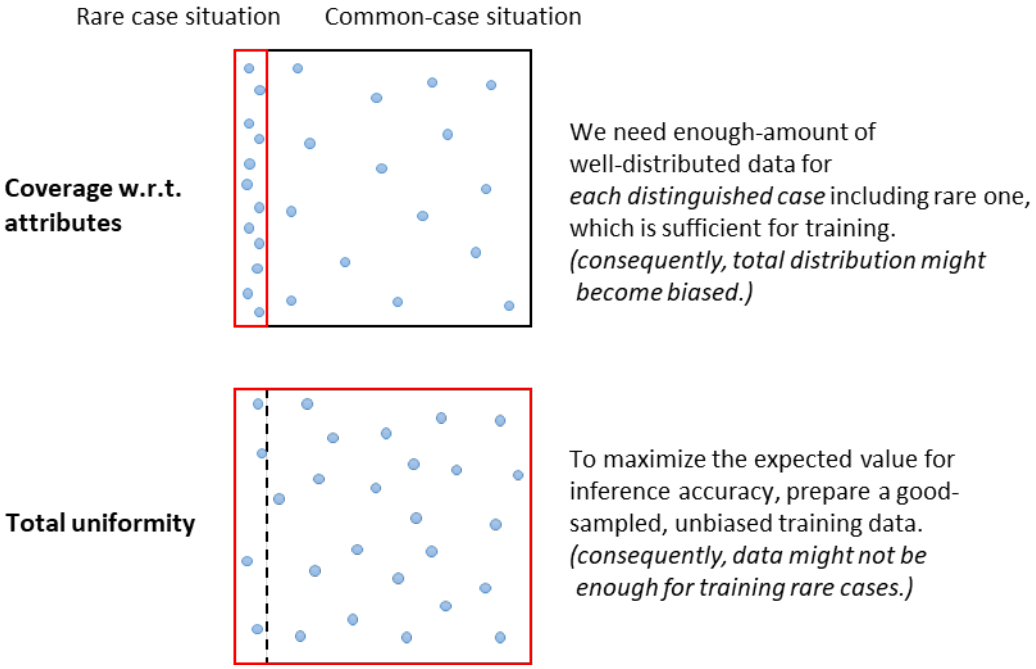

Figure 4: Relationship between coverage and uniformity

(Example 1)

When the recognition of traffic light images operates for full self-driving cars throughout the year in Tokyo, it may be necessary to include sufficient learning images of snowy days (assumed to be 1 or 2 days per year in Tokyo) in which there is little visibility. On the contrary, if it is excessive to prepare images of other climate conditions 360 times as many or snowy cases are assumed to be learnt insufficiently, we may increase a ratio of images with snow higher than the actual probability of appearing snowy days.

In this case, we might give more priority to “coverage of datasets” than to “uniformity of datasets”.

(Example 2)

On the other hand, if 1 or 2 snowy days a year are strongly trained in sales predictions of a retail store in Tokyo, the prediction performance in other climate conditions may deteriorate and the average profit cannot be maximized. In this case, we might need to give priority to “uniformity of datasets”. Of course, what kind of data is actually prepared as training datasets will finally be examined in the implementation process taking the balance of both internal quality characteristics into consideration.

Correctness of the trained model

The term “correctness of a trained model” represents that a machine learning component functions as intended upon the input from the learning dataset (consisting of training data,

validation data, and test data). In this guideline, this notion also includes the convergence of the training and the quality of training data (e.g., the dataset has only a small number of outliers and incorrectly labeled data).

Normally, the training dataset does not provide sufficient information to reflect all input given from the in-operation environments. Therefore, a trained model might show high performance with respect only to training data, but not to the data other than the training data. In other words, a trained model may overfit to the training data and may not generalize well to unseen data. When a machine learning component is evaluated using only the validation datasets and test datasets, it is not possible to check if the machine learning component functions as intended when given unseen data from the environment. Hence it is important to evaluate not only the correctness but also the stability, which will be explained in the next section.

Stability of the trained model

The term “stability of a trained model” represents that given an input that is not included in the learning dataset, a machine learning component behaves in a similar way to when given a nearest training data as input. When a trained model does not satisfy stability, it may not function correctly upon the input data that are not included in the training, validation, or test datasets. For example, a trained model may function incorrectly when the input is perturbed by natural/adversarial noise.

Reliability of underlying software system

The term “reliability of underlying software system” represents that the underlying conventional software (e.g., training programs and prediction/inference programs) functions correctly and reliably. This notion includes various software quality requirements such as the correctness of algorithms, the time/memory resource constraints, and the software security.

Maintainability of quality during operation

The term “maintainability of quality in operation” means that internal qualities fulfilled at the time when the operation started is maintained throughout the operation period. Therefore, internal qualities 1) can sufficiently respond to changes in external environments and 2) prevent the quality from deteriorating due to changes in trained machine learning models made for such response.

A specific method to realize quality maintenance depends on forms of operation, especially on how to carry out additional learning/iterative training. This point will be described in Section 6.8.1.

1.8 Concept of development process model

Relationship between iterative training and quality management lifecycle

Machine-learning based systems broadly apply more adaptative and flexible development processes such as the development of conventional waterfall process, the introduction of preliminary experiment stage called Proof of Concept (PoC) and agile development. Moreover, continuous development that realizes higher quality during operation and responds to environmental changes based on real data obtained in the operation stage is thought to be effective in machine learning.

Hence, the Guideline define not only early/late-stages of development in which software is actually built but also integrated stages including the stage of formulating system specifications and the actual operation stage as a system lifecycle process in which development and operation are integrated. The process of grasping quality goals in the problem domain analysis stage prior to system design and the processes of monitoring behaviors in operation, obtaining additional data and carrying out additional learning are positioned as one important stage for system development and treated as a part of initiatives for quality management.

The whole lifecycle is modeled as a hybrid process integrating agile (iterative) development process in which the progress is managed based on quality inspections (tests) and the whole top-down V-shaped development process focusing on process management (Figure 5).

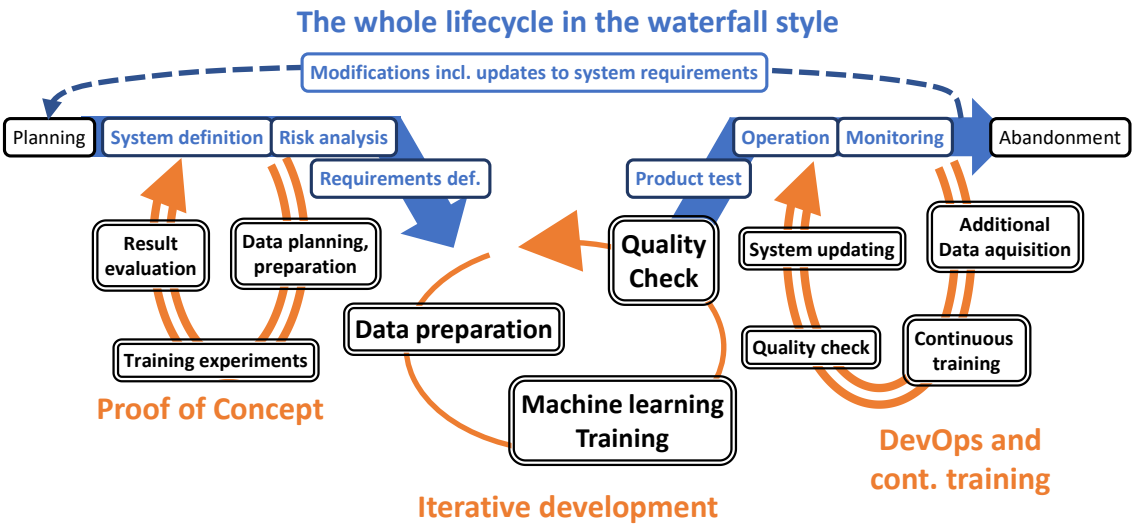

Figure 5: Conceptual diagram of mixed machine learning lifecycle process

Specifically, works to analyze requirements necessary for the whole system quality are regarded as a flow process similar to a top-down process and assumed to be analyzed in detail

as an independent process before actual system development and data reduction start. Reworking and iterative works at this stage suppose that the same level of consistency is ensured as final results even if works are restarted from the middle of the development process. Moreover, the process of obtaining additional data in operation and the monitoring process are positioned as a part of the iterative quality management process based on the concept of DevOps and should be compatible also to the concept of the top-down operation phase in RAMS specifications where necessary.

The stage of so-called PoC development which is seen often in the development of AI is categorized as a part of the process of the prior analysis stage leading to the definitions of requirements as an important quality management stage. This is a concept of identifying and categorizing requirements for quality again, when knowledge and data obtained from PoC are utilized, so that quality management in the main development stage and free explorative development in the PoC stage are balanced. In the actual development process, a trouble of re-categorization can be saved by partially introducing the concept of quality management of the main development stage in the PoC stage.

The lifecycle process in this guideline is a reference model, and development processes carried out by respective developers can be designed as their own. This reference model intends to help readers understand individual stages of the development process created in accordance with different circumstances in comparison to the Guideline and find out how quality management technologies included in each chapter of the Guideline correspond to development stages taken care thereby.

(Note)

The development process of AI has been often categorized as a non-waterfall iterative process. When iterative training is given, various works such as addition of collected data, change of selection and adjustment of parameters are carried out other than changes in algorithm implementation codes. Sometimes, the goals are achieved by modifying the principles for implementation and specifications based on those changes, giving training in accordance with new specifications and improving the accuracy. On the other hand, “gates of quality” are often established in the form of check before operation and inspections on orders, even if products that require quality adopt a circular development process model in reality.

A model shown in Figure 6 exemplifies, based on the development process of AI understood as a iterative system, its relation with the mixed process shown in Figure 5 which the Guideline basically assumes.

To be more specific, several development tests conducted until policies for implementation (also called specifications for implementation) are concretized are mapped to several circulations in the PoC stage to examine the quality. Then, one or several tests immediately prior to the operation stage leading toward training/quality tests and release based on the final specifications are positioned as “main development stage” in the mixed process.

At times, the specifications need to be modified again after going through the main development stage. From the perspective of waterfall process, this is a step back from the implementation stage to the requirement analysis stage, but from the perspective of iterate process, we do not need to consider it as a serious step back, because “it was found out that the product was still in the PoC stage”. The PoC development stage has an aspect of implementation forecast and knowledge obtained in this stage is reflected implicitly in the main development stage. Therefore, even if this process is considered as a waterfall type, efforts for implementation in the PoC development stage are not in vain.

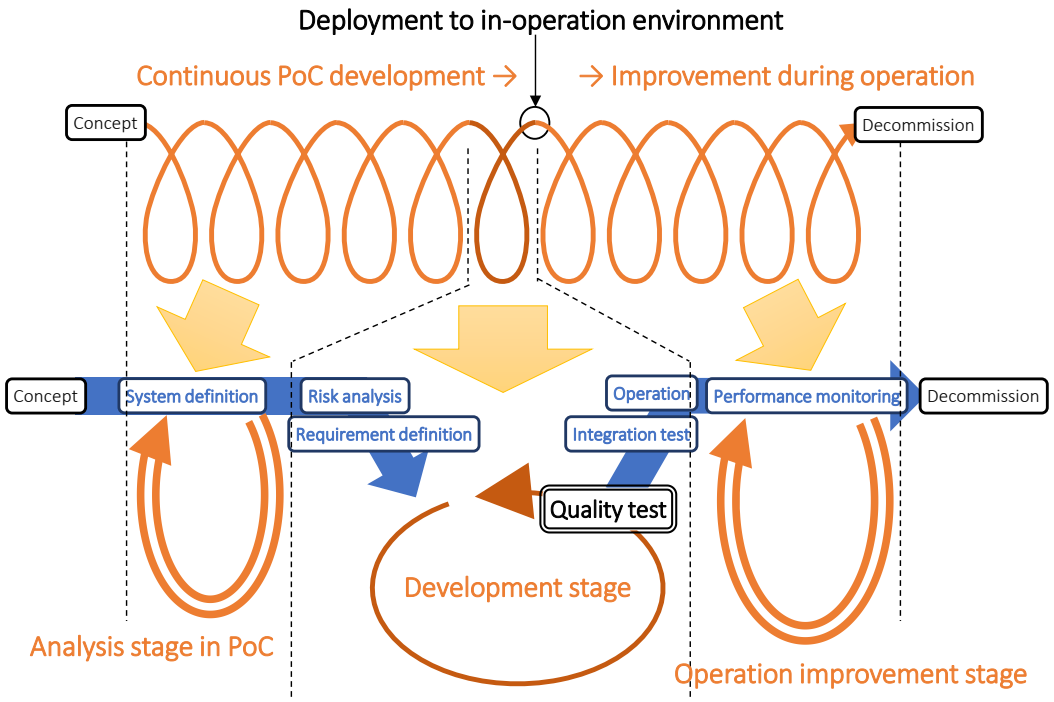

Figure 6: Relationship between the circular process modeling and the model in the Guideline (example)

(Note 2) Section 4.1.1 (Page 42) describes changes in operational situations and the concept of operation with various gates as an example of application.

development process by multiple stakeholders

In the actual development of AI, works may be divided in various forms. For example, a service provider itself may take care of planning, development, and operation, outsource only the training stage or entrust system design and build-out of a trained model and incorporate it into systems. In the quality management process defined in the Guideline, a development entruster responsible for product planning plays a central role and quality management

activities are carried out by everyone including developers, etc. Management works for individual stages in the process are shared upon agreement between workers. As a result, a development entruster or development entrustee may take responsibility for setting qualities in use and external qualities based on the purpose of applied systems depending on the form of division[7]. In any case, it is important to reach agreement to ensure sufficient quality for end users together, and clearly reflect the share of costs in the contract so that activities in the quality management process that continue until the operation starts are not disrupted.

Sections 5.2 and 5.3 explains some sharing models and remarks when several entities involve in development such as consignment development and delta development.

1.9 Relations with other documents and rules

"Social Principles on Human-centric AI"

In Japan, the Cabinet Office has defined an ideal relationship between operators and developers of machine-learning based systems and the society and general public in the form of "Social Principles on Human-centric AI"[8].

In relation to those Principles, the Guideline are primarily positioned as non-binding guidance which sorts out technical matters developers themselves refer to and practice in order to improve the quality and prevent safety and security from being compromised, when operators and developers fully understand those Principles and actually develop machine-learning based systems and machine learning components therein (Figure 7). Moreover, the Guideline are, together with other guidelines and future international standards, also positioned as an element that constitutes "Social Principles of Human-centric AI" mentioned therein.

[7] In cases where a requirement for validation that "certain performance indicators (accuracy, etc.) should be achieved on data given by a development entruster" is established as a form of order requirements common particularly in the development of AI, we need to take note that the development entruster is responsible for ensuring so-called "data quality ("the data is appropriate for learning consistent with the purpose of products") as explained in 1.7.1 to 1.7.4 of the Guideline. In cases where the development entruster is not capable of validating or ensuring data quality, it is necessary to explicitly entrust a part of the work in the form of "design support work" and consider a possibility that the system cannot be built due to the lack of data quality when it is actually trained (the work is returned to the development entruster).

[8] Social Principles of Human-centric AI, Integrated Innovation Strategy Promotion Council, Cabinet Office, March 29, 2019, https://www8.cao.go.jp/cstp/aigensoku.pdf.

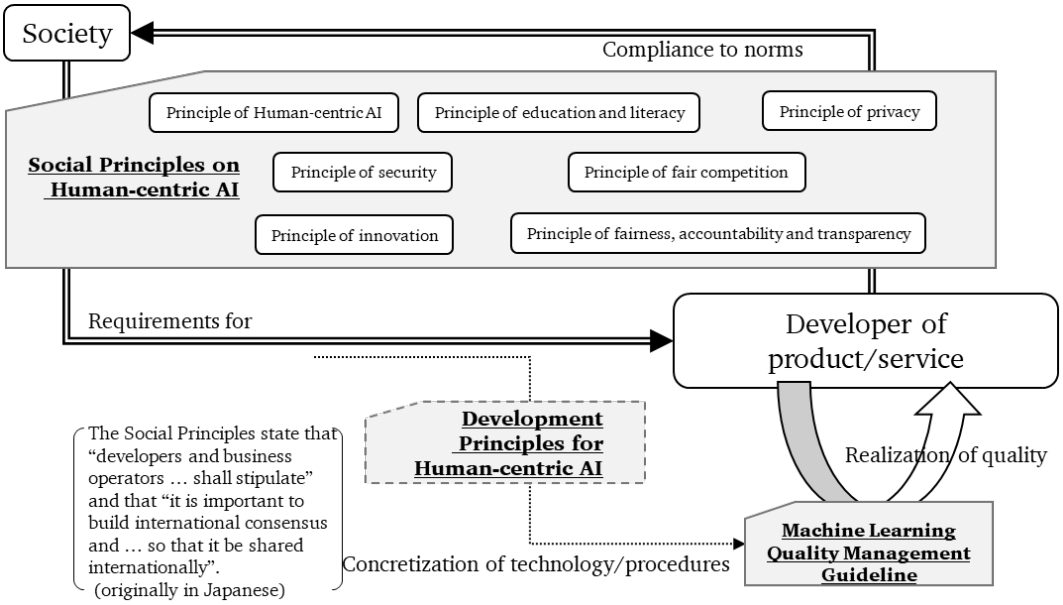

Figure 7: Relation with "Social Principles on Human-centric AI"

AI-related rules and guidelines of foreign governments and international organizations concerning AI technology

In addition to the above principles, norms for social nature, safety and ethics concerning the development and use of AI technology have been documented recently in different forms[9,10]. As for their relationship with the Guideline, those social norms are positioned under verbalized documents as in the case of the "Social Principles of Human-centric AI" mentioned in the previous section and categorized to present one specific method for realizing some of them in system development.

1.10 Structure of the Guideline

The structure of the remaining parts of the Guideline is as follows.

- Chapter 2 sorts out the scope of the Guideline and their relationship with existing standards again.
- Chapter 3 sorts out the details about qualities in use mentioned in Section 1.5 including decisions on leveling.

[9] Organization for Economic Co-operation and Development (OECD). OECD Principles on Artificial Intelligence, May 2019. https://www.oecd.org/goingdigital/ai/principles/.

[10] The High-Level Expert Group on AI, European Union. Ethics guidelines for trustworthy AI. 8 April 2019. https://ec.europa.eu/digital-single-market/en/news/ethicsguidelines-trustworthy-ai.

- Chapter 4 shows reference models with regard to the development processes whose outline was explained in Section 1.8.
- Chapter 5 mentions specific methods of operating and applying the Guideline.
- Chapter 6 sorts out the internal quality characteristics mentioned in Section 1.7 in more detail.
- Chapter 7 sorts out possible ways of realizing each internal quality characteristic described in Chapter 6.
- Chapter 8 shows reference information such as the relationship with other guidelines.
- Chapter 9 shows the process of analysis and concept until the Review Committee of the Guideline clarifies the internal qualities listed in Section 1.7 (and Chapter 6) as reference information.

(Note)
The Guideline can be read in the following ways (other than reading in order from the beginning).

Read Chapter 5 to understand the outline of the procedures for processes.

Read Chapter 4 to determine the quality level referred to in the development process.

Refer to each section of Chapter 6 (and the table in Section 10.1) to clarify check items required at each level.

Refer to each section of Chapter 7 as needed to grasp specific concepts of the above check items and applicable technology.

The definitions of the development stage, etc. referred to in each section are summarized in Chapter 4.

2 Overview

2.1 Scope of the Guideline

Products and systems subject to the Guideline

Products and services subject to quality management in this Guideline are those that use machine learning technology for the building of some of their software components among all information-processing systems such as industrial products, consumer equipment and information services (hereinafter referred to as “machine-learning based system” in this Guideline).

This Guideline envision that machine-learning components are mainly implemented in supervised learning. Although the basic approach can be also applied to other methods of implementation such as unsupervised learning, semi-supervised learning and reinforcement learning, related content on specific method of handling them will be added at the time of future revisions.

Products and services subject to quality management

The quality characteristics which this guideline set goals, manage, guarantee is the external quality corresponding to the effect on the system which implements the software component built with the machine learning technology (hereinafter referred to as "machine learning component") that is the internal component constructing the machine learning based system.

On the other hand, quality management mentioned in this Guideline directly target internal qualities machine-learning components have.

We consider that qualities in use of the whole system are realized comprehensively by external qualities of elementary parts of that system. Moreover, external qualities of each component depend on the quality of internal components/parts and internal quality characteristics which are the internal quality of those components themselves. Quality management is realized by sorting out the requirements for machine-learning components as internal quality characteristics and by managing the process toward the achievement thereof (Figure 8). However, among system components, software and hardware other than machine-learning components are explained only within the necessary scope of their relationship to this Guideline for the purpose thereof.

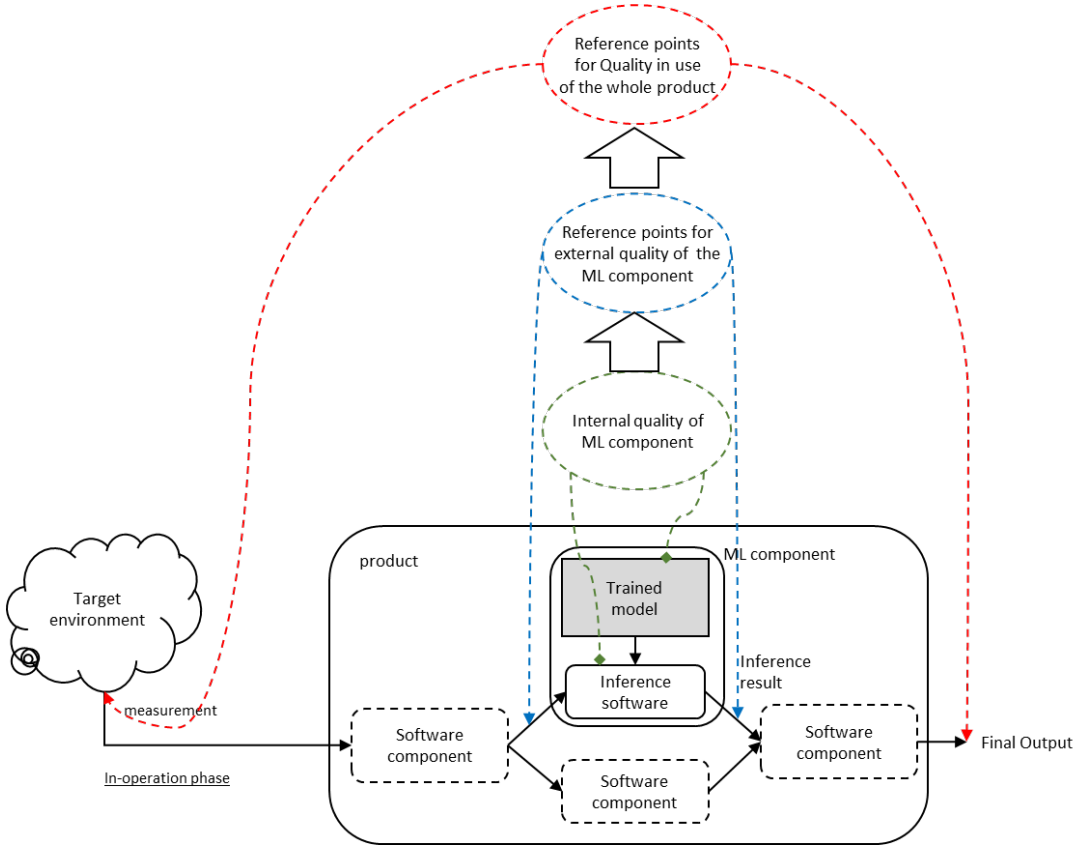

Figure 8: Concept of quality requirement of whole AI-based system and means for realizing them

Scope of quality management

In the Guideline, the term “quality management” refers to a process of quality activities across the lifecycle from the planning stage to the operational stage of machine-learning based systems and includes such meanings as setting of quality goals, planning, confirmation, quality assurance and management. Although the meaning of “quality management” is wider than quality assurance or management of general software, it does not include organization planning and management of responsibilities or resources which are included in the meaning of quality management defined in ISO9001.

2.2 Relationship between this Guideline and the other standards for system quality

Mainly, this part sorts out the relationship between the Guideline and the existing international standards for information system quality. Also See Section 1.9 for their relationship with guidelines equivalent to superordinate concept such as sociality.

The Guideline present quality management methods for a variety of applications of

machine learning and define systems that require safety as complement or extension of some of conventional standards for functional safety (IEC 61508-3, IEC61508-4, etc.).

- For systems that strongly require functional safety, functional safety standard IEC61508-1 or other equivalent standards are primarily applied.
- For ensuring the required safety level of the machine-learning component in the targeted system, this guideline sorts out the issues and methods for ensuring safety that is unique to the machine-learning and propose the methodology for complementing or replacing the methods introduced in IEC 61508-3 comparing with the conventional software.

Security standard ISO/IEC 15408

The Guideline and information system security standards such as ISO/IEC 15408 are independent, and they should be applied simultaneously when needed. Information security is a mandatory requirement for systems that require integrity and availability in order to realize risk avoidance included in the Guideline, but the basic countermeasures defined in those standards are also applicable to machine-learning based systems.

Software quality model ISO/IEC 25000 series

The ISO/IEC series which define software quality models, especially ISO/IEC 25010, can be partially applicable to machine-learning based systems.

Since product quality concerning software components are analyzed from the viewpoint of conventional software, quality properties sorted out in the above standards are supposed to be achieved also in machine-learning components. However, since there are differences from product quality and “analysis and decomposition into components focused in the Guideline”, we assume that there is no clear relationship between them. Although there is a proposal to improve the above standards to machine learning and AI at an international level, we continue to discuss this matter in view of future standardization.

2.3 Definitions of terms

Terms related to machine-learning based system structure

1. Machine learning based systems / systems using machine learning technology

   A system containing software components (machine learning components, 2.3.1.2) implemented by applying machine learning technology.

   (Note) The common terminology “machine learning systems” refers to machine-learning based systems or machine learning components (2.3.1.2) depending on the context.

2. Machine learning component

A software component implemented by applying machine learning technology. A machine learning component realize the functions of trained machine learning models (2.3.1.5) as software. Generally, this component consists of prediction/inference software component (2.3.1.6) implemented as software and trained machine learning model (2.3.1.5) incorporated as fixed input.

3. Machine learning algorithms

An algorithm that provides the calculation method for prediction/inference of machine learning and for obtaining said calculation method through training. Each of the calculation methods correspond to prediction/inference software component (2.3.1.6) and trained machine learning model (2.3.1.7).

There are a lot of types of machine learning algorithms, such as neural networks, support vector machines and decision trees and so on. Appropriate algorithms are selected based on the type and purpose of knowledge which machine learning components are to obtain.

4. Hyperparameter

A setting value input in training software component (2.3.1.7) to execute machine learning training. It may need to be adjusted in accordance with the progress of training. In some cases, a hyperparameter may not exist, adjustment of hyperparameter may be omitted intentionally, or a technique to adjust it automatically may be used.

(Note) As regards the diversity of mathematical structure that can be subject to machine learning, which scope is fixed as “machine learning algorithms” prior to the start of training and which scope is set/adjusted during training as “hyperparameters” are arbitral depending on how to implement software and selection of policies for training by developers. In some cases, the structure of calculation formula itself is treated as data and subject to automatic adjustment for optimization as a part of hyperparameters.

5. Trained model / trained machine learning model / knowledge base / trained parameter

Information required as output for training that defines functional operations of machine learning.

(Note 1) “Machine learning model” in the Guideline refers to trained machine learning model or in-processing data for obtaining said model.

(Note 2) What is simply called “parameter” in the context of machine learning technology often refers to this trained machine learning model or numerical data included therein.

(Note 3) Mathematical structures such as neural networks and Bayesian networks are sometimes called as “graphical model” or “statistical model”. Accordingly, the terms such as “selection of machine learning model”, “model design” and “untrained model” may be

used for the selection of machine learning algorithms and the setting of their parameters.

(Note 4) In literatures, the term “trained model” is used either in comparison with trained parameter as “output of model training” or as trained machine learning model implemented as specific software. However, in this Guideline, the term “trained model” is used for the former meaning (from the viewpoint of clearly distinguishing output from training itself and pre-implemented prediction/inference software component), while the latter is called “machine learning component”.

(Note 5) When the context on machine learning or AI is clear, there may be no problem in calling it simply as “trained model”.

6. Prediction/inference software component

A part of machine learning components used during operation which is fixed as the implementation of software of machine learning models. This component receives a trained machine learning model (2.3.1.5) as static (semi-static) input and data obtained from real environments as dynamic input.

7. Training software component

A component to generate trained machine learning models (2.3.1.5) using training datasets. This component implicitly corresponds to prediction/inference software component (2.3.1.6) as the implementation based on different aspects of the same machine learning algorithms so that they are usually used as a pair.

Although a training software component is not included in machine learning components used during operation in many cases, there are the cases (e.g. a system that runs the online learning during operation and reinforcement learning) where the training software component is included as a part.

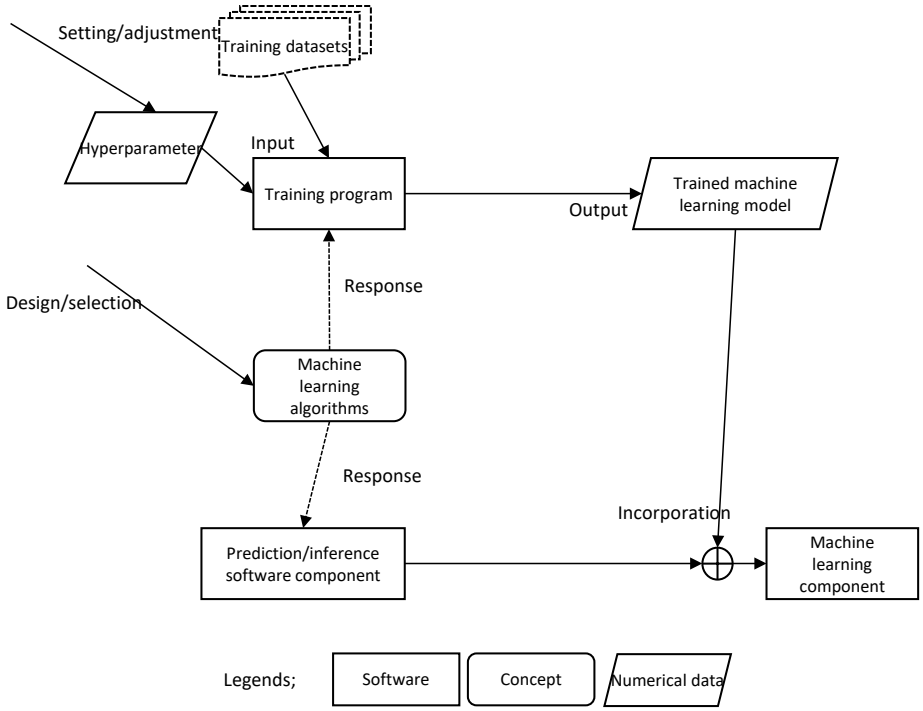

Figure 9: Relationship between terms concerning the structure of machine learning based systems

Terms related to stakeholders of development and their roles

1. Service provider

An entity that uses machine learning based systems for its own purpose or for selling or providing services for customers. Moreover, as regards a business structure in which software providers, etc. plan and develop in advance systems in anticipation of services provided or used by others and sell them as packaged or customized systems, those entities that engage in planning and development are treated according to the service providers.

2. Self-development entity

A product development entity designs and implements the machine learning components.

3. Development entruster

A product development entity that asks others to implement machine learning components regardless of the form of contract (service/entrustment).

4. Development entrustee

An entity that implements machine learning components upon request from the development entruster.

5. (Development) stakeholders

All stakeholders that engage in development and operation of machine learning

based systems, including production operators, self-development entities, development entrusters and development entrustees.

Terms related to quality

1. Harm avoidance (Risk avoidance/safety)

   A quality characteristic of avoiding negative effects such as human damage, economic loss and opportunity loss on operators, users of products or third parties caused by undesirable judgements of machine learning components. The improvement of “risk avoidance” corresponds to a concept of “risk reduction” in the safety field.
   [Defined in 1.5.1 and 3.1 in the Guideline]

2. AI performance

   This characteristic allows machine learning components to output expected by machine learning based systems and their users at a higher precision/probability than the average in the long run. AI performance is evaluated based on comprehensive performance rather than whether the output is acceptable or not. (Defined in Sections 1.5.2 and 3.2 in this Guideline.)

3. Fairness

   The output or distribution of machine learning components is not affected by differences in some of the attributes belong to the people or others (or the effect is kept sufficiently low).
   (See Sections 1.5.3 and 3.3 of this Guideline)

4. Attack resistance

   This characteristic prevents machine learning components (or machine learning based systems) from reacting contrary to the operator’s expectations (in line with an attacker’s intention) in relation to input data or external environments built intentionally by the attacker.

5. Ethicalness

   The behavior of machine learning based systems is appropriate in the human-centered society.

6. Robustness

   This characteristic allows systems to maintain their performance level under any circumstance.

Terms related to development process

1. System lifecycle process

A process management model looking down the flow from the planning of systems to the end of operation and disposal.

[Source of definition: ISO/IEC/IEEE 15288]

2. Agile development process

A collective term of agile and value-driven development approaches.

(Note) This term originates from the Agile Manifest announced in 2001 and can be utilized in various forms such as scrum and XP.

3. PoC (Proof of concept)

Preliminary development activities carried out with the aim of validating feasibility of ideas to be realized and solution for the problem instead of being realized as products.

4. Systems engineering

An approach across several fields to deploy balanced system solutions in response to diversified needs of stakeholders. It applies both management process and technological process and strikes a balance between them to reduce risks that affect the success in projects.

5. RAMS (Specification and demonstration of Reliability, Availability, Maintainability and Safety)

In the Guideline, RAMS refers principally to a concept of the comprehensive system lifecycle process specified in the standard IEC 62278 (EN 50126) about the reliability management process in the field of railway service.

(Note) “The RAMS standards” may refer generally to IEC 62278 itself.

Terms related to use environment

1. Environment

In the Guideline, this term is used in three contexts: 1) External environment, 2) computing environments, and 3) operational environment.

2. External environment

The physical environment or cyberspace to be the source of input to the machine learning based system, and there is interference from the environment to the systems or from the systems to the environment.

3. Open environment

The external environment whose system operation is greatly affected by the condition of the people other than users or the things such as nature.

4. Environmental condition

A characteristic which distinguishes the differences in conditions of external environment. From the viewpoint of machine learning quality management, we focus especially on changing characteristics such as the status of hazard and risks.

5. Computing environment

Software execution environment where machine learning components (prediction/inference software components) and training software components are executed. It may include hardware environment, operating systems and middleware on which computing environment is based depending on situations.

6. Operational environment

Operational environment includes computing environment and operational structure of humans.

7. In-operation environment

Production environment where machine learning based systems are provided.

8. Runtime (computing) environment

Computing environment of computers, clouds and IoT devices where machine learning based systems including machine learning components are operated in in-operation environment.

9. Development environment

Computing environment where machine learning components are built and modified and such modifications do not affect actual operation directly, and operational environment including computing environment thereof.

Terms related to data used for building machine learning

1. Training dataset

A set of data used for training of machine learning models in the iterative training phase.

2. Training data (instance)

Data included in training datasets.

(Note) Generally, the term “data” is used for both single instance and several instances (a set of instances). In this Guideline, the term “data” is used only for the former, while the term “dataset” is used for the latter, in order to avoid this ambiguity. That is, the term “dataset” is used for the case that it is treated as one set which can be the target of discussion about the distribution and the total quantity of a set. On the other hand, the

term “data” is used to refer to individual data points or discrete data points before it is captured as a block. The relationship between these terms is described by expressions of the relationship between set and element such as “added to dataset” and “included in dataset”.

3. Validation dataset

A set of data used for evaluating convergence of machine learning models in the iterative training phase.

4. Test dataset

An set of data used by machine learning models built as input in tests as one or more means for checking the desired quality and performance in the quality check/assurance phase.

5. Adversarial example(s)

Data built with intention so that the inference results to be different from assumptions/intuition are output when it is input into the machine learning components.

6. Overfitting

Trained machine learning models are adapted excessively to training data and it becomes impossible for them to return desired output in response to input data other than training data.

7. Attribute

Each itemized element to analyze and classify the characteristics of environmental conditions in ML problem domain analysis.

8. Value

Types of specific characteristics of environment included in each attribute classified in the ML problem domain analysis.

9. Labels

Identifiers belonging to data used for classification problems in supervised learning, indicating correct classes which they belong to.

Other terms

1. Software regression

In software engineering, this term means that, when software is improved, it stops to operate as expected in response to input which did not cause problem before.

(Note) In machine learning and statistical analysis, the term “regression” is often used as “regression analysis”.

2. SIL (Safety Integrity Level)

The classification of levels related to the achievement of functional safety of products as specified in IEC 61508 [Source of definition: 61508-1]

3. KPI (Key Performance Indicator)

An indicator which quantifies the attainment level of functional requirements to be attained by output from machine learning components through machine learning based systems.

4. Additional training

A form of operation to collect data and carry out additional training for machine learning during operation and to update trained machine learning models when necessary.

(Note) This is a concept including not only “online learning” described below but also “additional offline learning”.

5. Online learning

A form of system implementation in which additional training is carried out without going through the validation process in development environment and its results are reflected in operation environment.

(Note 1) In this Guideline, a form of carrying out additional learning in development environment and updating model parameters by such means as firmware updates after going through the assurance process without online learning is called “additional offline learning”.

(Note 2) The term “continuous learning” seems to be used as a synonym for either “additional training” or “online learning” here.

3 Levels for external quality characteristics

As described in Section 1.5 of this Guideline, this guideline defines three aspects of external qualities for machine learning components contained in machine learning based systems. In this section, we set target levels of quality for each aspect. The procedures for determining target levels in development will be provided in Section 5.1.2.

3.1 Safety / Risk avoidance

For the safety or risk avoidance aspects, quality levels are classified into seven "AI safety levels" (AISL 4, AISL 3, AISL 2, AISL 1, AISL 0.2, AISL 0.1, AISL 0). Throughout this guideline, causes of the "safety" risks include both physical hazards such as injury of humans and economic hazards as well.

The required AISL shall be determined using Table 1 and Table 2, depending on the nature of corresponding hazards[11]. If the cell containing the "*****" in Table 1 is referred, *the functional safety standard IEC 61508-1 or any similar standard for specified application fields is consulted* first, safety function is assigned for the components in the system, and the SIL 4–1 assigned to the machine learning component in question (or any equivalent assignments in other related standards) is translated to AISL 4–1 respectively. Optionally, even in the other cases, stakeholders can apply risk analysis based on ISO 61508-1 or others and assign AISL based on the SIL evaluation result.

In the meantime, this Guideline does not expect AISL 4 or 3 is directly assigned to any machine learning component. In that case, the overall system design should be reconsidered to reduce risks caused by the machine learning components in question.

Table 1: Estimation of AI safety levels for human-related risks

Expected severity of hazards/possibility of avoidance	Impossible to avoid	Possible to avoid through monitoring by humans	Always requires check/manual operation by humans
Simultaneous deaths of multiple persons	*	*	*
Death or injury of one person	*	*	*
Injury with left disability	*	AISL 2	AISL 1

[11] Each AISL is described to correspond roughly to safety integrity levels 4~1 of the standard IEC 61508 (functional safety). Furthermore, to precisely recognize safety requirement strength to be categorized as "No SIL applicable", the corresponding AISLs are divided into three classes as 0.2, 0.1 and 0.

Serious injury	*	AISL 1	AISL 0.2
Minor injury	AISL 1	AISL 0.2	AISL 0.1
Minor injury (where possible victims can avoid the hazard easily)	AISL 0.2	AISL 0.2	AISL 0.1
No damage is expected	AISL 0	AISL 0	AISL 0

Table 2: Estimation of AI safety levels for economic risks

Expected severity of hazards/possibility of avoidance	Impossible to avoid	Possible to avoid through monitoring by humans	Always requires check/manual operation by humans
Damages affecting continuity of business entities	(AISL 4)	(AISL 3)	AISL 2
Serious damage that undermines business operations	(AISL 3)	AISL 2	AISL 1
Considerable/specific damage	AISL 2	AISL 1	AISL 0.2
Minor loss of profit	AISL 1	AISL 0.2	AISL 0.1
No damage is expected	AISL 0.1	AISL 0	AISL 0

3.2 AI performance

For the AI performance aspects, quality levels of components are assigned by the agreement between stakeholders based on the following criteria.

- AIPL 2 (mandatory requirements):
  - When it is indispensable or strongly expected that the product or service satisfies certain performance indicators (e.g. accuracy, precision or recall) for the system's operation.
  - When the fulfillment of the certain performance indicators is clear stated as requirement in contracts.
- AIPL 1 (best-effort requirements):
  - When certain performance indicators are identified as an factor for satisfying purpose of the product, but AIPL 2 is not applicable. For example, when shorter development period is beneficial, or when gradual improvement of performance to a satisfactory level during in-operation phase is acceptable.

- AIPL 0
  - When performance indicators are not specified at the beginning of development, and discovery of a performance indicator itself is the purpose of development.
  - When the development completes at the Proof of Concept stage.

3.3 Fairness

Quality levels of components for fairness aspects are determined based on the following criteria.

- AIFL 2 (mandatory requirements)
  - When a certain level of fair treatment is required according to the laws, regulation or equivalent de-facto social guidelines.
  - When the product deals with personal data and its output directly affects right of individuals.
- AIFL 1 (best-effort requirements)
  - When it is possible to define the requirements that there is no bias into the product or the services.
  - When there is possible demand for the explanation of the fair treatment provided by machine learning component (or AI in general) that might affect social acceptability or operation of the product.
- AIFL 0
  - When there are no identifiable requirements for fairness of the product or service.
  - When the product or service would be affirmative even if their outputs are unfair or non-uniform.
    (For example, if there is an application to detect some mechanical defects, if a detection rate for some specific defect class is relatively high, there are no need to *decrease* the detection rate to match with those for other unfavored classes).

4 Reference models for development processes

This Chapter explains the development process reference models for machine learning based systems which the Guideline refer to on the premise of discussions.

The reference models mentioned in this Chapter do not force developers to adopt a certain development method of machine learning based systems. Rather, they aim to add referable names to internal components and development processes of general machine learning based systems and compare recommendations provided for in the Guideline with actual development processes. By this way, the respective unique configurations and development processes can be compared with this model.

As shown in Figure 5 (page 22), this Guideline roughly divides the development process of machine learning components (and the principal parts of machine learning based systems that contain them) into three phases.

4.1 PoC trial phase

PoC trial phase as a development lifecycle process is, in the earliest stage of system development, sorted out as the preparatory stage to identify a possibility of performance attained by a system, quality degradation risks and its possible cause in advance of defining the functions of the whole process. Quality management activities in this phase are *sorted out as important preparatory works for quality management activities in the main development phase* in the context of quality management defined in the Guideline. From the viewpoint of data science, analytical activities in this phase are closely associated with subsequent risk analysis and requirements analysis and their results are often used in the subsequent main development phase. In order to attain “sufficiency of requirements analysis” (Section 6.1), it is essential to make efforts in the PoC trial phase. This Guideline stipulate that the adequacy of those activities is checked again in each step of the main development phase ultimately. This allows quality management activities to be conducted on a trial-and-error basis in the PoC phase without restrictions.

Handling of PoC phase including trial operation

Moreover, “Proof of Concept” does not only mean simple data collection and analysis, trial training, and build-out of models but includes what is tested in an in-operation environment, although final products will be used in different situations (for example, under monitoring by humans). Furthermore, the operation stage may be divided so that quality requirements continuously shift to different operational conditions. The Guideline consider such type of development with physical operation divided in several stages as equivalent to the whole

development lifecycle including the “main development phase” in relation to the system building stage in each stage and treat all achievements in the early operational stage as knowledge obtained from the PoC stage in later stages of development (Figure 10).

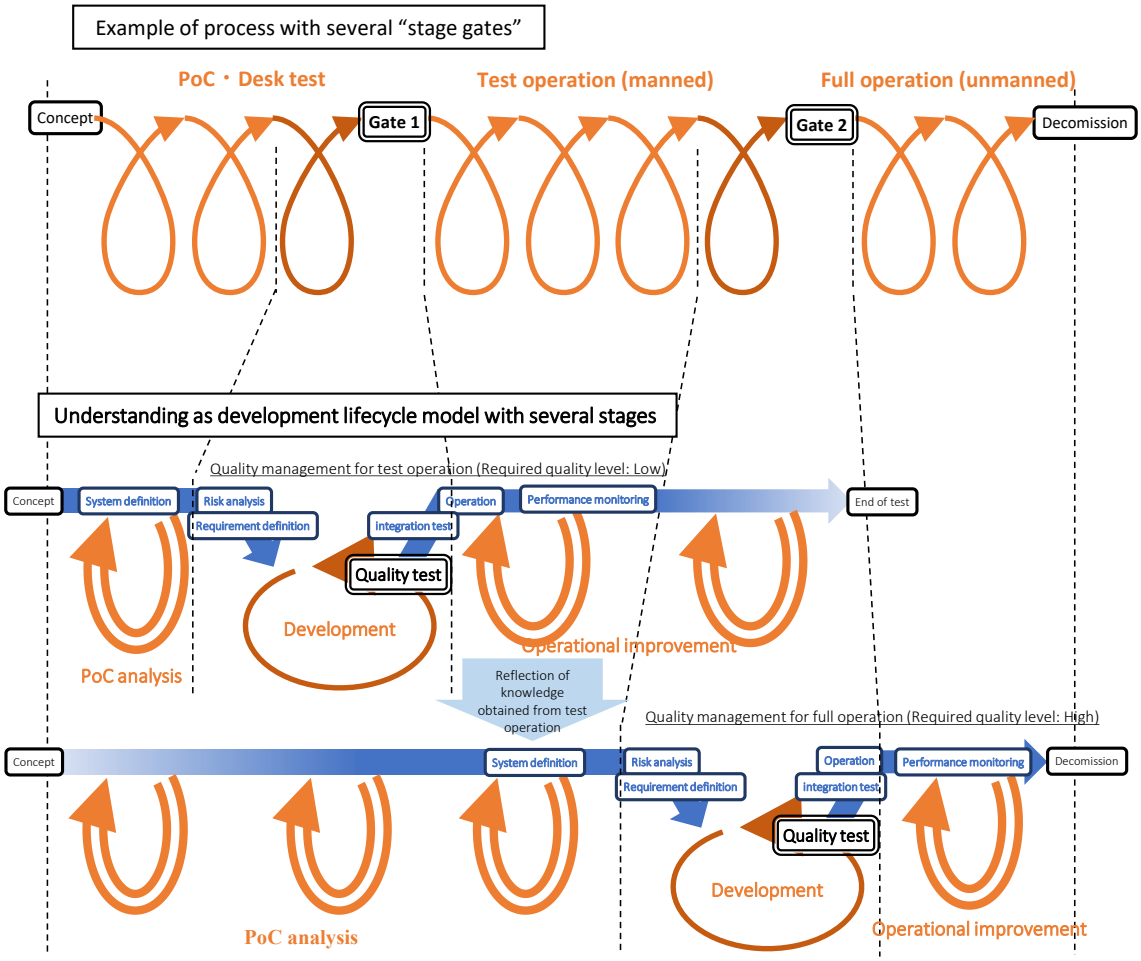

Figure 10: Handling of development process with several operational stages (Example)

4.2 Main development phase

“Main development phase” is a process from when system’s functional requirements are confirmed by means of the efforts in the PoC trial phase to immediately prior to the in-operation phase. This phase includes system definition equivalent to very early stages of development of conventional systems and integrated tests prior to final operation and shipment.

In the main development phase, a lifecycle which mixes a part of conventional waterfall early-stages of development and iterative development processes specific to machine learning components is defined as a model. To be more specific, system requirements for each component including functional definitions and risks analysis of the overall system, analysis

and decomposition of effects of components and machine learning components are decided. A series of processes which sets the goals for the last integrated tests prior to shipment are sorted out pursuant to the conventional waterfall model. Moreover, components of software and hardware other than machine learning components are assumed to adopt the conventional waterfall development model or agile development process [12] capable of ensuring the same level of quality as needed in late-stages of development. On the other hand, a development process of specific machine learning model equivalent to "late-stages of development" of machine learning components defines the iterative development process model again in the following section.

Machine learning model building phase

After risks which machine learning components have to deal with as well as quality requirements are specified in the main development phase, a machine learning model is built and its external qualities are tested (using indicators for internal qualities) and this process is repeated until the model passes a test. This is called "machine learning model building phase".

A model which breaks down this phase into more specific process is shown in Figure 11. Each process of this model aims to correspond it to the descriptions in the Guideline by comparing a development process specific to each developer with this model so that it is not to restrain the development process unambiguously.

[12] "Agile development process" here means test-driven development or another non-waterfall style of development process which includes quality assurance activities comparable to those of waterfall processes so that expected levels of quality is reasonably explainable. It does not include all forms of non-waterfall development.

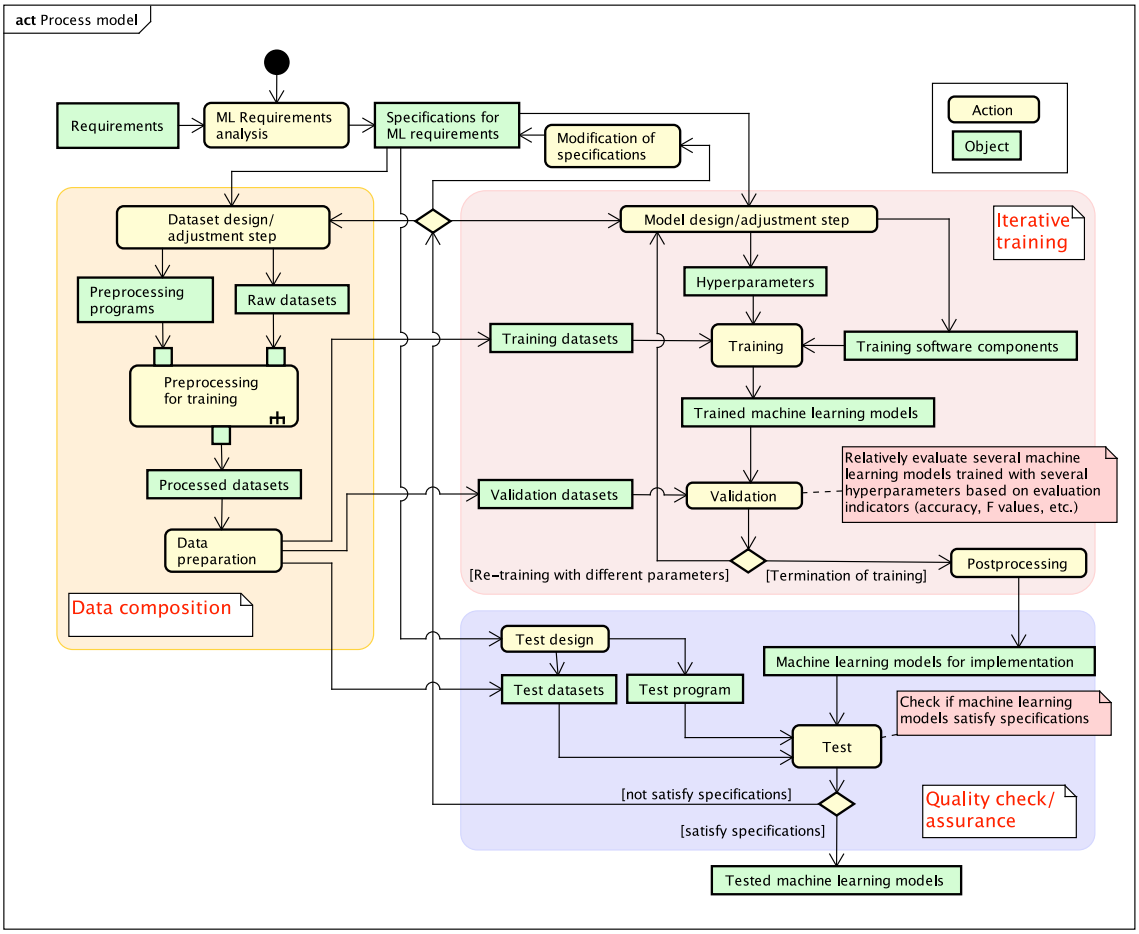

Figure 11: Process model in the stage of machine learning building

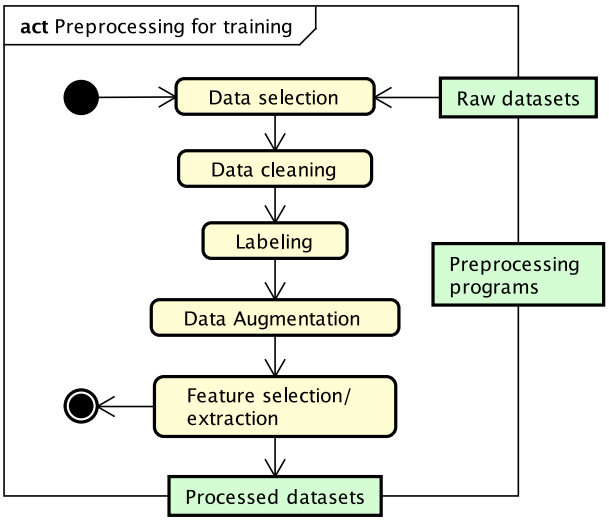

Figure 12: Model of preprocessing for training

4.2.1.1 ML requirements analysis phase

In the “machine learning requirements analysis phase”, conceptual quality requirements for machine learning components are concretized by sorting out characteristics of data input to or output from machine learning components and re-organized as specific requirements for build data in the ML model building phase. In the actual development of machine learning based systems, performance requirements (quality requirements) are specified in many cases based on data characteristics or specific datasets. By explicitly including this analysis phase, it becomes possible to sort out a specific method of managing data quality and the relationship with other components in terms of quality management. This phase is closely related to internal qualities mentioned in Section 6.1 and Section 6.2.

4.2.1.2 Training data composition phase

In the training data composition phase, training, validation, or test datasets are generated in accordance with specifications for ML requirements. This phase is explained below in line with “composition of training data” shown in the upper left side of Figure 11.

4.2.1.2.1 Dataset design/adjustment step

In the dataset design/adjustment step, sufficient data is collected for each case in accordance with the specifications for ML requirements and design is carried out to generate datasets suitable for training. Output of this step is unprocessed datasets and preprocessing programs. In cases where it is found that trained machine learning models do not satisfy the specifications after going through training, the process may return to this step to adjust the design of datasets.

4.2.1.2.2 Preprocessing step for training

In the preprocessing step for training, raw datasets generated in the dataset design/adjustment step in Section 4.2.1.2.1 are processed into datasets suitable for training. Figure 12 shows an example of preprocessing for training. Actually, there are cases where each process shown in the insertion diagram of the pre-processing for training in Figure 12 is performed in random order, in parallel, or repeatedly, but here, as an example, it is explained in the order shown in the figure.

Data selection

Datasets used for training and evaluations (validation and tests) are selected taking into account coverage and uniformity from a large quantity of raw datasets. Since coverage and uniformity are traded off, the balance between them should follow the specifications for ML requirements.

Data cleaning

Noise removal, data forming, complement of missing values and removal of outliers are carried out so that original characteristics of selected datasets can be trained.

Labeling

Labels of correct answer (for teachers) are placed to cleaned datasets in accordance with the specifications for ML requirements.

Data Augmentation

In order to ensure the amount of training data or test data and to prevent overfitting, data variants are generated and added to datasets. For example, controls such as rotation, scale-down, reverse, change in luminosity and background replacement of image data are carried out to generate variants.

Feature selection / extraction

Feature extraction is performed to add new features useful for model training, and feature selection is performed to reduce unnecessary features. For example, feature extraction uses techniques such as calculating frequency components by Fourier transform and finding the principal components of a plurality of features. If useful features are obtained, in many cases more unnecessary features can be reduced by feature selection.

4.2.1.2.3 Data preparation step

In the data preparation step, processed datasets generated are divided into training, validation and test datasets. Training datasets and validation datasets may be replaced during iterative training.

4.2.1.3 Iterative training phase

In the iterative training phase, machine learning models are designed in accordance with the specifications for ML requirements and training and validations are repeated using datasets created in the training data composition phase. This phase is explained below with reference to the flow of “iterative training” shown in the top right of Figure 11.

4.2.1.3.1 Model design/adjustment step

In the model design/adjustment step, hyperparameters necessary for training (including the structure of learning models, learning algorithms and various parameters) are designed in accordance with the specifications for ML requirements and training software components are created. Hyperparameters may be adjusted in response to the results of validations and tests.

4.2.1.3.2 Training step

Machine learning models are trained using training datasets based on designed hyperparameters.

4.2.1.3.3 Validation step

Validation datasets are input into trained machine learning models to evaluate models based on evaluation indicators (accuracy, precision, recall, F values, etc.) of learning models. Generally, several machine learning models are trained with several hyperparameters and evaluated relatively to judge their adequacy.

4.2.1.3.4 Postprocessing step

In the postprocessing step, models are transformed to adapt trained machine learning models to in-operation environment (inference servers, edge devices, etc.) and learning models for implementation are generated.

The following are some specific cases.

- Transformation of models appropriate for in-operation environment (computational performance)
  - Changes of calculation precision
  - Compression and speed-up of models (distillation, quantization, etc.)
- Optimization of models appropriate for in-operation environment (efficient inference)
  - Compiling of models (parallelization, vectorization, etc.).

The figure of a process model envisioned in the Guideline expects this postprocessing to be placed immediately after the iterative training phase and the quality check/assurance phase to be carried out under conditions close to the time of in-operation. However, postprocessing is often carried out as a part of the process of building the whole system after the quality check/assurance phase of machine learning components in actual development. In this case, the quality checked in the quality check/assurance phase may change (or deteriorate) at the time of in-operation, and it may become necessary to re-validate numerical equivalence in the test stage of the whole system.

4.2.1.4 Quality check/assurance phase

In the quality check/assurance phase, tests are designed in accordance with the specifications for ML requirements to carry out evaluations (tests) of learning models for implementation. This phase is explained below with reference to the flow of “quality check/assurance” shown in the bottom right of Figure 11.

4.2.1.4.1 Test design step

In the test design step, programs required for tests are created and test data is added in accordance with the specifications for ML requirements. Test data to be added may include cases where data is generated by means of pre-processing data augmentation and cases where trained learning models generates data which tend to make erroneous inferences.

4.2.1.4.2 Test step

In this step, the accuracy of learning models for implementation is evaluated based on regular indicators (accuracy, precision, recall, F values, etc.) using test datasets, and its stability is evaluated using neighboring data (data to which noise is added) of each data in datasets. In cases where any evaluation result does not satisfy the specifications for ML requirements, the process returns to the previous phase to adjust learning models and training datasets and modify the specifications for ML requirements.

System building / integration test phase

This stage is not part of the machine learning building, but it is described here due to the order of the procedures.

In an ideal development situation, all quality requirements for machine learning components are pre-arranged through the PoC trial phase, and all achievements can be confirmed up to the machine learning building phase.

It is hoped that no problems will occur at the subsequent stage of system building and integration testing.

In actual system development, complex real environment requirements analysis may not be perfect, unexpected interactions may occur with other components, and the effects of defects in other components may spread. Quality defects in machine learning components due to these various factors often occur during the so-called integration testing stage. Moreover, especially in a large-scale and complicated system, the prior data is inevitably insufficient as test data, and in many cases, inspection in the actual environment / system after integration is indispensable. In addition, it is conceivable to reproduce the inspection for rare cases that cannot be prepared in advance in the data composition phase at the stage of integration test.

In any of these cases, if a test fails, ML requirements analyses are partially modified and the whole machine learning model building phase is repeated again. In order to avoid this enormous amount of repeated work, it is desirable to accurately record the content of quality management activities conducted in each phase of each stage of machine learning building so that the effect of modifications can be grasped correctly and partial modifications are made without fail.

4.3 Quality monitoring / operation phase

After the system is deployed in actual operation, the activity to maintain the performance by continuously monitoring the quality at the operation stage and making necessary corrections is clearly positioned as the "quality monitoring / operation phase". Although this phase is placed outside the development phase in a more restricted sense in the waterfall V-shape development model, it already exists in the concept of system lifecycle such as the RAMS standards.

We consider that, in many cases, the quality needs to be managed during operation, from the viewpoint of 1) When qualitative requirements analysis based on prior data does not capture the actual environment and 2) When an environment at the time of operation is different from the environment when prior data was prepared, in machine learning based systems.

In various applications of machine learning, there are various patterns to update trained machine learning models in actual operation, because they cannot be classified uniformly. Therefore, the Guideline classify such patterns into two categories.

1) Pattern of carrying out re-training of machine learning in development environment and deploying it in operation environment after conducting tests.
2) Pattern of carrying out additional learning automatically in operation environment so that the system is self-adapted.

Section 6.8 summarizes the principles for these patterns individually.

5 How to apply this guideline

5.1 Basic application process

In this section, we overview the process of applying the Guideline.

Although this section focuses on the build-out of machine learning components, it includes processes such as analysis of whole systems which is deemed to have been carried out from the past in the scope necessary for explanation. Therefore, in cases where a specific development process has been built for the adaption to international standards and a person in charge of development can explain its adequacy, modifications (addition of any necessary stage listed in this section based on that existing process) may be made.

Identification of functions in charge in machine learning component system

This paragraph can be skipped in cases where the functions which machine learning components should fulfill within the system such as safety are fully identified in conventional system development processes (e.g. IEC 61508).

In cases where qualities in use of overall machine learning systems have been unidentified, qualities in use and external qualities of machine learning components are identified through the following process.

5.1.1.1 Prior examination on safety functions/check of applicable standards

- Whether a machine learning based system requires safety functions over a certain level (roughly higher than SIL 1/non-negligible personal injury is envisioned) is examined in advance.
- In cases where safety functions are required, applicable functions should be selected from IEC 61508 or standards of each application field and follow that process.

5.1.1.2 Identification of system function requirements (purpose/goals)

- The purpose of using the system, the scope of envisioned use environment and KPI to be achieved are identified at an overview level.

5.1.1.3 Examination on risk scenarios related to system use

- Examine a possibility that the system's functional requirements are not achieved or the system becomes incompliant to the purpose of use or social requests and list damages that may be caused in that case and other disadvantages. Respond to risk assessments.

5.1.1.4 Examination of requirements for characteristics of qualities in use of overall system

- Examine the quality required when the system is in use using quality metrics of any system.
  - As an example of means when there is no appropriate option, the three axes of external quality characteristics listed in Chapter 3 shall be applied to qualities in use to judge which disadvantage examined in the previous section is applicable. Then, the degree of severity thereof is judged based on the standards mentioned in each section of Chapter 3 to correspond them to the quality levels.
  - The maximum level should be identified for each of the three axes, and they shall be the level of qualities in use which the system should achieve.
- However, in cases where existing safety standards are adopted in Section 5.1.1.1, a risk avoidance level in relation to physical damages shall be decided based on quality indicators (functional safety, etc.) applicable to the existing standards.

5.1.1.5 Identification of components that contribute to the achievement of system component design / functional requirements and quality characteristics during use

- A combination of system components is designed to differentiate functions achieved by machine components and conventional software from those achieved by machine learning components.
- Moreover, it is analyzed which component of system the achievement of the characteristics of qualities in use depends on.

Identification of required level for achieving external qualities of machine learning components

- A level of achievement of external qualities is identified by means of the following procedures. At this time, machine learning components should specifically contribute to qualities in use.
- As regards risk avoidance, when any conventional standard for functional safety is applied to risks of physical or human damages, an analysis based on the conventional standard has priority, and a functional safety level to be achieved by machine learning components as software based thereon shall be replaced with a level of achievement required for risk avoidance of machine learning components.
- Other cases are defined as follows:
  - Basically, a characteristic level of qualities in use required for the whole system shall be a level of achievement of external qualities required for machine

learning components.

- In cases where undesirable output from machine learning components is monitored and corrected (modification by overwriting the output) by means of software components (Figure 8) processed serially or parallelly with machine learning components and sufficient quality is judged to be ensured for said software components by means of any conventional method, a quality characteristic level required for machine learning components is one level lower than the whole system level.
- When machine learning components do not contribute to or interfere with the achievement of the quality characteristic level required for the whole system at all, no level of achievement of said quality characteristic should be set for machine learning components (Level 0).

Identification of level required for internal qualities of machine learning components

– For each item of the internal quality characteristics listed in sections of Chapter 6, a level required for each characteristic is deducted from the level of achievement required for the determined external quality characteristics described in the preceding paragraph in accordance with the relationship set in Chapter 6.

Realization of internal qualities of machine learning components

– A method of realizing the level required for the internal quality characteristics mentioned in the preceding paragraph is examined in accordance with each sections of Chapter 7.
– The realization of each characteristic is assured by means of the technologies and processes listed in each paragraph and other technologies deemed to be equivalent thereto.

5.2 (Informative) Entrusting AI developments

The content of this section is informative.

The work explained in the previous section sometimes cannot be covered within one organization, thus requiring a development entrust. This section mentions points development entrusters and development entrustees should pay attention to when they work together, especially in dealing with AI specific aspects. Regarding the contract issues around the ownership of IP and sharing of compensation for damage, please see Section 8.1.1 “Contract Guidelines on AI” by Ministry of Economy, Trade and Industry.

Exploratory approach

When a development entrust starts for some machine learning component, it is often difficult for entruster to clearly provide KPI and acceptance criteria to entrustee. Therefore, an exploratory approach is required. The difficulties of this approach is similar to the problems we encounter when applying so-called agile development to the product development process that requires strict cost management and quality. Thus, “enterprise agile practices” are expected to be beneficial.

For example, the PoC phase described in Section 4 can be viewed as “Inception” phase as described in DA (Disciplined Agile[13]). In this phase, the development entruster and the development entrustee should build a consensus on the purpose of the PoC phase of AI (PoC exit criteria) and deliverables to the next phase. The deliverable list (e.g., decision on test strategy and basic architecture) recommended in “Inception” phase in DA can be utilized for this activity. . It is highly important to set goals for the PoC stage (stage gate). To prevent any missing out important points, and to appropriately promote subsequent phases, the following two aspects should be covered in the PoC phase. i.e., “requirement clarification” and “feasibility study” (see Figure 13). For both activities, two perspectives should be taken care: “Terminate condition of PoC” and “PoC outcome to be provided to the entruster”.

[13] A summary of best practices for businesses to adopt agile development. It has been used broadly since 2019 when it was acquired by PMI (Project Management Institute).

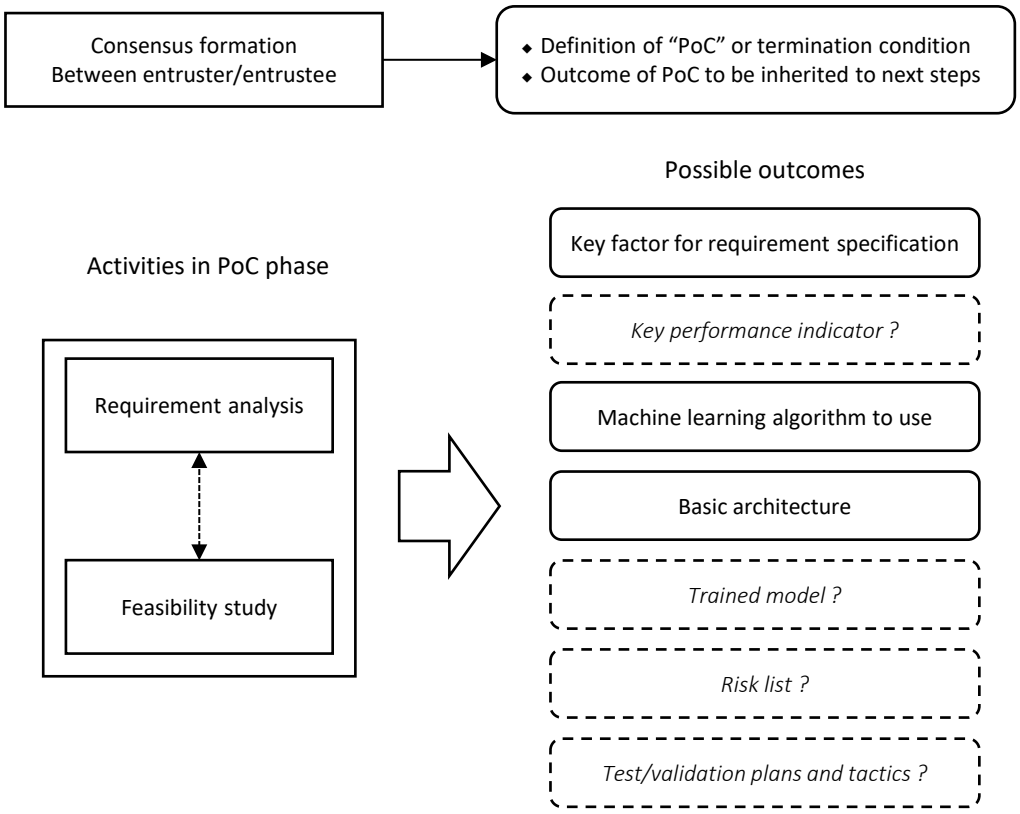

Figure 13: Activities and deliverables in the PoC phase

Role clarification of entrusters and entrustees

The clarification of roles to be performed by development entrusters and development entrustees, for each work in the process described in Section 5.1, will protect the both side from risks and prevent overlook of necessary works, leading to a quality assurance of machine learning components.

One example of developing tailor-made AI from scratch is shown below. Terms XXX, YYY, etc. are to be filled in appropriately.

Table 3: Example division of roles

Step	**Task for development entruster**	**Task for development entrustee**
1. safety standards	To examine primarily checking applicability of existing standards	To confirm the examined result
2. Functional requirements of the system	To describe/provide in natural language document,	To review provided document to clarify the purpose and

	based on the requirements of XXX. Qualitative expressions of qualities in use to be included.	goals.
3. Risk scenario	To prepare risk list for some similar products YYY Presentation of constraints	To analyze based on some established method of ZZZ, Update where necessary.
4. Qualities in use / external qualities	To present priorities and constraints between quality aspects (e.g., quality characteristics indispensable for the given product)	To assert quality levels to be targeted. If applicable, use three external quality characteristics.
5. Design of system components	To review and confirm	To perform system design. For reviews by the entruster, provide materials AAA.
6. Levels for external quality aspects	To review and confirm	To identify and propose target levels
7. Internal qualities	To review and confirm	To examine and propose the levels of target quality aspects and methods for achieving these.
8. Realization of internal qualities	To provide dataset sources. To review test reports	To perform quality enhance methods examined above and report.

The details in the table would differ depending on the project, and the contents once decided may change, when the work is repeated.

(Example 1)

In some cases, no specific KPI nor quality target can be presented by entruster, and “using particular training dataset” may be the only clear request as a practical measure (the condition defined in the contract). In this case, additional training data is likely to be required as a result of conducting an analysis on internal quality related to data described in Section 6. It is desirable for the both sides to recognize such a risk and reach an agreement on necessary arrangements (their roles) in advance, considering Point 7 in the above table.

(Example 2)

When the development entruster "lacks understanding" or "has ambiguity" about the quality to be achieved by the machine learning components in the early stages, such activities as "Presentation of priority and constraints" in the above table can be hardly made. In this case, the development entrustee side may present an initial idea as to qualities in use (in a bottom-up manner taking the outline of system design into consideration) in the first PoC phase. In the subsequent main development phase, the roles are to be updated as the entruster vision becomes clearer.

In the case of the PoC phase, the content of each step in Table 3 may change, but entire step is not unnecessary. On the other hand, in the main development phase, the loop within Step 8 is usually sufficient, if the goals for the stage gate (PoC exit criteria) have been set and cleared appropriately.

However, depending on the projects, the goals for the PoC stage gate must be dropped or compromised. This means that there may be no clear separation between the PoC phase and the main development phase, leading to uncertainty of the outlook for quality management. In this case, the development entruster shall take the risk, and it is usually left to the development entruster's judgment, whether to accept it and what to do with the work division for mitigating the risk.

Notes on determining the detailed roles

We explain here the results of extracting and examining the items to be noted from the perspective of AI with reference to "Guide to Quality Assurance in Connected World[14]" of IPA/SEC, to concretize works and make arrangements in the development lifecycle as described in the previous section.

1) System and organization that can fulfill accountability

In AI development, quality evidence are often based on the process perspective only. It is so important to clarify the definition including "who can approve it". Also, we should consider underlying risks in the process.

Example: Regarding the training dataset preparation, it is not enough to describe the preparation of raw data provided by entruster We should care about , who will perform preprocessing (e.g., data cleansing) and who will (and how) confirm that those preparation has been appropriately achieved.

It is important to consider and specify "compromises and logics the both sides can agree on" in advance in the face of time and cost constraints.

[14] https://www.ipa.go.jp/sec/publish/tn18-001.html

2) Test

Even if the training of models is completely up to the entrustee side, the entruster usually must have deep understanding of the content in the following phase; i.e., “quality check/assurance”. For that purpose, it is desirable to not only define evidence but also make arrangements as to the method and scope of tests, and what to give up in terms of effectiveness and efficiency, so that the both sides feel convincing as much as possible when the contract made.

It is important that both sides cooperate for the purpose of “final quality improvement”. For example, any act just focusing on clearance of verification, (such as using test datasets presented by entruster for training), must be strictly avoided.

3) Quality management during operation

Regardless of immediacy of model update, continuous quality management is indispensable for the machine learning component even after its release, as described in 4.3 “Quality monitoring / operation phase”. Therefore, it is necessary to include the design and implementation of optimal quality management methods during operation in accordance with system functional requirements in the scope of works.

For example, both sides may discuss on identification of “data to be obtained for performance evaluation” and “environmental data which cannot be grasped completely during development”, and the system implementation for obtaining and saving those data during operation.

5.3 (informative) Notes on delta development

Existing software components are often recycled not only in machine learning based systems but also in systems and services which use software components. In machine learning, additional learning is applied to certain data to customize it based on trained models. It is complicated to guarantee the quality of recycled products.

As a fundamental principle, in order to ensure both conventional functional safety and the quality of machine learning covered in the Guideline, the guidelines for assuring quality in situations where a new system is used (context) is consistent from the analysis of conditions in use through the definition and implementation of functional requirements to tests with regard to the quality of systems which use recycled software components. There are, for example, the following methods to realize this consistency, any of which should be chosen in accordance with a quality level required by the system and the state of components provided.

1. A new quality management process is established in the context of new system including detailed checks of datasets in which recycled machine learning components (older datasets) are installed, and quality management is carried out independently from older components.

2. Quality management activities equivalent to those described in the Guideline are carried out with respect to recycled machine learning components. When a level of quality management higher than the requirement of the new system is carried out, especially when we can clearly confirm that an envisioned condition in use is a subset of envisioned situations of older components (including the case where they are the same) with regard to “sufficiency of problem domain analysis”, it is necessary to check if the quality is not deteriorating due to additional learning as needed with reference to records of original quality management.

   It is difficult to apply this method if said components are not conscious of quality from the beginning, since quality management of older components should have been carried out and recorded sufficiently. Moreover, because the analysis of conditions in use implicitly assumes certain context of use, it may be difficult to judge comprehensiveness.

   Furthermore, a developer that provides machine learning components as general-purpose parts can improve reusability by completing quality management activities in advance in anticipation of this type of recycling and by clarifying the envisioned quality levels.
3. When a relatively low-quality level is required for application, quality management activities with a focus on the test phase should be examined on the assumption that existing datasets are considered as an unknown black box. For example,
   – For 6.1 “Sufficiency of requirement analysis” and 6.2 “Coverage for distinguished problem cases”, make a fresh analysis for a new system to set a new quality targets.
   – For 6.3 “Coverage of datasets” and 6.4 “Uniformity of datasets”, prepare new test datasets based on new result of requirement analysis, and check achievements of these characteristics to a certain degree during the test phase.
   – For 6.5 “Correctness of trained model” and 6.6 “Stability of trained model”, If additional learning is carried out, make an evaluation in the validation/test phase using performance indicators obtained during additional learning, and using training datasets added based on new requirement analysis. When additional learning is not carried out, make a validation based on new result of requirement analysis in the test phase, or to perform evaluation during integration tests on the whole system.

   However, the current descriptions of Chapter 6 and Chapter 7 reveal that it is difficult to carry out sufficient quality management activities by using only sampling-type tests of training results without analyzing datasets used for training. It is practical at this stage to obtain detailed information on older datasets and combine it with a white box approach described in the previous paragraph 2.

6 Requirements for quality assurance

This chapter sets the following eight internal qualities as the quality management characteristics to manage quality of machine learning components specifically for the purpose of the achievement of the two qualities in use, “safety” and “AI performance”. Section 9.1 explains the analyses leading to the setting of these internal qualities. We will further analyze the other quality in use “fairness” and add internal quality characteristics in the future.

6.1 Sufficiency of requirement analysis

General

“Sufficiency of requirements analysis” means that usages of a target machine learning based system are analyzed sufficiently and every requirements for the system is captured (described in Section 1.7.1).
Requirements analysis is important especially in the early stage of development of a conventional software which is used for a safety application. The main purpose of requirements analysis for the development of machine learning based systems in this guideline are as follows.

① Sufficient identification of cases in which risk management is needed, mainly in applications required “safety”.
② Sufficient identification of attributes of objects for which inequality is not allowed, mainly in applications required “Fairness”.
③ Sufficient analysis of real world in order to verify that training datasets and test datasets are comprehensive and appropriate extractions of the real world, to all requirements including “AI performance”.

“Sufficiency of requirement analysis” is always required not only for machine learning based systems but also equipment and services controlled by software.

However, if any situation where machine learning based systems are used is overlooked at this stage, there are few opportunities for taking note of an error from the data collection stage to the actual training/test stages, so that it is likely to cause malfunction that occurs for the first time in the stage of final system test in real environment or the actual operation stage.

On the other hand, if the details of all possible situations where machine learning based systems are used are analyzed including minute differences, the analysis results can be implemented as a regular software component, and there is no merit in using a machine learning technology.

To put it another way, it is impossible to make comprehensive and thorough analyses in advance for such applied systems. That is why there is demand for having systems acquire

knowledge through machine learning in any way.

From these two viewpoints, it is extremely important to appropriately set “levels of detail of requirements analyses” in machine learning based systems in terms of both quality assurance and feasibility and efficiency of implementation.

They correspond to the judgments of “what do humans need to analyze” and “what should we have machine learning acquire”. The maintenance of appropriate balance between these two viewpoints is an important goal of requirements analysis in quality management of machine learning.

Approaches

The two goals of this internal quality aspects described in this paragraph are;

- Clarifying what is required for machine learning components; and
- Clarifying the limited scope which machine learning components have to deal with.

6.1.2.1 Extracting and listing attributes (characteristic viewpoints) and attribute values (specific features)

At first, inputs in real world which can be extracted as specific characteristics are organized from the aspect listed below. Each identified viewpoint will be referred to as “attribute”, while the specific features belonging to the viewpoint as “an attribute value”.

The following examples can be given as attributes and their viewpoints.

1. Attributes that characterize the differences in output required for machine learning components as functional requirements.
   - In supervised machine learning, “supervision labels” are different.
   - For example, in the recognition of characters (numbers), 10 distinctions from 0 to 9. In anomaly detection, whether there is any abnormality is detected.
2. Attributes whose output is the same but require machine learning components to explicitly respond at a functional specification level.
   - One example is a case where, in character recognition, the specifications are set that both “l and ℓ” and “l and /” should be recognized correctly.
   - Another example is that, when the specification says that various kinds of obstructions such as “pedestrians”, “bicycles” and “automobiles with crossing movements” are to be avoided in automated driving, each kind of object should be identified as a separate attribute value. Or, “gender” and “age groups” in the scoring for recruitment that requires fairness apply to this paragraph.
3. Attributes for which deterioration risks of performance expected for machine learning components are likely to be different significantly in real operation

- Some examples include “climate”, “distinction of day and night”, “backlight and front light” and “movement speed” in the case of outdoor object recognition.

4. Elements whose expected output is the same due to the characteristics of machine learning but it is difficult to find a common ground by the model after learning or they can be recognized as different internally.
   - For example, in the case of outdoor object recognition, different characteristics of objects may be captured in the daytime and nighttime, or another example is the distinction of the horizontal and vertical installation of traffic lights.
   - Another example is that different forms of handwritten numbers such as “1 and /” are categorized into several characteristics which differ greatly depending on countries of origin. In order to recognize all different types, it is necessary to intentionally include them at least in training datasets and test datasets as different general populations, even if they are not clarified in the functional specifications.
5. Characteristics which make it difficult to specify the method of evenly collecting learning data as a process when data is collected.
6. Differences which humans can easily capture, though there are no differences as described above; e.g. Kinds of animals, skin color, etc.
7. Object which humans cannot recognize or explain their characteristics with word, but it is possible to extract a sufficient number of samples without bias. For example, it is difficult to analyze all possible ways of describing the number (8) in advance. However, if people do not intentionally write various different styles of the number “8” and we have a sufficient number of randomly-extracted samples, they are expected to be unbiased samples.
8. Cases where it is extremely easy to mechanically cover and complement differences. For example, once required specifications are established, it is possible to mechanically generate samples of parallel shift of images, color changes and expansion of rotations within a certain range, and intentionally synthesize training data and test data.

We make up a list for candidates of possible attributes from requirements analyzed on the overall system, and then we should determine whether each attribute it either picked up for explicit target of management by the developers, or is “left to machine learning” without picked up. To determine that, the level of risks related to values of each attribute and the type of the application using machine learning (results of PoC trials may be useful as a reference) should be considered. In general, the characteristics shown earlier in the above list (eg 1～3) are highly likely to be extracted as attributes. In contrast, it is often reasonable to leave the characteristics like listed lower (in 7～8) for machine learning without extracting them as attributes. However, the diversity of data types corresponding to attributes left to machine learning will be examined in the third internal quality “Coverage of datasets”, so that it is necessary to take this point into consideration in the analysis, too.

6.1.2.2 Combinations of attributes that should be excluded

"Impossible combinations of attribute values" should also be examined. The purpose of examination is to distinguish cases which should be excluded as functional requirements such as "snow in summer (in Tokyo)" from cases which should be handled as functional requirements such as "snow coverage in winter" which occurs very rarely. It is important to make this distinction, because it will become impossible to determine whether lack of data is acceptable and to explain "the quality of quality management method itself" in subsequent quality management.

To be more specific, after attributes and attribute values corresponding thereto are listed, impossible combinations of attribute values are selected based on system requirements.

Requirements for quality levels

For "sufficiency of requirement analysis", it is required to handle the following three levels in accordance with the level of external quality.

The correspondence between external qualities and internal qualities regarding the required level is as follows.

- "Safety"
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: More than Lv 3 (some requirements will be added to Lv 3)
- "AI performance"
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- "Fairness"
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each level of internal qualities are as follows.

- Lv 1
  - Examine and record the major cause of possible deterioration of quality.
  - Based on the examination results, design data and reflect it in necessary attributes.
- Lv 2
  - Analyze risks of deterioration of quality in use in overall system and their impact with a certain level of engineering coverage and record the results in documents.
  - Analyze if any measure is required for each of those risks, and analyze attributes

related to the risk which are contained in an input to machine learning components.

- Analyze and record the application-specific characteristics of environments which will generate machine learning input, with regards to the difficulty for machine learning and other aspects.
- Examine sets of attributes and attribute values, based on the results of those analysis and record the background of such decisions.

– Lv 3
  - The following activities are carried out in addition to those listed in Lv 2.
  - Investigate documents on own past examination results and those of others with regard to elements to be extracted as characteristics of system environment and record the background of examinations leading to the extraction of necessary subsets.
  - Investigate past examination results in line with application fields of systems with regard to deterioration risks of qualities in use of overall systems and record the examination results including the background of selection.
  - Moreover, extract deterioration risks of qualities in use of overall systems using engineering analysis such as Fault Tree Analysis and record their results.

6.2 Coverage for distinguished problem cases

General

On the basis of the sufficiency of requirements analysis described in the previous paragraph, it requires careful consideration of data design as “Coverage for distinguished problem cases” in order to secure sufficient training data and test data with respect to various situations systems need to respond to. More specifically, the number and details of combinations of attribute values focused in the stage from training data preparation to testing process is considered at this stage.

In an ideal situation, for example, if there are sufficient data corresponding to all combinations of attribute values (direct products of attributes) for combinations of attributes that are supposed sufficient as described in the previous paragraph, it can be said that it covers all possible situations in the real world. However, the number of attributes may reach 10 or more in actual system development so that the number of combinations of attribute values may often reach about 10,000–one million. In this case, it is crucial to consider appropriate “coverage” and “design” for quality management in this paragraph.

It is important to focus on two viewpoints in actual quality management. First, combinations of attributes that may cause malfunction or misjudgment need to be reliability addressed at the training or testing stage. Second, at the same time, it is necessary to cover as much as possible all situations which the machine learning based system to be implemented

may encounter during operation from the viewpoint of both the quality of training and the quality of deliverables.

Such a problem has been addressed as a task of “test design” in developing conventional software. However, in machine learning where not only the testing process but also the implementation process is performed based on datasets, it is unique that the same task is required in the implementation process.

Some practical solutions to the excessive number of combinations for test design have been already presented in conventional software engineering. This guideline aims to provide practical and sufficient training and quality tests by applying such existing knowledge to machine learning applications.

Approaches

Firstly, if the number of attributes to be recognized is very small and the total number of all attributes (the impossible cases described in the previous paragraph are deducted) is only 10–20, their combinations are named as “cases”, and they are checked if all cases are included in test datasets and training datasets in later stages.

On the other hand, If there are too many combinations when the total of those attributes are used or sufficient data cannot be acquired especially in rare cases, combinations of some attribute values should be extracted under certain standards and set as “cases” to cover all those combinations. For example, in the example of traffic lights listed in Example 1 in Section 1.7.1, such combinations as “daytime green”, “daytime yellow”, “nighttime red”, “daytime rainfall”, “nighttime rainfall” and “red signal in rain” are extracted and called as “cases” to make sure that data applicable to these combinations is included in test datasets. Even if all attributes cannot be covered completely, it is possible to intend to achieve a certain degree of “coverage” by avoiding cases where high-risk situations such as “nighttime rainfall” are not included at all in datasets or eliminating cases where “red signal” is not included in training at all. When considering such “simplified coverage”, the data of “red signal and daytime rainfall” are included in the several cases such as “daytime rainfall”, “daytime red signal” and “red signal and rainfall” at the same time.

Furthermore, completeness should be guaranteed more specifically in applications that require high quality by introducing mathematical “standards for coverage” to such extraction works. In the case of black box tests in software engineering, methods such as the sampling rate of random sampling, the orthogonal table and pair-wise test and the “standards for coverage” based on them are known, we should select appropriate means for each application.

Requirements for quality levels

The handling of the following three levels is required for coverage of distinguished problem

cases in accordance with a level of each external quality.

The correspondence between external qualities level and the required level of this internal qualities are as follows:

- "Safety"
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2～4: Requirements to be added to Lv 3 should be examined.
- "AI performance"
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- "Fairness"
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each required level for the above internal quality are as follows.

- Lv 1
  - Set cases for each of attributes corresponding to major risk factors.
  - Moreover, set cases corresponding to combinations of composite risk factors.
  - Furthermore, extract attributes of differences in particularly-important environmental factors and prepare cases corresponding to combinations with serious risk factors.
- Lv 2
  - Satisfy all requirements listed in Lv 1.
  - Particularly-important risk factors should satisfy, in principle, the standards for pair-wise coverage. To be more specific, a case of combining "an attribute value of combination of those factors" and "individual attribute values included in all attributes other than those to which the attribute value belongs" should be included.
- Lv 3
  - Based on engineering consideration, set standards for coverage of attributes and establish sets of combinations of attribute values that satisfied standards for coverage as cases.
  - The level of strictness of the standards for coverage (pair-wise coverage, triple-wise coverage, etc.) should be set taking into account system usage and risk severity. Standards can be set individually for each risk where necessary.

6.3 Coverage of datasets

General

The term “coverage of datasets” means that a sufficient amount of data is given to each cases covered by establishing the standards as described in the previous paragraph without omissions for the input possibilities corresponding to each cases.

When conventional software is developed, the details of all characteristics in real world which software operations depend on are captured at any stage from the machine learning requirements analysis phase to the implementation phase and reflected in software components in the form of conditional branching or calculation formula. On the other hand, when a machine learning component is built, differences in details exceeding a certain level are not captured as attributes of ML requirements analysis or “labels” at the time of training but are reflected in ultimate operations through the training phase of machine learning as a distribution of datasets used for learning, as described in Section 6.1. The purpose of establishing this characteristic axis is to guarantee that no inappropriate learning behavior occurs due to lack of data with regard to characteristics of those unidentified details as “attribute values”.

Approaches

The main purposes of this quality aspects are to achieve sufficient level of “quantity” and “coverage over input situation”. However, since there are characteristics whose minor differences have not been identified as attributes, the latter coverage often has no choice but to depend much on the process of collecting and processing data. On the other hand, when the standards for coverage are introduced in Section 6.2, it would be possible to test if attributes which “are not included in cases” are distributed without bias.

Moreover, as regard the former quantity, it is important to appropriately define granularity of how to set the cases described in the previous section, but it is possible that sufficient data for specific cases cannot be obtained, for example, in rare cases where the frequency of occurrence is low. In those cases, it may become necessary to take measures in the whole development process for evaluating the response status by means of tests on the whole system by partially abandoning “coverage of datasets” for cases that lack specific data and after covering looser coverage standard.

Requirements for quality levels

The correspondence between external qualities level and the required level of this internal qualities are as follows:

- "Safety"
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2～4: Requirements to be added to Lv 3 should be examined.
- "AI performance"
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- "Fairness"
  - AIFL 1: Lv 2 or above
  - AIFL 2: Lv 3
    (note: We consider that this quality characteristic is important in dealing with data bias, which is an important factor that impairs fairness.)

The requirements for each required level for the above internal quality are as follows.

- Lv 1
  - Consider the source and method of acquiring test datasets to ensure that no bias is found in application situations.
  - Extract samples without bias from original data for each case to ensure that no bias is found.
  - Record activities carried out to prevent bias from entering.
  - Check that there are sufficient training data and test data for each analyzed case in the training phase, validation phase, and so on.
  - When sufficient training data cannot be acquired for any case, review and loose the coverage standards and record what should be checked individually by system integration tests in line with the original standards.
- Lv 2
  - The following activities are carried out in addition to those listed in Lv 1.
  - Grasp an approximate probability of occurrence for each attribute value or each case.
  - Check if acquired data is not deviated from the distribution.
  - Positive check other than acquisition methods should be made regarding the coverage of the data included in each case.
    - For example, in each case, when there is any attribute not included in that case, extract the distribution related to attribute and check if there is no significant bias.
- Lv 3
  - Acquire certain indicators for coverage of data included in each case in addition to those listed in Lv 2.
    - For example, check if there is no correlation between data other than

attribute values included in combinations of cases using feature extraction or any other technique.

- ✧ Or consider an expected distribution of attributes not included in each case, and analyze and record differences.

6.4 Uniformity of datasets

General

Evaluating only the “coverage of each case” in the previous paragraph does not always mean that all datasets are good sampling of the overall environment expressed by input data. When the probability of occurrence differs significantly from case to case, simply preparing samples for each case will generate datasets with a large bias as a whole, which may significantly impair performance, especially in terms of AI performance. On the other hand, when performance is required for rare cases that probability of occurrence is very low, it cannot be generally coexist the preparation of the practical amount of uniform data without bias for all input and the preparation of the sufficient amount of data for rare cases. For example, when 100 training data are required for an event that occurs at a frequency of one millionth, it is not usually acceptable that the number of cases of all unbiased data is 100 million. From this viewpoint, it is considered that the uniformity in this paragraph needs to make an appropriate compromise in some cases, contrary to the coverage in the previous paragraph.

In general, it is required to prepare sufficient training data for combinations of attribute values with risks which should be avoided by making correct judgments when risk avoidance is strongly sought. When such a risk occurs on rare occasions, an enormous amount of data might be required for training all other cases with a sufficient “amount of data”. In such case, it is quite conceivable to "focus" on training of particularly-rare risky cases.

On the other hand, when overall performance (AI performance) is required, by focusing on training rare cases more than the actual probability of occurrence, the inference accuracy deteriorates in other cases, and it may also worsen average performance. In such case, it is not always appropriate to deeply explore the coverage of detailed cases in the previous section.

Moreover, when fairness is strongly required, it may change whether the training should be artificially equivalent between cases or should be randomly trained according to the distribution of extracted training datasets, depending on a type of fairness required.

Approaches

The basic concept itself is to focus on cases of “overall” in the topic of coverage in Section 6.3.2. While taking care not to bias in the process of acquiring whole datasets, it is necessary to monitor the frequency of occurrence of each attribute value as appropriate.

Rather, as stated in general, in this section, it is important to consider how to coexist with

the coverage in the previous section and data design.

Requirements for quality levels

The correspondence between external qualities level and the required level of this internal qualities are as follows:

- "Safety"
  - AISL 0.1: Lv S1 or above
  - AISL 0.2/1: Lv S2 or above
  - AISL 2～4: Requirements to be added to Lv 2 will be examined.
- "AI performance"
  - AIPL 1: Lv E1 or above
  - AIPL 2: Lv E2 or above
- "Fairness
  - AIFL 1/2: Lv E2 or above

The requirements for each required level for the above internal quality are as follows.

- Lv E1
  - Same as Lv 1 in "Coverage of datasets" in the previous section.
- Lv E2
  - Same as Lv 2 in "Coverage of datasets" in the previous section. However, assumed probabilities of occurrence are compared with the whole sets of assumed events.
- Lv S1
  - Regarding the amount of data for each case considered in L1 of the previous section, explicitly check if there is a sufficient amount of data for risk cases.
  - When data of rare cases is insufficient for training, comparing the amount of the whole sets of training data with a probability of occurrence of rare cases, consider focusing on learning of rare cases. However, especially when Lv E2 is required, with prioritized, the impact of reduced training of other cases on whole system quality should be considered.
- Lv S2
  - In addition to what is listed in Lv S1, estimate and design in advance the amount of data of each case, based on the estimated probability of occurrence for each risk event / case.

6.5 Correctness of the trained model

General

The term “correctness of a trained model” represents that a machine learning component functions as expected upon the input from the learning dataset (consisting of training data, test data, and validation data). In this guideline, this notion also includes the convergence of the training and the quality of training data (e.g., the dataset has only a small number of outliers and incorrectly labeled data).

Approaches

The correctness of a trained model is usually evaluated in terms of quantitative measures, such as accuracy, precision, recall, or F-value. These measures may overfit to the training dataset, that is, they may perform well for the training data, but badly for the data that are not included in the training dataset. Therefore, in the test phase, the correctness needs to be evaluated using a test dataset that is disjoint from the training dataset, or more generally, by applying cross validation.

Data scientists should select concrete techniques suited to the application and need to explain their selection. In Section 7.5 we will explain some examples of techniques that can be used to test the correctness of a trained model.

Requirements for quality levels

The relationships between these internal qualities and the required levels for external qualities are as follows:

- “Safety”
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 will be examined in future.
- “AI performance”
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- “Fairness”
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

The requirements for each quality level for the above internal quality are as follows.

- Lv 1
  - Prepare a test dataset by taking into account the coverage of data and the past experiences, e.g., obtained in the proof of concept (POC) stage.
  - Decide and explain how to ensure the quality of the test dataset, such as excluding outliers and modifying incorrect labels.
  - Prepare a training dataset in an analogous way to the test dataset. Note that the training dataset may not necessarily follow the same distribution as the test dataset.
  - Decide and record how to deal with the incorrect behavior of a trained model (e.g., false negative/false positive in the test) before the validation phase.
  - If a machine learning component is required to satisfy fairness, decide and record the methods and criteria to evaluate fairness before the validation phase.
- Lv 2
  - All the requirements listed in Lv 1.
  - Provide certain evidence to show the correctness of labels in the training, validation, and test datasets.
  - Decide and explain methods and criteria to validate the trained model (e.g., accuracy and its threshold) before the validation phase.
  - Test the trained model using the given test dataset and additional test data (e.g., generated by data augmentation techniques).
  - If possible, analyze internal information on the trained model (e.g., the neuron coverage to evaluate the adequacy of testing).
- Lv 3
  - All the requirements listed in Lv 2.
  - Perform the validation/testing of the whole system (e.g., integration tests), especially by focusing on risky cases.

6.6 Stability of the trained model

General

The term “stability of a trained model” represents that given an input that is not included in the learning dataset, a machine learning component behaves in a similar way to when given a nearest training data as input. When the trained model is not robust enough, it may function incorrectly upon first seen input, and may violate the safety/security of the system. Therefore, evaluating stability is important when the system is required to satisfy safety/security. For example, insufficient stability may cause the following problems.

- The trained model can function incorrectly when the input is far from the training

data. Typically, this may be caused when the model is overfit to the training dataset.

– The trained model can behave significantly differently when a small amount of noise is added to the input to the model. Such input noise can be either random noise in nature (e.g. dirty camera lens) or adversarial perturbation caused by malicious attacks. For example, those attacks are triggered by data poisoning in the learning dataset, and by certain tricks on physical objects (e.g. attachment of tiny stickers to road signs).

Approaches

Stability can be evaluated/improved mainly in the following three phases in the machine learning lifecycle. Some useful techniques will be explained in Section 7.5.2.

– Training phase: A trained model's overfitting to the training dataset can be avoided e.g., by separating the validation dataset from the training dataset, by regularization techniques, by evaluating the impact of small input noise, and by monitoring the training process.
– Evaluation phase: Correctness measures (e.g., accuracy, precision, recall) can be evaluated by using synthetic data that are obtained by adding random/adversarial noise to test data.
– Operation phase: By monitoring the input to the trained model, some of adversarial input may be detected and eliminated.

Requirements for each quality level

It is required to record the methods applied to improve the stability of a trained model and the evaluation results of the stability. In the list below, neighboring data refers to data generated by adding small perturbation noise to original data. Examples of concrete techniques will be explained in Section 7.5.2.

The relationships between these internal qualities and the required levels for external qualities are explained as follows:

– "Safety"
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 will be examined in future.
– "AI performance"
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
– "Fairness"

- AIFL 1: Lv 1 or above
- AIFL 2: Lv 2 or above

The requirements for each required level for the above internal quality are as follows.

– Lv 1: Record the concrete techniques applied to improve the generalization performance of a trained model (e.g., cross validation and regularization are widely used to prevent overfitting to the training data).
– Lv 2: Record the evaluation results of stability by using neighboring data.
  - In the evaluation of stability at Lv2, it is required to use some synthetic data obtained by adding a small amount of noise to the learning datasets. In particular, it is recommended to apply techniques to prevent attacks based on adversarial examples; e.g., robustness evaluation using adversarial examples, adversarial training to train a robust model, and dynamic detection of adversarial examples. Currently, these new methods are still being studied and developed in academic research, but might be applied to system development more effectively in the future.
– Lv 3: Provide formal guarantee to the stability for neighboring data.
  - At Lv 3, it is required to formally guarantee a certain level of stability for neighboring data. For example, methods for certifying adversarial robustness have been studied recently and might be used in system development in the future.

6.7 Reliability of underlying software systems

General

The term “reliability of underlying software systems” represents that the underlying conventional software (e.g., training programs and prediction/inference programs) functions correctly. This notion includes the software quality requirements such as the correctness of algorithms, the time/memory resource constraints, and the software security. When electronic hardware such as MPU is developed, this notion also includes the hardware dependability.

In many applications of machine learning systems, open-source software is used to train machine learning models and to develop conventional software systems. Although the quality of open-source software may not be guaranteed, the developers of machine learning systems should be responsible for ensuring sufficient quality even when using open-source software.

Furthermore, the correctness of software should usually be tested under the same environment as actual operation environments. When the test environment is different from actual operation environments, it is necessary to evaluate the differences between them. In the development of machine learning systems, however, there are often significant differences between an environment used for training (e.g. computing environment with cloud and GPU)

and an actual operational environment (e.g. built-in computer). Even the behaviors of numerical calculations (e.g., the accuracy of floating-point computation) can change between them. In some cases, numerical processing itself may be changed e.g. by model compression. When implementing a machine learning system, the developer needs to cope with these environmental differences that may affect the AI performance of the system.

Approaches

Quality management methods for conventional software systems can be applied in order to ensure the dependability of the underlying software system.

As regard the use of open-source implementations, appropriate quality assurance measures such as those listed below should be taken to achieve the safety and reliability of the system.

Moreover, when the in-operation environment is different from the environment in the training process (e.g., when performing model compression), this Guideline recommends to test the software that reproduces the same calculations as the in-operation environment in the test phase. If this is difficult, the developer should test the whole system to check if the quality deterioration falls within the acceptable range.

Requirements for quality levels

- "Safety"
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2∼4: Requirements to be added to Lv 3 will be examined in future.
- "AI performance"
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- "Fairness"
  - AIFL 1: Lv 1 or above
  - AIFL 2: Lv 2 or above

- Lv 1
  - Select reliable software used for the machine learning system, and record the process of this selection.
  - Monitor the system's operation to check and update the selected software.
  - Examine in advance the impact of differences between the environment in the training/test phases and the environment in the actual operation phase.
- Lv 2

- Evaluate the reliability of the software used for the system by testing.
- If possible, use software whose reliability is SIL 1 or equivalent.
- Prepare for the maintenance of software during its operation.
- In the validation and test phases, conduct validation tests in an environment that simulates the environment used in the actual operation phase. Alternatively, validate the consistency of operations of the trained model between in the test phase and the actual operation phase.

– Lv 3
  - Check the quality of software for SIL 1 (or a higher SIL level when required by the system).
  - Perform testing or formal verification of the behaviors of the trained model in an actual environment.
  - Check the consistency of those models and operations in an actual environment in any stage after integration tests.

6.8 Maintainability of qualities in use

General

The term “maintainability of qualities in use” means that internal qualities satisfied at the beginning of operation are maintained throughout operation. This concept means that internal qualities can fully respond to changes in operational environments outside the system and that any change in trained machine learning models do not cause unnecessary deterioration of quality.

A specific method of realizing the maintainability of qualities in use depends largely on forms of operation, in particular, how to carry out additional learning and retraining. This guideline envision the following two patterns of additional learning and retraining.

(a) Cases where the results of additional learning are reflected in an operation environment for the first time after going through additional learning and quality tests outside (developmental environment) of actual systems in service (operational environment) through explicit updates of software. Solutions such as automatic driving envisioned now belong to this pattern.

(b) Cases where trained models are updated on a real-time basis during operation and updates complete without any human tests in an operational environment. Online learning used to process streaming data, language recognition and application of conversations in projects in which development and operation are integrated belong to this pattern.

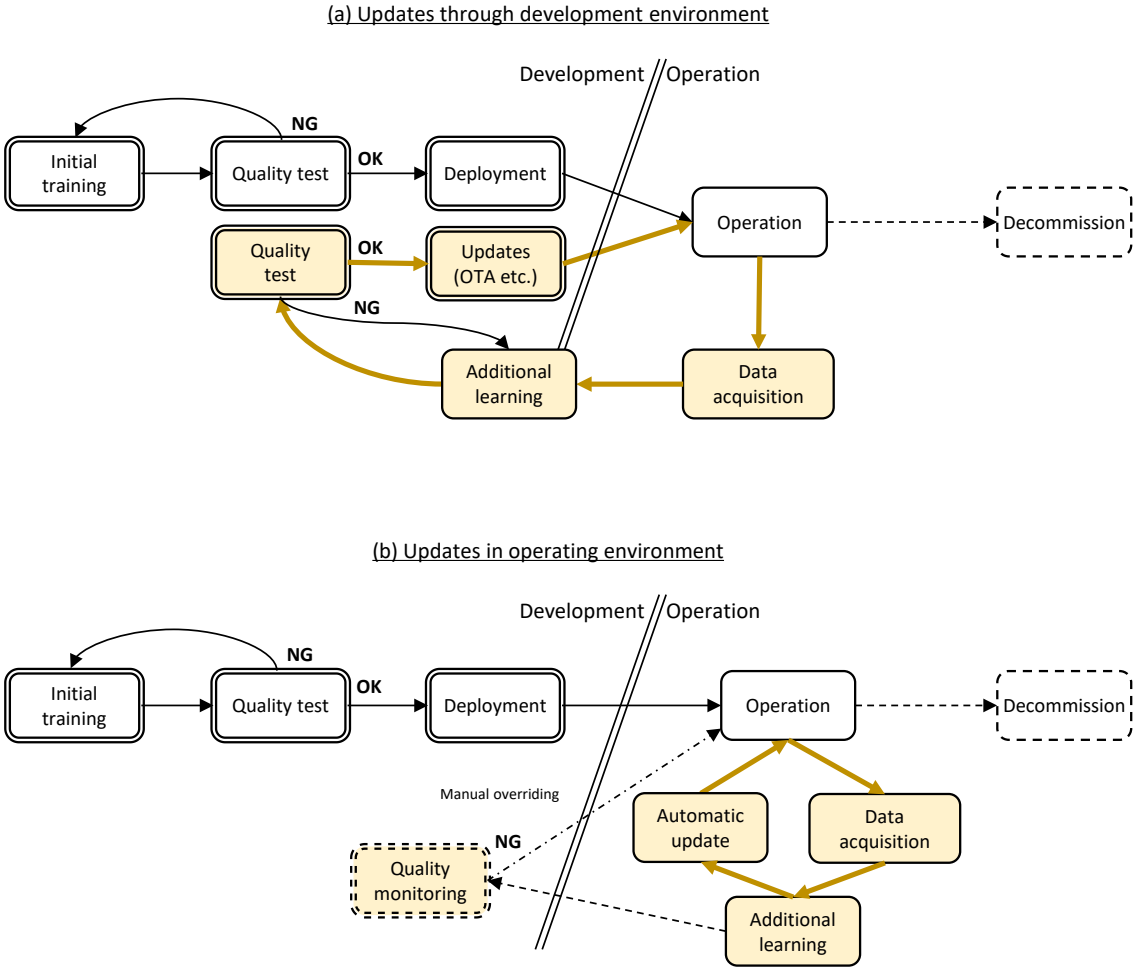

Figure 14: Forms of updates of machine learning components in operation

As described in the above figure, quality tests are always carried out before updates in the update pattern in the development environment described in (a). If the content of tests is the same as those conducted in the test phase of initial development, a certain level of quality can be maintained. On the other hand, in the update pattern of the operational environment described in (b), there is a higher risk that a model whose quality has been deteriorated is reflected in operation, because updates are fully automatic. In this case, it is important to incorporate a quality monitoring mechanism and a mechanism dealing with deterioration in an operational system.

Moreover, as regards maintainability of qualities in use, in addition to the deterioration of overall performance, it may be necessary to give attention to a problem that a machine learning component makes a misjudgment on specific input after its update[15], although it used

[15] In software engineering, this problem is often called “regression”. The content mentioned in this

to draw out correct answers prior to the update. We might think that it is enough if the external qualities described in Section 1.5 improve or are maintained as a whole, but in actual industrial fields, it may be difficult to accept that any component which was implemented and used to operate correctly stops operating properly at a later stage. On the other hand, additional learning inevitably draws out different output values from past input due to the characteristics of machine learning based systems. Therefore, it is necessary to examine in advance how to handle additional learning in the form of operational guidelines.

Moreover, though the Guideline do not cover directly, it is required to give consideration to legal positions of contracts and privacy of data in real environment used for additional learning, when operation is examined. From the viewpoint of quality management, it is desirable to save all data used for training including additional learning and use it for validating and monitoring the quality and verifying performance deterioration when a software component used to give correct answers. However, some restrictions on handling of data may be imposed from the viewpoint of privacy in actual applications. When data which is prerequisite for quality management at the time of designing a machine learning based system is not available at the time of its operation, the overall logic of system quality assurance may collapse. This is the reason why the identification of available and reservable data is an important factor to examine an operational system.

Approaches

There are the two patterns described earlier depending on whether quality tests are conducted by a developer, etc. when a system is updated.

(a) Cases where the quality check process comes before updates

- In advance, estimate the frequency of updates or examine judging criteria for a necessity of updates.
- Analyze the update process required at the time of operation in the design stage and envision and analyze its outline to design specific procedures prior to the beginning of operation. It is necessary to take note of at least the following points.
  - How to collect available data from an operational environment and deploy such data in a developmental environment to be updated.
  - Preprocessing method of data for additional training and method of placing filters and labels.
  - Range of data used for additional training and model updating. How to eliminate old data especially when environmental changes are expected due to

section is particularly regarded as “regression test”. However, “regression” is used in totally different contexts in the field of machine learning. Therefore, we intend to avoid the use of those terms in this document.

the passage of time.

- Examine prior to the beginning of operation a method of quality tests at the time of updates, especially, judging criteria for acceptable updates (or a method of decision-making).
- It may be necessary to decide in advance how to handle some specific cases where the quality deteriorates depending on its application.

(b) Cases where updates are made without going through the quality check process in a developmental environment

- Envision in advance a possibility of notable quality deterioration caused by additional learning and the range of impact on systems if such deterioration occurs.
- When said impact can be unacceptable, examine a technical or operational treatment to accommodate risks for overall system caused by quality deterioration of learning models due to additional learning within the acceptable range and incorporate this treatment into operation. For example, there are the following ways.
  - Technically limit the range of output values and use surrounding system components (software) to prevent any deviation from the envisioned normal range.
  - Rewind learning and stop or suspend operation based on performance monitoring from the outside of an operational environment.
- If there is a possibility of monitoring the quality prior to updates in an operational environment and operational systems, such monitoring is deemed to be useful for quality management (see Figure 15).

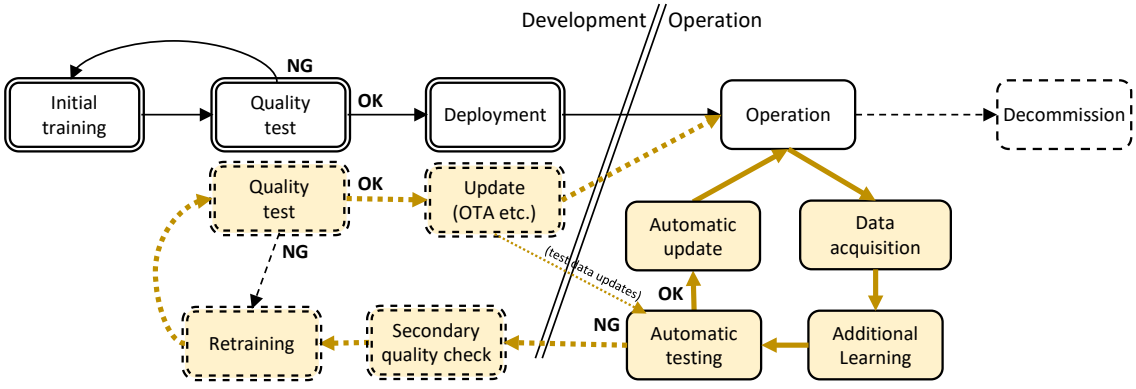

Figure 15: Possibility of automatic quality monitoring in operational environment

Requirements for quality levels

The relationship between required external characteristic levels and levels required for these internal characteristics are explained as follows:

- "Safety"
  - AISL 0.1: Lv 1 or above
  - AISL 0.2: Lv 2 or above
  - AISL 1: Lv 3
  - AISL 2~4: Requirements to be added to Lv 3 should be examined.
- "AI performance"
  - AIPL 1: Lv 1 or above
  - AIPL 2: Lv 2 or above
- "Fairness"
  - AIFL 1: Lv 2 or above
  - AIFL 2: Lv 3 or above

The requirements for each required level for the above internal characteristics are as follows.

- Lv 1
  - Examine in advance how to respond to notable system quality deterioration caused by changes in external environment.
  - In the case where online learning is given, examine in advance the impact of unexpected quality deterioration and take measures from the system side such as the limitation of operation range if necessary.
  - When additional learning is given off-line, quality management in line with the previous seven paragraph should be introduced.
- Lv 2
  - Monitor system quality deterioration and misjudgments by comparing with operation results within the range permitted by system use. It is necessary to sufficiently examine factors other than product quality such as privacy at the time of monitoring.
  - When online learning is given, regularly monitor additional learning results by any method. When any deviation from the requirements for performance is found as a result of monitoring, an immediate handling should be taken.
  - When additional learning is given off-line, conduct "regression tests on quality deterioration" with test datasets used in the system development stage to check if the quality has been maintained prior to updates. Update test datasets using the same method used in the system development stage where necessary.
- Lv 3

- Make sure to establish measures for monitoring system quality, including an operational system, compatible to privacy.
- When online learning is given, before the results of additional learning are reflected on systems, implement a mechanism to check quality to some extent within those systems so that updates are suspended if it becomes impossible to ignore unexpected quality deterioration. Make sure to ensure measures for making updates and modifications off-line.
- When additional learning is given off-line, the quality should be managed using data collected from operation, test datasets used for the initial system building and test datasets updated on a regular basis using the same method.

7 Technologies for quality management

7.1 Sufficiency of requirement analysis

A problem domain analysis (called ML problem domain analysis) is a step in the development process. Diversity of data input to machine learning components is analyzed from various viewpoints in regard to the real-world situations, in which machine learning systems or services are used. In particular, the analysis aims to concretize the identified risks or divergence in system requirements as attributes of data employed for training and testing of the machine learning products. The analysis activities are basically similar to the risk analysis, hazard analysis, or problem domain analysis, all of which are subjects in Systems Engineering. These existing technologies, or the body of knowledge, can be helpful in conducting the ML problem domain analysis.

The ML problem domain analysis, viewed from Software Engineering, is a part of initial stages of software development processes, in which software engineers often rely on trial-and-error styles of the development. In this Guideline, the steps are basically meant to establish a piece of information necessary to achieve required quality levels of machine learning products, and require activities such as what data scientists conduct in developing PoC systems with their informal know-hows concerning with the quality aspects of machine learning products.

Initial hints

Recent standard textbooks on Software Engineering (e.g. [OUJ 2019]) may present an overview of the analysis methods/steps that are employed in software development processes. Because safety analysis, or risk avoidance, is one of the primary foci in the system analysis steps, several modeling notations or methodologies have been proposed and studied. They include Causal Loop Diagram and STAMP/STAP [IPA 2016-2019]. Furthermore, the methodologies are equipped with the other detailed notations such as Fault Tree Analysis, Fault Mode and Effects Analysis, Loop Diagram, Feature Tree. In this guideline, such results are concretized as characteristics of data used to build, in particular, machine learning components. Therefore, this guideline recommends that Feature Tree is suitable as the modeling tool for the final artifacts obtained using the other modeling notations.

There might be two major problems if the analyses follow the mentioned approach; (a) modeling of input incidents, especially of risk factors, (b) design of the problem structure to be concretized as data characteristics. This section mainly refers to those concepts specific to machine learning based systems and machine learning components from these two viewpoints.

[OUJ 2019] Nakatani. T and Nakajima. S., Software Engineering (in Japanese), material of the Graduate School of the Open University of Japan, Foundation for the Promotion

of the Open University of Japan, March 2019.
[IPA 2016-2019] STAMP/STPA for Beginners (in Japanese), Information-technology Promotion Agency, Japan, 2016 - June 2019.
https://www.ipa.go.jp/ikc/reports/20190329.html

Modeling of risk factors in input space

Identifying risk factors mostly involves trial-and-error activities without any systematic means, and thus may often be conducted through brainstorming sessions by various stakeholders. It is, indeed, an analysis process regarding to Known –Unknowns. How well the risk identification is done is difficult to exhibit with quantitative measures, but may be confirmed only through the inspection by human experts.

As an example of such brainstorming activities, NASA Hazard Analysis Process, published by the National Aeronautics and Space Administration (NASA) [Deckert 2010], presents an outlined list of what should be considered in the early stage of the hazard analysis.

1) Standard Hazard List
2) Histrical experience/documentation from legacy systems
3) Your engineering training and experience

This literature also presents the “NASA Generic Hazards List” consisting of 25 items as the initial standard hazard list. Although these items include some hazard causes specific to particular applications, not causing any problem in the others, this list can be general purpose as the causes of hazard are concerned with mechanical operations of systems operating in outdoor environments. The list serves as a starting point for analyzing other systems.

Moreover, 2) is similar to what data scientists conducted in the past in order to build machine learning based systems. Since findings on similar systems in the past or early versions of the same systems are valuable, it is desirable to be utilized actively. In addition, a PoC system is recommended to act as a test-bed so that it helps the PoC developer understand those characteristics of final machine learning products. Finally, 3) can be considered as a form of utilizing what is called “domain knowledge.” It would be desirable to be incorporated in the body of knowledge for the users as well as the vendors involved in the machine learning software business. The NASA publication promotes to conduct the analysis activities based on these viewpoints, and then summarizes a conclusion as “*[Hazard analysis] [r]equires rational to justify hazard classification*”.

This guideline recommends that developers or engineers conduct the modeling of risk factors basically from the above three viewpoints together with those findings collected from the PoC stage, and record precisely the modeling process and modeling results as well. Specifically, the followings are compiled into an integrated list of identified risk factors.

– Utilization of basic knowledge on the existing functional safety design, the existing

hazard lists in each application domain, or brainstorming results based on the above NASA Hazard List;
- Utilization of analysis cases on prior systems and similar machine learning based systems;
- Introduction of domain knowledge by brainstorming with users (e.g. entrusters in contracted developments); and
- Knowledge about data employed preliminarily and exceptional cases identified in the trials of the training in the PoC stage.

[Deckert 2010] George Deckert, "NASA Hazard Analysis Process". Johnson Space Center, National Aeronautics and Space Administration. 2010.
https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100040678.pdf

Design of problem structure as characteristics of data

This stage of the development process produces, as an output artifact, a piece of information on the attachment of attributes to data used for machine learning. It is affected inevitably by the characteristics of the next process, machine learning and training. Considering the training process that follows, the main activities of the process is to identify "attributes which are handled distinctively throughout the machine learning process," which consists of the following two aspects.

4) Problematic if data are not recognized as distinct with respect to the differences in the output result values or risks; and

5) Problematic if data, resulting in the same output results, are likely to be understood as different due to their input values and characteristics of machine learning models.

4) is drawn mainly from the system specifications and the hazard analysis. A difference in output values is a difference in labels of training data, and risk incidents with different levels can basically be enumerated once the cause of hazard is identified.

On the other hand, 5) is dependent on application domains, for example, depending on differences in background images or character fonts for image recognition tasks. Moreover, 5) may be affected by preprocessing or architecture network of machine learning models. It is necessary to forecast a final implementation form to some extent in order to conduct such an analysis. This is called "Implementation Forecast" in software engineering because the information relating to the final implementation is indeed necessary to conduct the analysis at the early stage of the development. That may result in distorting risk analysis if carried out inappropriately, but it is inevitable for the quality management of machine learning components. Specifically, the activities are concerned with the implementation consideration made in the PoC development stage, and the analysis of PoC behavior using data.

In order to extract viewpoints with which the forecast is actually carried out, the three points listed in the previous section are considered applicable in view of “a risk that AI built makes different judgments in similar situations”. In the future, it would be desirable to combine 2) and 3) in the previous section and 5) in this section to accumulate system analysis results for each application area to some extent and compile them as detailed standard hazard lists in each area.

7.2 Coverage of distinctive problem cases

General

The internal quality in this section has two purposes, which follows the analyses described in Section 7.1. First is to prepare data needed at the time of training/testing so as to avoid “an unknown situation which a machine learning based system has not been learnt nor tested”. Second is to either decimate or integrate systematically an enormous number of combinations of prepared attributes so that building machine learning software of a required quality level is practical.

If a system intends to solve a simple problem and the number of combinations of attributes is a few dozen, it is possible to consider the amount of data that achieves a certain recognition level for each attribute. (Example: Only 10 characters x 2 fonts need to be recognized in number recognition and there is no need to take into account differences in paper quality. Then, there are 20 combinations.) If each attribute has such an amount of data, sufficient and comprehensive training can be practiced. However, since the number of combinations of attributes often exceed several tens of thousands if calculated in a naive manner, it is not practical to prepare data representing all combinations for testing. In a certain case, training data or testing data may be a few or even null as a result of breaking up attribute combinations. It would then be impossible to rely on the Law of large numbers to make a judgment such as the convergence of training.

“Combination testing” is developed in conventional software engineering for mitigating the complexity problem. The method, employing the notion of “test coverage,” establishes the degree to which combinations of attribute are detailed and the degree to which combinations of test condition are covered, yet devises them separately. The method is often used to evaluate test data in the conventional software testing, and is also expected to be effective for the case of machine learning, because training is data-centric. For example, [Barash 2019] presents examples to apply the combination test technology (t-way) to evaluate dataset for machine learning.

[Barash 2019] Guy Barash, Eitan Farchi, Ilan Jayaraman, Orna Raz, Rachel Tzoref-Brill, and Marcel Zalmanovici, Bridging the gap between ML solutions and their business requirements using feature interactions, ESEC/FSE 2019, pp.1048-1058. 2019.

https://dl.acm.org/citation.cfm?id=3340442

7.3 Coverage of datasets

Establishing coverage of datasets is an extremely difficult but important problem in quality management of machine learning. A dataset, if a certain anomaly is associated with it, may derive trained machine learning models to exhibit unstable or unfavorable functional behavior in a certain specific envisioned situation, Lowering fairness levels.

As the problem domain analysis is finished, this section focuses on the issues with small differences in the dataset. The differences are so small that they are easily overlooked at the time of constructing the dataset. The discussions in this section are mostly related to the preliminary stages of data preparation or of data evaluation in the development lifecycle, auxiliary checks may be sometime conducted in the software testing stage as well.

Plan for data acquisition

A data acquisition plan in the data preparation stage is a basis to study the issues relating to the coverage of dataset. In order to construct a dataset that are unbiased in view of operation time circumstances, the plan must refer to the range, period and size of data collection. Specific plans should be considered for each application following outcomes of the brainstorming sessions, which may be those activities, in the stage of analyzing “domain analysis problem,” to utilize prior knowledge about the similar applications so as to examine in advance the degree of diversity.

Pre-flight tests in data scrutinization stage

Auxiliary checking of data distribution belonging to certain attributes is desirable in some cases. These attributes are what, although identified as candidates in the early stages of the analysis, may not be adopted, or what may become latent, because they are composed to be a new attribute. The composed attributes are accompanied with explicit specifications, and in some cases those checks may be conducted mechanically, because data labels are given.

This is not perfect, because attributes overlooked in the brainstorming sessions cannot be recovered. It is worth considering.

Moreover, for some data, what is called data specific featuremay be identified in the preprocessing stage, which makes it possible to check the distribution.

Additional tests in testing stage

In a case where the data distribution itself has some problems, what can be re-checked in the test stage is limited. For instance, there may be latent correlations between the adopted

features and overlooked unidentified ones. In such cases, analyses of correlation or influence between the input values and their resultant counterparts output from the trained machine learning model may be helpful to see whether the model behaves as expected. The model may make inferences based on features different from what are assumed. It is worth considering this analysis when extreme safety is a major concern.

7.4 Uniformity of datasets

The uniformity of datasets is equivalent to considering the coverage of datasets, described in Section 7.3, as a single “case” of all the input data. The measures listed in each section of 7.3 are deemed applicable here as well.

7.5 Correctness and stability of trained machine learning models

Checking the correctness and stability of trained machine learning models corresponds to “unit test” of software testing in the conventional software development. This section explains

- Issues, which must be considered in software testing, that come in particular from the characteristics of machine learning software when the software testing technology [1] is applied to “inference engines” whose behavior is governed by the machine learning models; and
- Future possibilities of technologies to check directly the stability aspects.

Software testing of machine learning components

First, for machine learning components, the quality of inference results is viewed in terms of the correctness. The correctness levels are checked by means of software testing in which test data are input to the program. The testing problem is, however, more complicated than the case for conventional software systems or programs. It is because two kinds of programs are involved, a training or learning program and a prediction or inference program, both of which may have much impact on the correctness.

Second, correctness criteria are not apparent, which is an instance of the oracle problem [2]. A training or learning program accepts a training dataset as an input and produces a trained machine learning model as its output result. However, the correct result with respect to the input training dataset is not known in advance, and thus checking whether the program behaves as required is not possible. Moreover, a prediction or inference program returns a probabilistic value, which implies that absolute correctness criteria are hard to be defined [3]. Because of these characteristics, machine learning components are categorized as non-testable [4].

7.5.1.1 Two views on software testing

In general, software testing is sometimes discussed in terms of two different objectives, either positive or negative testing. The differences manifest themselves particularly in the ways to generate test data.

Positive testing aims to check whether test target programs behave properly as envisioned. Now, a test target program is assumed to have pre- and post-conditions. The conditions are predicate logic formulas, which states as “if the pre-condition is satisfied, then the post-condition becomes true in the program state in which the program execution is completed.” Positive testing is conducted with test data satisfying the pre-condition. The program passes the test if the execution result satisfies the post-condition. Therefore, pre-conditions must be known for positive testing, and post-conditions act as the test oracle.

Negative testing aims to check whether test target programs do not cause any destructive failure even under unexpected conditions. Generally, programs are so implemented that they behave not improperly in response to data not satisfying the pre-conditions. Programs must not cause fatal damage on their execution environment, for example, no core dump. Generating test data for negative testing may make use of known pre-conditions because such input data are so specified not to satisfy the pre-conditions. Furthermore, the test oracle is not explicit, called Implicit Oracle, if the program does not terminate properly.

Fuzz testing [5] (or fuzzing) is a particular method for negative testing, which is found successful for open systems such as operating systems, Web systems, or security systems. Generating fuzz, test data, appropriate for the test objectives is mandatory.

7.5.1.2 Metamorphic testing

Metamorphic testing (MT)[6][7] was proposed as a systematic method of testing non-testable programs whose test oracles are not apparent. MT adopts a powerful notion of partial oracle that is based on golden outputs of the test target program. Successful application cases include testing of numerical programs, translators including compilers, or security programs. Currently, metamorphic testing is one of the standard methods for testing machine learning components [8].

7.5.1.3 Generating test input in metamorphic testing

Test data generation, called follow-up data generation, is one of the key technical elements in metamorphic testing (MT). The generation method must make use of the characteristics of the test objectives, either positive or negating testing, because the two views are based on different requirements on test data selection methods (see 7.5.1.1). For the case of applying MT to machine learning components, the notion of dataset diversity[9][10] plays an important role to identify test input, which are slightly different from the already available

data, but serve as the same machine learning task. The idea of dataset diversity is particularly successful in unit-testing of training or learning programs.

Test data for prediction or inference programs must take into account various cases, depending on the test objectives. Data that do not appear in the training dataset are mandatory for negative testing. Those methods to figure out such data are known, in the machine learning research, as data augmentation. Because it is adapted in testing, the method is sometimes called test-time augmentation. Some such methods make use of machine learning technology to generate data for testing of machine learning components. There are several methods for image classification tasks, such as classical methods of data augmentation [11], or generative adversarial networks (GAN) [12]. Data generated with the GAN technology increase the coverage of testing cases, and thus are effective for positive testing as well.

[1] P. Ammann and J. Offutt: Introduction to Software Testing, Cambridge University Press 2008.
[2] E.T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo: The Oracle Problem in Software Testing: A Survey, IEEE TSE, 41 (5), pp.507-525, 2015.
[3] S. Elbaum and D.S. Rosenblum: Known Unknowns - Testing in the Presence of Uncertainty, In Proc. 22nd FSE, pp.833-836, 2014.
[4] E.J. Weyuker: On Testing Non-testable Programs, Computer Journal, 25 (4), pp.465-470, 1982.
[5] B.P. Miller, L. Fredricksen, and B. So: An Empirical Study of the Reliability of UNIX Utilities, Comm. ACM, vol.33, no.12, pp.32-44, 1990.
[6] T.Y. Chen, S.C. Chung, and S.M. Yiu: Metamorphic Testing - A New Approach for Generating Next Test Cases, HKUST-CS98-01, The Hong Kong University of Science and Technology, 1998.
[7] T.Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, Y.H. Tse, and Z.Q. Zhou: Metamorphic Testing: A Review of Challenges and Opportunities, ACM Computing Surveys, vol.51, no.1, Article No.4, pp. 1-27, 2018.
[8] S. Nakajima: A Brief Introduction of Machine Learning Software Testing (in Japanese), a report of the Technical Committee Submission System, Institute of Electronics, Information and Communication Engineers, 2020.
[9] S. Nakajima: Software Testing of Dataset Diversity (in Japanese), Computer Software, 35(2), pp.26-32, 2018.
[10] S. Nakajima and T.Y. Chen: Generating Biased Dataset for Metamorphic Testing of Machine Learning Programs, Proc. IFIP-ICTSS 2019, pp.56-64, 2019.
[11] A. Krizhevsky, I. Sutskever, and G.E. Hinton: Imagenet Classification with Deep Convolutional Neural Networks, In Adv. NIPS 2012, pp.1097-1105, 2012.
[12] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio: Generative Adversarial Nets, In Adv. NIPS 2014, pp.2672-

2680, 2014.

Technologies on stability issues

The technologies on stability issues are broadly divided into the evaluation and improvement. Figure 16 illustrates how each technology is related to the level described in Section 6.6.3.

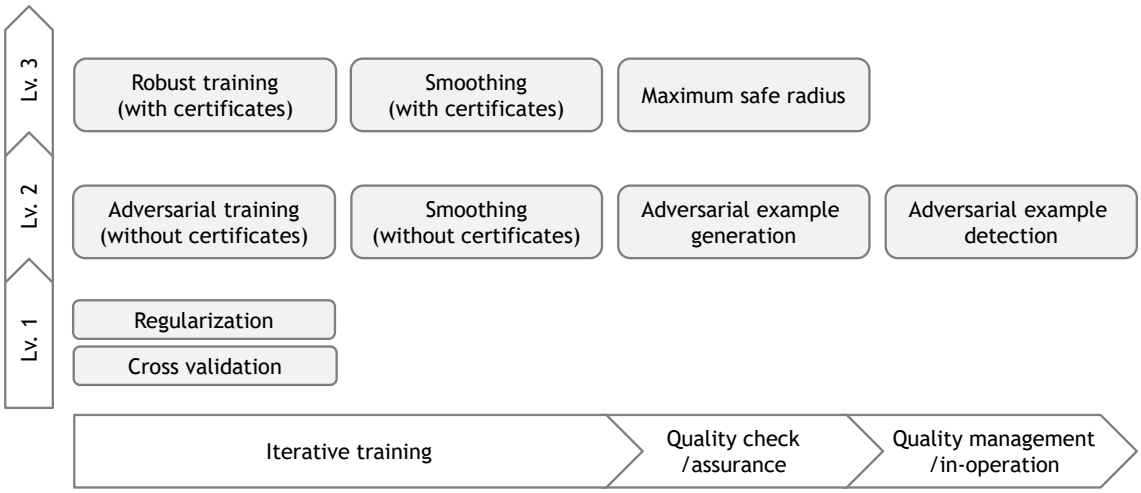

Figure 16: Technologies to evaluate and improve stability (process and levels to which technologies are applied)

7.5.2.1 Cross validation

Cross validation is a classical method to mitigate the over-fitting problem, and thus to improve stability levels. The idea is simply dividing a whole dataset into training, validation, and testing datasets [Kohavi 1995]. For example, in K-division cross validation, a whole dataset is divided by K, and $1/K$ of the dataset is used for validation and the remaining $((K-1)/K)$ datasets for training. The roles of datasets, either training or validation are interchanged. Cross validation is recommended to Lv 1.

7.5.2.2 Regularization

Regularization is a methodology to mitigate the over-fitting problem, and thus to improve stability levels. Especially, regularization methods suppress the absolute values of learnt weight parameters not to become excessively large [Nowlan 1992]. For example, loss functions may include a regularization term to increase as absolute values of parameters become large). Dropout is another regularization method [Srivastava 2014], in which neurons are randomly excluded from the training target during the training process. Dropout

may obtain the same effects as cases where several machine training models are trained simultaneously. Regularization is recommended to Lv 1 if the stability is an issue.

7.5.2.3 Adversarial example generation

Adversarial examples are those data that cause miss-inference in machine learning components. Such data are augmented with a slight perturbation, a semantic noise, that human eyes are not able to recognize. Various methods to generate adversarial examples have been proposed [Pei 2017, Carlini 2017]. A stable machine learning component is expected not to cause miss-inference even when the perturbation with the input adversarial data is large. Therefore, the degree of perturbation is used to evaluate the stability. Unfortunately, no practical tool to generate such adversarial examples have been established, this technology will be hopefully applicable to Lv. 2 evaluations in the future.

7.5.2.4 Maximum safe radius

The maximum safe radius refers to the minimum distance between the original data and its adversarial example. Adversarial example generation (7.5.2.3) looks for nearby adversarial examples, while the maximum safe radius guarantees that there is no adversarial example in the neighborhood (inside the hyper-sphere of the maximum safe radius). Methods to accurately calculate the maximum safe radius are proposed in [Katz 2017, Tjeng 2019] that use SMT solvers. However, a cost of accurately calculating the maximum safe radius is so high that the size (number of neurons) of networks is limited. Alternatively, methods to approximately calculate a safe radius smaller than the maximum safe radius are proposed in [Weng 2018, Boopathy 2019, Wu 2019]. Moreover, a method to approximately calculate a radius that probabilistically guarantees safety (though not 100%) is proposed in [Weng 2019]. These technologies are still in the research stage, but they can guarantee that there is no adversarial example inside the hyper-sphere of the maximum safe radius. Therefore, they are expected to be applicable to Lv.3 in the future.

7.5.2.5 Adversarial training/robust training

Adversarial training is a learning method, which looks for neighborhood data that is likely to cause miss-inference [Madry 2019]. Compared to the standard training method, it mat be able to improve the resistance of trained machine learning models against adversarial examples. Although this technology is still in the research stage, it is expected to be applied to Lv.2 in the future.

On the other hand, a robust training method is to eliminate neighborhood adversarial examples [Wong 2018]. An approximate calculation method of the maximum safe radius is given together to guarantee that adversarial examples do not exist near each data in the

training dataset. Although this technology is still in the research stage, it is expected to be applied to Lv.3 in the future.

7.5.2.6 Randomized smoothing

Randomized smoothing is a method to augment data with randomized noise so as to calculate the final result value to be a mean with respect to the noise distribution. The method is reported to improve the stability against the L2-norm adversarial attacks [Liu 2018]. Furthermore, methods of adding randomized noise are proposed so as to guarantee the correctness of inference results [Lecuyer 2019, Cohen 2019]. In these methods, the randomized smoothing is applied to the neighborhood of input data. Because the statistically expected value is approximated by an average, multiple running results are needed. Furthermore, the stability degree is increased as the randomized noise is large, which in turn lowers the correctness. Such a trade-off relation must be taken into account. Nevertheless, this technology is expected to be applied to Lv.3 in order to guarantee the stability of inference results.

7.5.2.7 Adversarial example detection

Adversarial examples detection is a method to check, at runtime, whether incoming data are adversarial or not, and is one of the active research areas [Xu 2018, Ma 2019]. Although such a technology is still in the research stage, they will be expected to be applied to Lv.2 for protecting potential adversarial examples in machine learning components.

References:

[Kohavi 1995] Rob Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, The 14th International Joint Conference on Artificial Intelligence 2 (12): 1137–1143, 1995.

[Nowlan 1992] Steven Nowlan and Geoffrey Hinton, Simplifying neural networks by soft weight-sharing. Neural Computation, 4(4), 1992.

[Srivastava 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15(Jun), pp.1929−1958, 2014.

[Pei 2017] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana, DeepXplore: Automated Whitebox Testing of Deep Learning Systems, The 26th Symposium on Operating Systems Principles (SOSP 2017), pp.1-18, 2017.

[Carlini 2017] Nicholas Carlini and David Wagner, Towards Evaluating the Robustness of Neural Networks, IEEE Symposium on Security and Privacy (SP), pp.39-57, 2017.

[katz 2017] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer, Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, International

Conference on Computer-Aided Verification (CAV), 2017.

[Tjeng 2019] Vincent Tjeng, Kai Xiao, and Russ Tedrake, Evaluating robust- ness of neural networks with mixed integer programming, International Conference on Learning Representations, 2019.

[Weng 2018] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S. Dhillon, and Luca Daniel, Towards Fast Computation of Certified Robustness for ReLU Networks, The 35th International Conference on Machine Learning, PMLR 80, pp.5276-5285, 2018.

[Boopathy 2019] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel, CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks, The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019), pp.3240-3247, 2019.

[Wu 2019] Min Wu, Matthew Wicker, Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska, A Game-Based Approximate Verification of Deep Neural Networks with Provable Guarantees, Theoretical Computer Science (2019), https://doi.org/10.1016/j.tcs.2019.05.046

[Weng 2019] Tsui-Wei Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel, PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach, The 36th International Conference on Machine Learning (ICML 2019), PMLR vol. 97, pp.6727-6736, 2019.

[Madry 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks, The Sixth International Conference on Learning Representations (ICLR 2018). 2018.

[Wong 2018] Eric Wong and J. Zico Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope, The 35th International Conference on Machine Learning (ICML 2018), PMLR vol. 80, pp.5283–5292, 2018.

[Liu 2018] X. Liu, M. Cheng, H. Zhang, and C. Hsieh, Towards robust neural networks via random self-ensemble, The European Conference on Computer Vision (ECCV 2018), 2018

[Lecuyer 2019] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana, Certified Robustness to Adversarial Examples with Differential Privacy, The IEEE Symposium on Security and Privacy (SP), 2019.

[Cohen 2019] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter, Certified Adversarial Robustness via Randomized Smoothing, The 36th International Conference on Machine Learning (ICML 2019), 2019.

[Xu 2018] Weilin Xu, David Evans, and Yanjun Qi, Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, Network and Distributed Systems Security Symposium (NDSS), 2018.

[Ma 2019] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang, NIC: Detecting Adversarial Samples with Neural Network Invariant Checking,

Network and Distributed Systems Security Symposium (NDSS), 2019.

7.6 (skipped)

The technologies with regard to the stability of trained machine learning models are described in Section 7.5 together with those regarding to the accuracy.

Section 7.6 is intentionally skipped so as to maintain the consistency between Chapter 6 and Section 1.7 and the corresponding section numbers in Chapter 7.

7.7 Dependability of underlying software system

General

The dependability of underlying software system is an important item even in conventional software. Quality management is expected to be difficult especially in implementation of machine learning AI which uses a large number of libraries including open-source libraries. Open-source software does not usually count on guarantee. Therefore, in relation to end users, open-source users (developers and operators) are responsible not only for differences in malfunction but also for discovering and monitoring potential errors and, in some cases, making modifications.

Moreover, when a machine learning model is built, it has been revealed that a training process incorporates bugs (program error) so that the impact of bugs does not appear in tests, causing quality deterioration [Nakajima 2020].

On the other hand, in relation to conventional software, configuration management and quality monitoring of libraries and fundamental software are emphasized and infrastructure and services for configuration management and quality monitoring are improving. Some data included in those services contain information such as libraries specific to machine learning, although it is somewhat limited. These existing infrastructures are deemed worth being utilized in applications of machine learning.

Quality management of open-source software

It is desirable for each business operator to think about how much open-source libraries can be trusted and their quality maintained by itself or outsourced.

It can be expected that supported libraries whose quality is assured and software that went through the quality inspection process are used where necessary in relation to a necessary quality level.

Configuration management and tracking of bug information

For example, common Platform Enumeration (CPE) [Mitre:CPE] is used for configuration management of software components as a common ID to list system constituent components, in the security field. There are commercial products that manage versions of software components in use and extract information on their updates based on CPE. A list of vulnerability information related thereto called Common Vulnerability Enumeration (CVE) [Mitre:CVE] does not include bugs that are not directly related to security vulnerability. However, at least tools related to CPE are very likely to be beneficial to track the latest version of libraries and infrastructure software.

Possibility of specific check thorough testing

Moreover, when constituting software components have any bug, that bug does not always have a direct effect on actual ML results in machine learning components different from conventional software. If software with bugs is placed in a feedback loop of learning or training, a trained learning model memorizes behaviors of said bug and training seems to be superficially successful in some cases. In these cases, it is reported that potential quality deterioration caused by software bugs can be found by analyzing statistical behaviors using any of the metamorphic test technologies listed in Section 7.5.1.2.

Software update and possible adverse effects on performance and operation

On the other hand, actual software whose configuration is complex often operates unintentionally even if a developer of software libraries updates it with the intention of improving performance and operation. Depending on how a machine learning based system is configured, behaviors of bugs are sometimes adapted or learnt in the training process. Therefore, when any software constituent component is updated, its operation and performance must be reviewed. In some cases, it is important, to make a judgment to start training over or keep using an older version taking into account vulnerability.

This Guideline cannot recommend any option, because a specific judgment depends on each situation. However, it is important to record the background of each judgment for accountability reasons in order to claim “the proper quality management”

References

[Mitre:CPE] MITRE, Common Platform Enumerations. http://cpe.mitre.org/
[Mitre:CVE] MITRE, Common Vulnerability Enumerations. https://cve.mitre.org/
[Nakajima2020] S. Nakajima: Quantitative Indicators of Distortion of Trained Machine Learning Models, Study Group on Software Science, Institute of Electronics,

Information and Communication Engineers ,2020

7.8 Maintainability of qualities in use

This section describes technologies to maintain internal qualities satisfied at the commencement of operation throughout the operation period. In order to maintain internal qualities realized at the beginning of operation in spite of changes in external environments that may arise during operation, a machine learning component needs to respond to those changes. As described in Section 6.8.1, there are two operational patterns therefor. One is to make batch-processing updates by returning to the developmental environment and deploying it again. Another is to automatically update the software component when needed or at a high frequency in the operational environment. In the former pattern, monitoring to determine the timing of update and update processing are main elements, while in the latter pattern, automated update processing is a main element. In order to realize this, on-line learning [Shalev-Shwartz 2012] is adopted. Even if updates are made automatically, it is necessary to monitor if automatic updates operate properly and process updates to respond to cases where there is any deviation from normal operation conditions.

Some of monitoring technologies necessary for both patterns are presented here. We focus especially on technologies to detect changes in data distribution over time called concept drift [Gama 2014]. Moreover, technologies to retrain machine learning models which play the core role of updating trained machine learning models and technologies to create additional training data used therein will be presented.

Monitoring

The relation between newly-acquired input data and output (inference) results may have changed at the time of operation from the relation between them at the time of training due to various factors including changes in external environments. When a machine learning model trained in the design or development stage using such data whose input and output relations change is kept using during operation, its performance (accuracy) may deteriorate and result in serious damage. Therefore, it is required to continuously monitor behaviors of machine learning based systems and machine learning components for the purpose of checking if the quality fulfilled at the beginning of operation is maintained throughout the operation period.

There are the following four tasks of monitoring during operation.

- Accuracy monitoring
- KPI monitoring
- Model output monitoring
- Input data monitoring

"Accuracy monitoring" directly measures the accuracy of trained machine learning models. This monitoring is divided into some patterns in accordance with the method of collecting correct answers compared with inference results of trained machine learning models required for calculating the accuracy. That is, after making an inference, (1) cases where correct answers are acquired automatically after a certain period, (2) cases where correct answers cannot be acquired automatically so that it is necessary to manually put labels and (3) cases where manual labeling is beyond budget or it is impossible to put labels. It is necessary to select an appropriate monitoring method in accordance with the above grouping of cases where correct answers are collected in order to appropriately monitor the accuracy. Moreover, some applications emphasize not only monitoring the accuracies of models but also monitoring from the viewpoint of KPI, "KPI monitoring", in line with a conversion rate and the benefit of users. In this case, it is required to monitor the consistency between the accuracies of the models and the KPIs of the applications.

"Model output monitoring" and "input data monitoring" refer to the monitoring of results of inferences made by a trained machine learning model and the monitoring of its input data, respectively. The monitoring methods are divided into human monitoring (100% sampling), automated monitoring (alert conditions are known) and filtering (conditions with higher possibilities of alert are known). Model output monitoring is further categorized into a case where each output inference is checked by experts as in the case of medical diagnostic and a case where all inferences are checked altogether after a certain period of time. In the same way, various conditions may be imposed in the case of monitoring by filtering (For example, false positive is acceptable but false negative is unacceptable). In addition, when input data is monitored, whether an alert is issued in the case of monitoring by filtering or input is disregarded depends on an application.

Concept drift detection methods

Concept drift is one of major causes of the deterioration of the accuracy of trained machine learning models during operation. A variety of monitoring (detection) methods have been proposed recently. Concept drift detection methods are categorized as shown in Table 4 in accordance with whether correct-answer labels on data acquired during operation are used [Sethi 2017].

Table 4: Classification and characteristics of concept drift detection methods

Use of label	Method	Characteristics
Labeled detection (Supervised detection)	Sequential analysis	Monitoring of absolute values such as accuracy
	Statistical Process Control	Monitoring of the increase or decrease of error rate (Early detection by finding indicator)
	Window based distribution	

	monitoring	Monitoring of distribution differences between training time and operation time
Unlabeled detection (Unsupervised detection)	Novelty detection / clustering method	Simple detection by clustering
	Multivariate distribution monitoring	Detection by applying statistical hypothesis testing to data distribution
	Model dependent monitoring	Output dependent of models, such as confidence score

Studies of unlabeled detection methods have been carried out actively in recent years. For example, a literature [Faria 2013] proposes an unknown class detection algorithm (MINAS) in multi-class problem based on clustering methods such as k-means. Moreover, another literature [Reis 2016] proposes an algorithm of incremental KS test which improved the amount of calculation by developing incremental Kolmogorov-Smirnov test to judge whether samples are generated from the same distribution. Moreover, a literature [Lindstrom 2011] proposes a method of detecting data changes by using confidence scores associated with output from trained machine learning models without using labels (CDBD).

Retraining

When any change in data distribution or deterioration in the accuracy of a trained model is detected as a result of the monitoring describe above, it is necessary to retrain the machine learning models using datasets which add recent data or replaced recent data with existing training data. Many researches have been made about this retraining. For example, a literature [Tian 2018] provides a platform design method to automatically determine the appropriate timing of model update in relation to existing machine learning frameworks. Moreover, it has also been proposed a method of applying balanced parameters of both “existing models” and “models that learnt new tasks” to retrain models in order to reduce forgetting of old tasks caused by retraining of a neural network called catastrophic forgetting [Lee 2017].

Creation of additional training data

There are many cases where labels cannot be collected automatically. In this case, it takes much cost to manually place labels to new data acquired at the time of operation. A study on reduced costs of retraining by reducing the number of data to which labels are attached has been proposed through a method of reducing labeling work using software equipped with GUI (Graphical User Interface) [Vostrikov 2019] and active learning have been proposed to solve this problem [Settles 2009].

References:

[Shalev-Shwartz 2012] S. Shalev-Shwartz, Online learning and online convex optimization, Foundations and Trends® in Machine Learning, 2012, 4.2, pp. 107-194.

[Gama 2014] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, A survey on concept drift adaptation, ACM computing surveys (CSUR), 2014, 46.4, pp. 1-37.

[Sethi 2017] T. S. Sethi, M. Kantardzic, On the reliable detection of concept drift from streaming unlabeled data, Expert Systems with Applications, 2017, 82, pp. 77-99.

[Faria 2013] E. R. Faria, J. Gama, and A. C. Carvalho, Novelty detection algorithm for data streams multi class problems, In Proc. of the 28th annual ACM symposium on applied computing, 2013. pp. 795-800.

[Reis 2016] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, Fast unsupervised online drift detection using incremental Kolmogorov Smirnov test. In Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2016, pp. 1545-1554.

[Lindstrom 2011] P. Lindstrom, B. M. Namee, and S. J. Delany, Drift detection using uncertainty distribution divergence In Proc. of the 2011 IEEE 11th Int. Conf. on Data Mining Workshops, 2011, pp. 604-608.

[Tian 2018] H. Tian, M. Yu, and W. Wang, Continuum: A platform for cost aware, low latency continual learning, In Proc. of the ACM Symposium on Cloud Computing, 2018. pp. 26-40.

[Lee 2017] S. Lee, J. Kim, J. Jun, J. Ha, and B. Zhang, Overcoming catastrophic forgetting by incremental moment matching. In Proc. of the Advances in Neural Information Processing Systems, 2017. pp. 4652-4662.

[Vostrikov 2019] A. Vostrikov, S. Chernyshev, Training sample generation software. I. Czarnowski, R. Howlett, L. Jain (eds), Intelligent Decision Technologies 2019. Smart Innovation, Systems and Technologies, vol 143. Springer, Singapore, 2019.

[Settles 2009] B. Settles, Active learning literature survey, University of Wisconsin Madison Department of Computer Sciences, 2009.

8 (informative) Information on related documents

The content of this Chapter is informative.

8.1 Relation with other guidelines

Contract Guidelines on AI of the Ministry of Economy, Trade and Industry

"The Contract Guidelines on Utilization of AI and Data[16]" published by the Ministry of Economy, Trade and Industry summarizes considerations concerning contracts with business partners when they cooperatively develop systems containing AI (e.g. machine learning) in accordance with contracts such as sales order or quasi-mandate.

The Contract Guidelines clarify roles and responsibilities in contracts between business partners engaged in development, while the Contract Guidelines summarize quality of service which providers have to provide to users of developed system. From the standpoint of this Guideline, qualities in use of products and services defined in this guideline are elaborated with the cooperation of all stakeholders listed in the Contract Guidelines for development, and then they are provided to users. It is out of the scope in this Guideline how to concretely realize the quality with the cooperation of business partners and how to conclude a contract and share responsibilities. Stakeholders should agree on these matters based on the Contract Guidelines. On the other hand, this Guideline can serve as a base of examining technical matters related to quality when stakeholders share responsibilities and exchange information. In Section 5.2, we briefly analyzed a possibility of sharing roles as one example.

The Contract Guidelines state that a non-waterfall model is appropriate for sales orders in the development process, while the Contract Guidelines (p.46) focus not only on (3) "Development phase" but also phases such as (4) "Additional learning phase" before and after (3). Therefore, this Guideline defines a system lifecycle process by a wide range from system planning to disposal after operation. A model mixed with waterfall model is basically used and summarized as "Figure 5: Conceptual diagram of mixed machine learning lifecycle process" in page 22. With regard to the relation with "gradually exploratory" development processes recommended in the Contract Guidelines, the developmental phase in the center of Figure 5 in this Guideline corresponds basically to the development phase in the Contract Guidelines.

Relations with Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence (QA4AI)

The Consortium of Quality Assurance for Artificial Intelligence-based products and service

[16] Guidelines for Contracts of Use of AI and Data, Ministry of Economy, Trade and Industry, June 2018, https://www.meti.go.jp/press/2018/06/20180615001/20180615001-3.pdf.

(QA4AI) in Japan has published “Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence 2019.05”[17] in May 2019. The QA4AI Guidelines propose the following five axes to be considered in quality assurance of AI products with the aim of giving “common guidelines for quality assurance of AI products”.

A) Data integrity
B) Model robustness
C) System quality
D) Process agility
E) Customer expectation

Moreover, the QA4AI Guidelines present check lists of these axes and a list of specific quality management technologies.

As regards the relation between this Guideline with the QA4AI Guidelines, the three quality-assurance axes above (A, B and C) are considered to correspond to the internal qualities listed in Section 1.7 (page 14) of this Guideline. Therefore, the technologies specifically listed in the QA4AI Guidelines correspond to the quality management method for each internal quality characteristic presented in Chapter 7 and give some insight to engineers who are responsible for realizing the quality with the complementary help of the two guidelines. Table 5 explains the specific relation between the items listed in the checklists for those three axes of the QA4AI Guidelines and the internal qualities listed in Section 1.7 of this Guideline.

Although no clear quality management axis has been established in this Guideline for the quality assurance axes (D and E above), they can be realized through the application process (Section 5.1) envisioned in this Guideline (Section 5.1) and a business process with customers included therein.

Consequently, the QA4AI Guidelines published by the Consortium are beneficial as a reference for engineers who actually create machine learning-based AI to find feasibility of technologies to improve and elaborate internal qualities of machine learning components. On the other hand, this Guideline intends to comprehensively analyze matters necessary for businesses that plan and develop overall systems containing machine learning-based AI to ensure qualities in use throughout the lifecycle process and list them as much as possible. Therefore, it is thought that these two guidelines complement each other.

Table 5: Analysis of the relation with the QA4AI Guidelines (Version of May 2019)

Internal quality characteristics in this	Checklist in the QA4AI Guidelines (note: The circled numbers refer to individual items listed

[17] Consortium of Quality Assurance for Artificial-Intelligence-based products and service, Guidelines for Quality Assurance of Machine Learning-based Artificial Intelligence, May 2019, http://qa4ai.jp/download/.

Guideline	in Section 2.2 of these Guidelines.)
1.7.1 Sufficiency of problem domain analysis	2.2.1 Data integrity ② Are samples selected from a required population? Does it use real data? Does it exclude unnecessary data or inappropriate population data? ⑤ Is data too complex or too simple? Does each sample include necessary elements appropriately? Are labels appropriate? 2.2.2 Model robustness ⑲ Is quality validated with sufficiency-diverse data taking into account mathematical diversity, semantic diversity, social diversity and cultural diversity?
1.7.2 Coverage for distinguished problem cases	2.2.1 Data integrity ① Is the amount of data appropriate? Are costs appropriate? Is the amount meaningful? Is it allowed to augment data? ② Same as above
1.7.3 Coverage of test datasets	2.2.1 Data integrity ④ Is unintended bias or contamination removed? Is intended bias appropriate? ⑥ Are the characteristics in data (e.g. multicollinearity) considered appropriately?
1.7.4 Uniformity of training datasets	2.2.1 Data integrity ④ Same as above
1.7.5 Correctness of the trained models	2.2.1 Data integrity ③ Are the requirements for data satisfied? Is it monitored if restrictions on data are not violated? ⑦ Does data show normal values? Does each outlier really fall outside the range? Does missing indicate anything? Are outliers and missing values handled appropriately? ⑨ Are training data and test data separated? 2.2.2 Model robustness ⑫ Is the accuracy such as the rate of correct answers, precision and recall rates, and F values appropriate? ⑬ Is generalization performance ensured? ⑭ Are indicators of models (AUROC, etc.) sufficient? ⑮ Are models appropriately trained? Are local optimal

	solutions avoided? ⑯ Is the appropriateness of algorithms and hyperparameters examined? ㉒ If it is difficult to measure target indicators, is measurable metrics appropriately applied instead?
1.7.6 Robustness of the trained model	2.2.1 Data integrity ⑬⑮⑯ ⑰ Is cross validation carried out sufficiently? ⑱ Is data robust against noise?
1.7.7 Reliability of underlying software system	2.2.1 Data integrity ⑪ Is the meaning of data damaged by malfunction of any training software component or data generation component?
1.7.8 Maintainability of quality during operation	2.2.1 Data integrity ⑩ When online learning is available, is its effect taken into account appropriately? 2.2.2 Model robustness ⑳ Is degradation acceptable? Is the range of degradation effect ascertained? Is it possible to reproduce learning? Is there any difference between behaviors during learning and those in operation? ㉑ Is a component not obsoleted? Is the prediction quality of real data not deteriorated? 2.2.3 System quality ㉔ Do behaviors of overall system such as performance not deteriorated?
Items corresponding to external qualities	2.2.3 System quality ㉓ Are values provided appropriately? ㉕ Are evaluations made for a whole system or in meaningful unit? ㉖ Is criticality of possible quality incidents kept to an acceptable level? ㉗ Is the frequency of events that may cause quality incidents considered low? Are the frequency of occurrence, coverage and environmental control of incidents are examined sufficiently? ㉘ Are the attainment level of system incidents, safety

	functions and resistance to attacks sufficient? ㉙ Is the contribution level of AI controlled? Is a system appropriately and rapidly adapted to changes in the other (AI or non-AI) systems which the system depends on? Is the impact of malfunction kept low sufficiently? ㉚ Are the levels of guarantee, accountability and conviction sufficient?
Items that have no corresponding characteristics	2.2.1 Data Integrity ⑧ Are ownership, copyrights/intellectual property rights, confidentiality and privacy considered appropriately?

8.2 Relations with international initiatives for quality of AI

Currently, diverse international initiatives for quality of AI have been taken including those mentioned below. This Guideline adopts adaptable parts from these initiatives. Moreover, we will actively present outstanding knowledge acquired from this guideline toward international standardization.

Quality and safety

An examination on quality and safety has started at ISO/IEC JTC 1/SC 42/WG 3. Gaps with existing standards for quality characteristics and quality assurance technologies of AI (ISO/IEC 25000 (SQuaRE), ISO/IEC/IEEE 29119-4 (Testing technology) and functional safety (IEC 61508, ISO 26262)) have been analyzed. In addition, discussions on various themes such as data quality have started at SC42.

Transparency

The High-Level Expert Group on Artificial Intelligence (AI-HLEG) compiled required conditions for transparency in EU, and they are considered to have a global impact mainly on the EU member states. A checklist to ensure transparency was presented in April 2019 and the conditions are validated through actual validation tests with businesses (the 2nd version is expected to be issued at the end of 2019).

On the other hand, IEEE is currently examining IEEE P7001 (transparency of autonomous systems) and it may have a certain level of influence over future standardization in terms of the definitions of terms and concept. The six transparency levels (0～5) have been defined for the five types of stakeholders (users, accident investigation committee, etc.) (a higher number does not always mean that it is stricter). The certification of its compatibility is demonstrated through the pilot validation project called SCPAIS.

ISO includes terms and concepts related to transparency in the document TR24028 at WG3

(trustworthiness) of ISO/IEC JTC 1/SC 42.

Fairness (bias)

EU AI high-level expert group (AI-HLEG) compiles high-level principles for problems of ELSI of AI including bias. The above IEEE P7003 (algorithmic bias) specifies a method to identify “negative bias” (both legally-prohibited discrimination based on race or gender and non-legal discrimination) in the development of algorithms and keep bias within the acceptable range in the lifecycle from system planning to operation. A demonstration experiment of this method is under way in the pilot project (ECPAIS (Ethics Certification Program for Autonomous and Intelligent Systems)) to validate its conformity.

ISO/IEC JTC 1/SC 42/WG 3 (trustworthiness) is also preparing documents such as TR 24028 (Overview of trustworthiness in Artificial Intelligence), TR 24027 (Bias in AI systems and AI aided decision making).

Other ethical qualities

Privacy and nudge (induction of behaviors) are examined in the IEEE P7000 series, and ISO/IEC JTC1/SC42 examines governance and other themes.

9 (informative) Analytical information

This Chapter sorts out a process of analysis to draw out the characteristic axes of internal qualities listed mainly in Section 1.7 and Chapter 6 as reference information.

The content of this Chapter is informative.

9.1 Analysis of characteristic axes of internal qualities with respect to risk avoidance

First, a quality deterioration mode was identified with regard to external quality axes of risk avoidance. This quality deterioration occurs when “individual inference results of machine learning components (to describe it in an extreme manner, vector values of neural-network results) is not correct, favorable or desirable”. Thus, we analyzed a possibility that a machine learning component gives [undesirable] answers based on a concept similar to Fault Tree Analysis (FTA) using an abstract and binary tree failure mode on the assumption of abstract machine learning components. The analysis results are shown in Figure 17.

This analysis aims to comprehensively decompose the causes of failure modes. It should be noted that, when any misjudgment was made, it might be impossible to identify the cause without so-called oracle perspective. On the other hand, it is possible to reduce a possibility of all failure modes by taking measures for all causes of failure without oracle perspective. Of course, it is impossible to perfectly realize each paragraph. Moreover, errors included in training input data and various gaps such as mathematical issues in super-multidimensional space data are ignored intentionally in this analysis. However, if these items are focused on as a direction of the overall quality improvement process, it can make a sufficient contribution to the improvement of quality despite gaps.

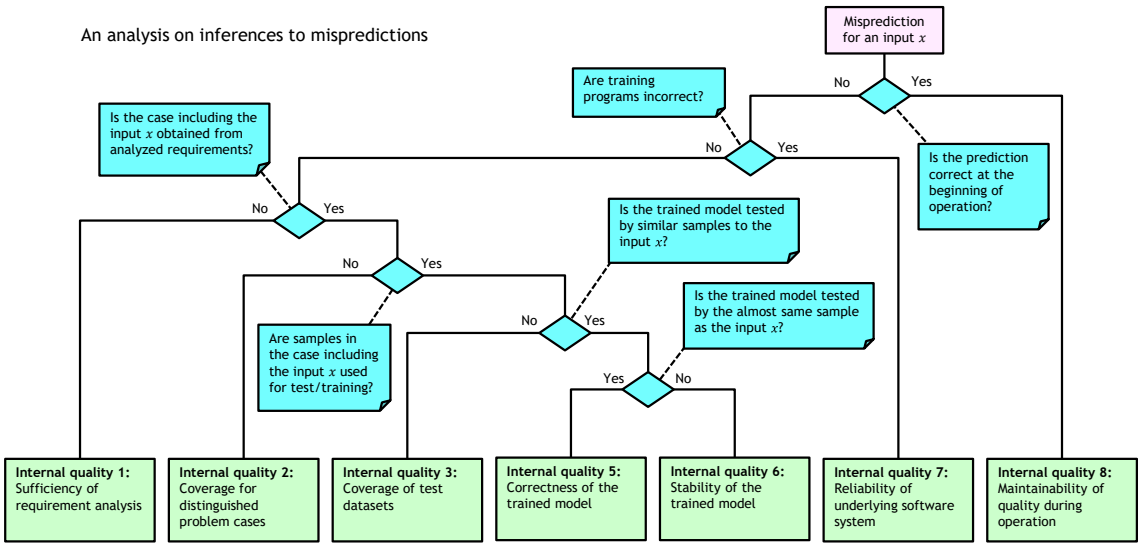

Figure 17: Example of analysis of failure mode with respect to machine learning

9.2 Analysis of quality management axes with respect to AI performance

AI performance was analyzed in a similar way based on the internal quality management axes with respect to “risk avoidance” mentioned in the previous section.

Strictly speaking, AI performance also deteriorates, when “individual inference results of machine learning components (to describe in an extreme manner, vector values of neural-network results) is not correct, favorable or desirable”. A difference from risk avoidance is a difference in overall evaluation functions caused by different weighting of individual cases. Therefore, the same internal quality characteristic axes can be basically used as they are.

However, risk avoidance focuses on sufficient assignment of training data as measures for each anticipated risk case and does not take into account the balance of overall training data intentionally. This is because sufficient learning cannot be achieved by uniformly sampling training data especially from serious risk cases, although their frequency of occurrence is very low. From the viewpoint of AI performance, however, it is widely known that this type of “intentionally-biased reinforced learning data” may cause deterioration of overall performance. That is why we added “Uniformity of training datasets” listed in 6.4 as an internal quality axis which mainly envisions AI performance. The results are eight internal characteristics listed in Chapter 6.

9.3 Examination of quality management axes with respect to fairness

Fairness is in the initial examination stage socially and academically. At this moment, sufficient knowledge has not been acquired as to specific methods to improve the quality.

Therefore, we present the following five items considered as “kinds of fairness” from the viewpoint of checking and inspecting the quality. This paragraph will be reviewed as needed based on the future progress of research and trends of international standards.

- Results are not discriminatory:
  - A judgment process (trained learning model) does not include discriminatory elements
  - The results distributions are statistically equal
- Building processes are not discriminatory:
  - Training data does not include discriminatory elements
  - Training data is statistically equal
  - Training data corresponds to the real world (deemed to be fair) as distribution.

Moreover, the requirements analyses on risk avoidance and AI performance are used also for analyzing “the targets of fairness” such as gender and race, so that they come down to each item of the existing internal qualities.

10 Tables and figures

The content of this Chapter is informative.

10.1 Tables of relations between external quality characteristics and internal quality characteristics

The numbers and symbols in the tables refer to the levels of check items listed in "Requirements for quality levels" of applicable sections. The symbol "+" means that additional examinations will be made in the future. The bolded items mean that they require special attention.

AISL	0	0.1	0.2	1	2	3	4
6.1 Sufficiency of problem domain analysis		1	2	3	+	+	+
6.2 Coverage for distinguished problem cases		1	2	3	+	+	+
6.3 Coverage of test datasets		1	2	3	+	+	+
6.4 Uniformity of training datasets		S1	S2	S2	+	+	+
6.5 Correctness of the trained model		1	2	3	+	+	+
6.6 Robustness of the trained model		1	2	3	+	+	+
6.7 Reliability of underlying software system		1	2	3	+	+	+
6.8 Maintainability of quality during operation		1	2	3	+	+	+

AIPL/AIFL	PL 0	PL 1	PL 2	FL 0	FL 1	FL 2
6.1 Sufficiency of problem domain analysis		1	2		1	2
6.2 Coverage for distinguished problem cases		1	2		1	2
6.3 Coverage of test datasets		1	2		1	2
6.4 Uniformity of training datasets		E1	E2		E2	E2
6.5 Correctness of the trained model		1	2		1	2
6.6 Robustness of the trained model		1	2		1	2
6.7 Reliability of underlying of software system		1	2		1	2
6.8 Maintainability of quality during operation		1	2		2	3

Artificial Intelligence Research Center (AIRC)/
Cyber Physical Security Research Center (CPSEC),
National Institute of Advanced Industrial Science and Technology (AIST)

List of members of the Committee for machine learning quality management

Shin Nakajima	National Institute of Informatics (chair)
Yoshiko Seo	National Institute of Advanced Industrial Science and Technology (vice-chair)
Takashi Egawa	NEC Corporation
Shintaro Fukushima	Toyota Motor Corporation
Chinami Hamatani	Ad-Sol Nissin Corporation
Kenichi Kobayashi	Fujitsu Laboratories Limited
Hiroshi Kuwajima	Denso Corporation
Yosuke Motohashi	NEC Corporation
Yusei Nakashima	TechMatrix Corporation
Hideto Ogawa	Hitachi, Ltd.
Yutaka Oiwa	National Institute of Advanced Industrial Science and Technology
Tamao Okamoto	Panasonic Corporation
Naoto Sato	Hitachi, Ltd.
Tomomichi Suzuki	Tokyo University of Science
Satoshi Tsuchiya	Fujitsu Limited

List of authors of the Machine Learning Quality Management Guideline

- Japanese edition:
  - Overall structure, writing and editing: Yutaka OIWA (AIST)
  - Section 5.2: Chinami Hamatani (Ad-Sol Nisshin Corporation)
  - Section 7.5: Shin Nakajima (National Institute of Informatics)
  - Section 7.5.2: Yoshinao Isobe (AIST)
  - Section 7.8: Kenichi Kobayashi and Yoshihiro Okawa (Fujitsu Laboratories Limited)
  - Section 8.2: Contribution of Egawa Takashi (NEC Corporation)
  - Supervision: Members of
    - Committee for machine learning quality management
    - Guideline Taskforce
- English edition: members of the Guideline Taskforce

List of members of the Guideline Taskforce

Yutaka Oiwa	National Institute of Advanced Industrial Science and Technology (leader)
Takashi Egawa	NEC Corporation
Shin Nakajima	National Institute of Informatics
Shintaro Fukushima	Toyota Motor Corporation
Chinami Hamatani	Ad-Sol Nissin Corporation
Yoshinao Isobe	National Institute of Advanced Industrial Science and Technology
Yusuke Kawamoto	National Institute of Advanced Industrial Science and Technology
Kenichi Kobayashi	Fujitsu Laboratories Ltd.
Hiroshi Kuwajima	Denso Corporation
Yosuke Motohashi	NEC Corporation
Kazumasa Miyake	Sumitomo Electric Industries, Ltd.
Yusei Nakashima	Techmatrix Corporation
Tamao Okamoto	Panasonic Corporation
Yoshiki Seo	National Institute of Advanced Industrial Science and Technology
Satoshi Tsuchiya	Fujitsu Limited