AQI Automotive Quality Institute

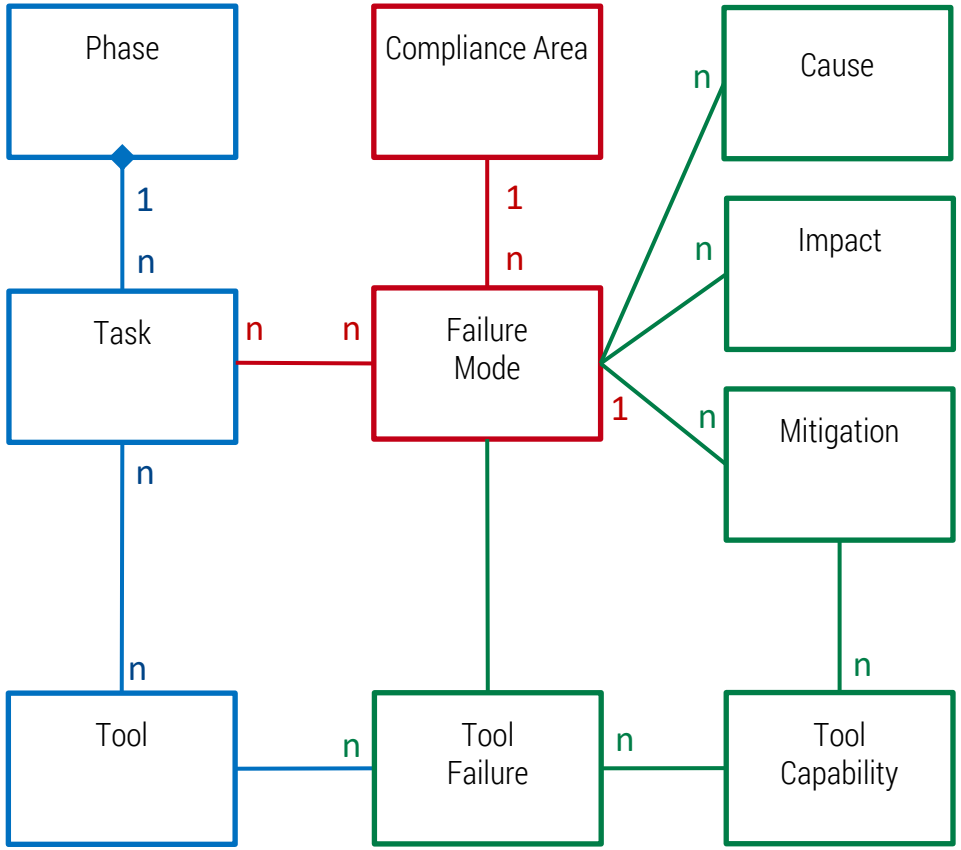
Q in AI: Requirements for tools and processes for assurance

Instructions for qualifying tools

Dr. Björn Schünemann (bjoern.schuenemann@aqigmbh.de)
Dr. Jürgen Großmann (juergen.grossmann@fokus.fraunhofer.de)

Relationship between concepts

Information model: Project procedure and terminology used



- Systematic listing of **life cycle phases (phase)** and **AI tasks (task)** as well as exemplary **tools (tool).**
- Identification of the **failure mode (failure mode)** along the **AI tasks** for each **compliance area (compliance area).**
- Identification of **causes, effects (impact)** and possible countermeasures (mitigation) for each **failure mode.**
- Identification of specific tool failures **(tool failure)** and **required tool capabilities (tool capability)** for a **failure mode.**

Phase: Phase from the AI life cycle

Task: Task during the AI life cycle that is supported by tools .

Tool: Tool that supports/executes a task.

Compliance Area: Area of compliance (functional safety, data protection, AI regulation).

Failure Mode: Hazard in the event of non-compliance.

Tool Failure: Tool error as the cause of an error condition.

Cause: Causes of the risk.

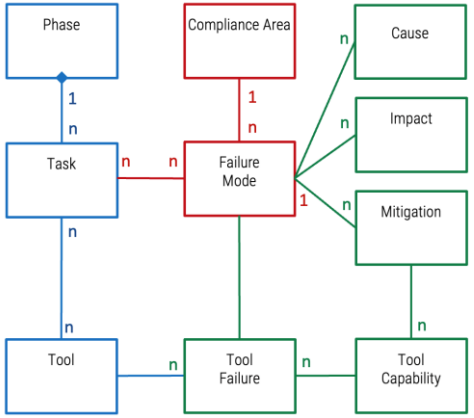
Impact: Effects of an error condition.

Mitigation: Measures to reduce the risk.

Tool Capability: Capabilities required of a tool to address the risk.

Systematic presentation of the developed content

Result is an interactive Excel sheet



Contents of the Excel table are compiled according to the terminology used

Compliance Area	Phase	Task	Failure Mode	Impact	Severity	Causes	Mitigation	Occurrence	Detectability	RPN	Potential Tool Failure	Recommended Tool Capability
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Lack of Real-Time Processing	Trained models and supporting frameworks are unable to deliver low-latency, real-time inference performance required for critical applications such as autonomous systems.		Model architectures are too large or complex for target real-time environments. Frameworks do not leverage hardware acceleration efficiently (e.g., GPU, TPU, Edge devices). Pipeline overhead (e.g., preprocessing, postprocessing) adds excessive latency during inference.	Apply model optimization techniques such as pruning, quantization, or knowledge distillation. Ensure deployment-ready models are benchmarked and optimized for specific hardware accelerators. Optimize full pipeline latency by batching operations and reducing unnecessary transformations.				May fail to guarantee deterministic training execution and runtime validation, resulting in trained models that do not meet real-time safety-critical performance requirements.	Framework support for model optimization (e.g., TensorRT, ONNX Runtime optimizations). Support for hardware-specific runtime optimizations and deployment targeting (e.g., EdgeTPU, CUDA kernels). Framework support for pipeline fusion, minimal preprocessing APIs, and asynchronous inference.
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Inadequate Error Handling & Logging	Frameworks and models lack robust error detection, recovery, and structured logging capabilities, making root cause analysis and system recovery difficult.		Training and inference exceptions are not properly caught or logged. Losses, gradients, or other training dynamics are not systematically monitored for anomalies. Training and deployment pipelines lack comprehensive logging and metadata tracking.	Integrate structured exception handling and mandatory logging for all critical training and inference operations. Implement runtime monitors for numerical instabilities (e.g., NaNs, divergence) and alert on thresholds. Embed pipeline-wide metadata capture and systematic experiment logging into the training framework.				May fail to provide comprehensive error logging and runtime monitoring, resulting in trained models with limited traceability for failure analysis in safety-critical systems.	Framework support for structured error reporting, custom callbacks, and logging integration. Built-in hooks for anomaly detection during training and validation (gradient checkers, divergence detectors). ML lifecycle management tools (e.g., MLflow, Weights & Biases) integration for metadata and error tracking.
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Model Instability & Poor Generalization	Inconsistent training leading to unreliable AI behaviour in critical applications.		Lack of deterministic/reproducible training settings due to parallel computation and floating-point arithmetic. Limited built-in validation or monitoring. No built-in numerical stability mechanisms.	Implement deterministic training pipelines with seed control and precision configuration to ensure repeatability. Integrate continuous performance monitoring during training with automatic validation checkpoints. Use frameworks that incorporate gradient clipping, normalization, and regularization for improved numerical robustness.				May fail to enforce deterministic training procedures and runtime validation, resulting in trained models with unpredictable behavior and poor generalization in safety-critical applications.	Explicit deterministic training support with reproducibility settings and controlled parallelism. Built-in validation dashboards and metric tracking APIs. Numerical stability modules with configurable regularization options.

Risk-based tool qualification (details)

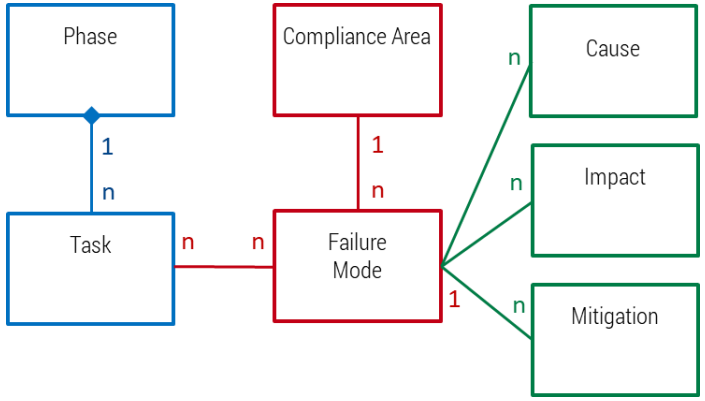
Two-stage process for identifying task-related risks and qualifying tools based on the risks

AQI Automotive Quality Institute

1. Identification of risks in development tasks (according to FMEA)

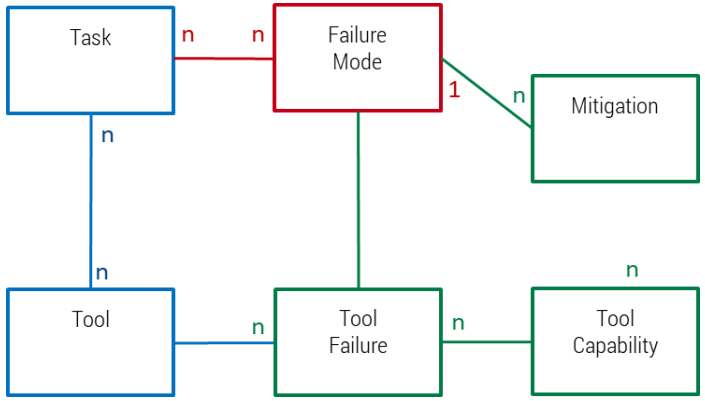
RPN = Severity * Occurrence * Detection

1.1 Selection of relevant tasks phases	1.2 Selection of area of compliance and applicable failure modes	1.3 Calculation of the RPN for selected failure modes in a task



2. Identification of qualification requirements for AI tools

2.1 Assessment of the significance of the tool for an error (Tool Impact)	2.2 Evaluation of tool error detection and determination of the tool confidence level (TCL) for each conformity area	2.3 Qualification recommendation based on RPN and TCL



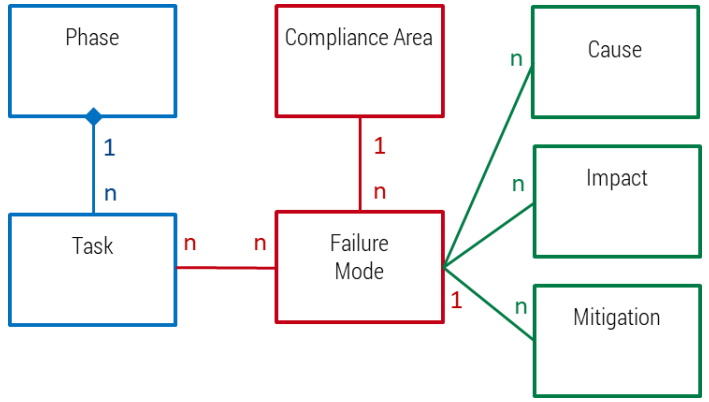
Risk-based tool qualification (details)

Two-stage process for identifying task-related risks and qualifying tools based on the risks

1. Identification of risks in development tasks (according to FMEA)

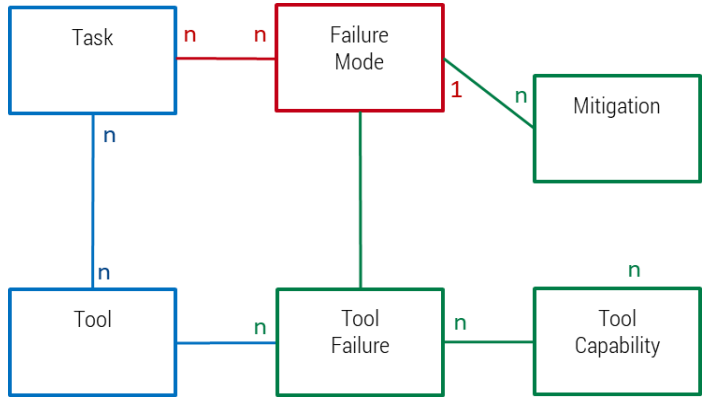
RPN = Severity * Occurrence * Detection

1.1 Selection of relevant tasks phases	1.2 Selection of area of compliance and applicable failure modes	1.3 Calculation of the RPN for selected failure modes in a task



2. Identification of qualification requirements for AI tools

2.1 Assessment of the significance of the tool for an error (Tool Impact)	2.2 Evaluation of tool error detection and determination of the tool confidence level (TCL) for each conformity area	2.3 Qualification recommendation based on RPN and TCL



AQI Automotive Quality Institute

1. Identification of risks in development tasks

1.1. Selection of relevant tasks phases

FMEA Excel: selection of relevant tasks and phases

Automotive Quality Institute

Not every ML task is relevant

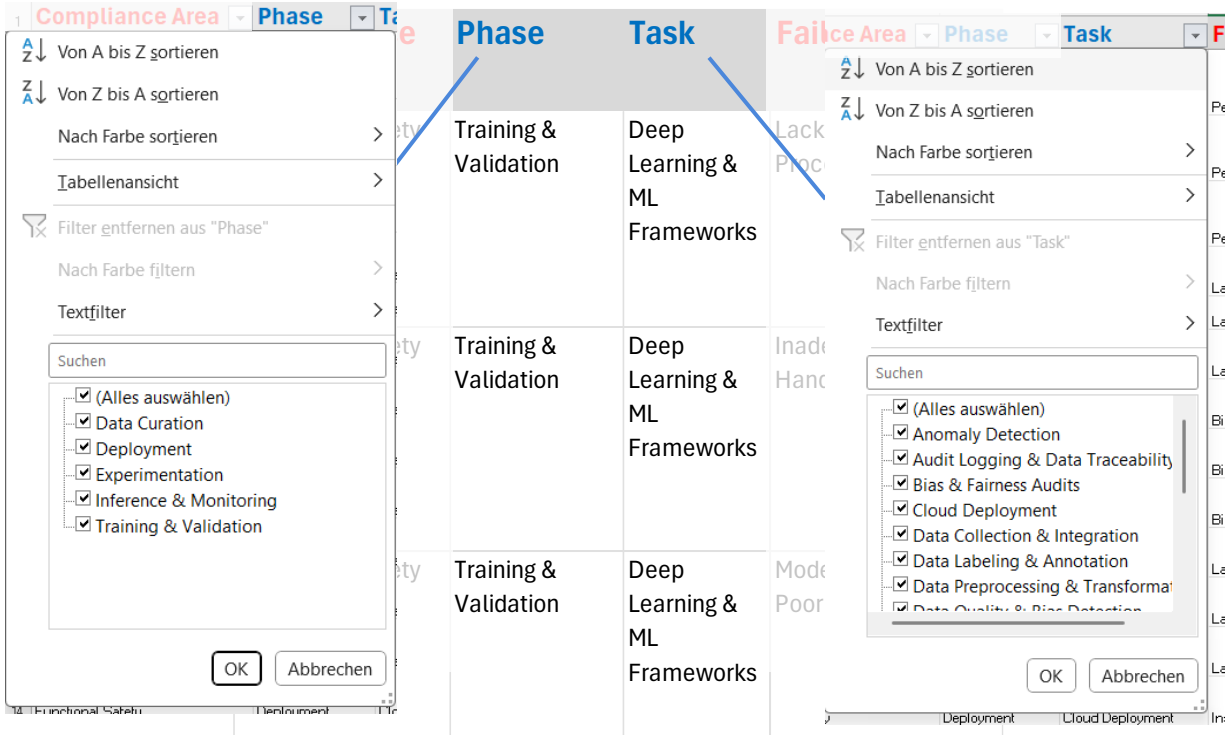
Initial situation: Is this ML task part of the development system?

Selection — Relevant phases (list) and relevant tasks (list)

Scale — Boolean (Yes/No)

Purpose — Limits the desired phases and tasks.

Task:
Select the relevant ML task from the Excel table.



AQI Automotive Quality Institute

1. Identification of risks in development tasks

1.2 Selection of area of compliance and applicable failure modes

FMEA Excel: selection of failure modes

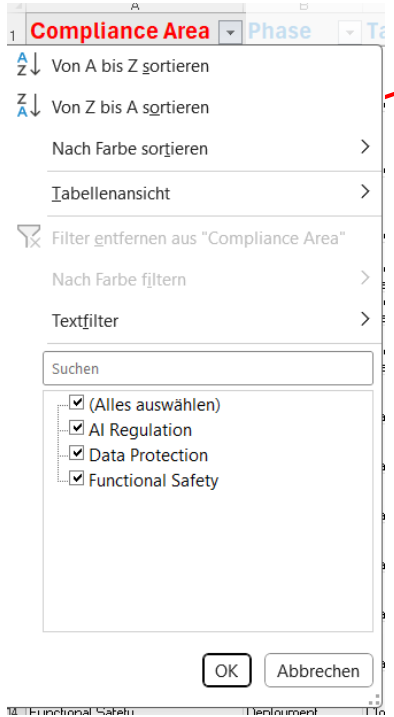
Automotive Quality Institute

Not every ML task is relevant

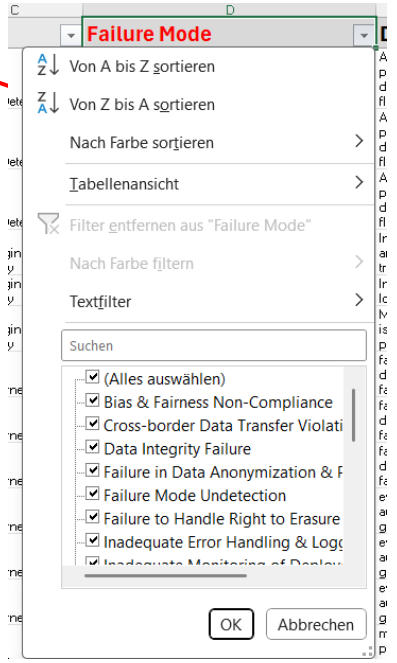
Initial situation: Is this ML task part of the development system?

	Selection	Relevant areas of compliance (list)
	Scale	Boolean (Yes/No)
	Purpose	Limits the failure modes to the desired areas of compliance

Task:
Select the relevant areas of compliance from the Excel table.



Compliance Area	Phase	Task	Failure Mode
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Lack of Real-Time Processing
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Inadequate Error Handling & Logging
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Model Instability & Poor Generalization



AQI Automotive Quality Institute

1. Identification of risks in development tasks

1.3 Calculation of the RPN for selected failure modes in a task

FMEA Excel: evaluation of failure modes

AQI | Automotive Quality Institute

Risk priority indicator

RPI = Severity * Occurrence * Detection

Task:
Calculation of the **risk priority number (RPI)** for each fault mode.

Input	Assessment of severity	#(rating from 1 to 10)
Input	Assessment of the probability of occurrence	#(rating from 1 to 10)
Input	Assessment of recognisability	#(rating from 1 to 10)

Compliance Area	Phase	Task	Failure Mode	Impact	Severity	Causes	Mitigation	Occurrence	Detectability	RPN
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Lack of Real-Time Processing	Trained models and supporting frameworks are unable to deliver low-latency, real-time inference performance required for critical applications such as autonomous systems.		Model architectures are too large or complex for target real-time environments. Frameworks do not leverage hardware acceleration efficiently (e.g., GPU, TPU, Edge devices). Pipeline overhead (e.g., preprocessing, postprocessing) adds excessive latency during inference.	Apply model optimization techniques such as pruning, quantization, or knowledge distillation. Ensure deployment-ready models are benchmarked and optimized for specific hardware accelerators. Optimize full pipeline latency by batching operations and reducing unnecessary transformations.			
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Inadequate Error Handling & Logging	Frameworks and models lack robust error detection, recovery, and structured logging capabilities, making root cause analysis and system recovery difficult.		Training and inference exceptions are not properly caught or logged. Losses, gradients, or other training dynamics are not systematically monitored for anomalies. Training and deployment pipelines lack comprehensive logging and metadata tracking.	Integrate structured exception handling and mandatory logging for all critical training and inference operations. Implement runtime monitors for numerical instabilities (e.g., NaNs, divergence) and alert on thresholds. Embed pipeline-wide metadata capture and systematic experiment logging into the training framework.			
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Model Instability & Poor Generalization	Inconsistent training leading to unreliable AI behaviour in critical applications.		Lack of deterministic/reproducible training settings due to parallel computation and floating-point arithmetic. Limited built-in validation or monitoring. No built-in numerical stability mechanisms.	Implement deterministic training pipelines with seed control and precision configuration to ensure repeatability. Integrate continuous performance monitoring during training with automatic validation checkpoints. Use frameworks that incorporate gradient clipping, normalization, and regularization for improved numerical robustness.			

FMEA Excel: evaluation of failure modes

Automotive Quality Institute

Significance of the error sequence (severity)

RPN = Severity * Occurrence * Detection

Input	Assessment of severity
Scale	1 – 3: 4 – 6; 7 - 10

Task:
Assess the severity level for the failure modes assigned to the selected ML tasks in the Excel table.

Compliance Area	Phase	Task	Failure Mode	Impact	Severity
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Lack of Real-Time Processing	Trained models and supporting frameworks are unable to deliver low-latency, real-time inference performance required for critical applications such as autonomous systems.	2
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Inadequate Error Handling & Logging	Frameworks and models lack robust error detection, recovery, and structured logging capabilities, making root cause analysis and system recovery difficult.	6
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Model Instability & Poor Generalization	Inconsistent training leading to unreliable AI behaviour in critical applications.	3

FMEA Excel: evaluation of failure modes

Automotive Quality Institute

Probability of occurrence of the cause of the error (Occurence)

RPN = Severity * Occurrence * Detection

Input	Assessment of the probability of occurrence
Scale	1 – 3: 4 – 6; 7 - 10

Task:
Evaluate the probability of occurrence for the failure modes assigned to the selected ML tasks in the Excel table.

Causes	Mitigations	Occurrence	Detectability	RPN
Model architectures are too large or complex for target real-time environments. Frameworks do not leverage hardware acceleration efficiently (e.g., GPU, TPU, Edge devices). Pipeline overhead (e.g., preprocessing, postprocessing) adds excessive latency during inference.	Apply model optimization techniques such as pruning, quantization, or knowledge distillation. Ensure deployment-ready models are benchmarked and optimized for specific hardware accelerators. Optimize full pipeline latency by batching operations and reducing unnecessary transformations.	2		
Training and inference exceptions are not properly caught or logged. Losses, gradients, or other training dynamics are not systematically monitored for anomalies. Training and deployment pipelines lack comprehensive logging and metadata tracking.	Integrate structured exception handling and mandatory logging for all critical training and inference operations. Implement runtime monitors for numerical instabilities (e.g., NaNs, divergence) and alert on thresholds. Embed pipeline-wide metadata capture and systematic experiment logging into the training framework.	6		
Lack of deterministic/reproducible training settings due to parallel computation and floating-point arithmetic. Limited built-in validation or monitoring. No built-in numerical stability mechanisms.	Implement deterministic training pipelines with seed control and precision configuration to ensure repeatability. Integrate continuous performance monitoring during training with automatic validation checkpoints. Use frameworks that incorporate gradient clipping, normalization, and regularization for improved numerical robustness.	3		

FMEA Excel: evaluation of failure modes

Automotive Quality Institute

Probability of detecting the error or its cause (Detectability)

RPN = Severity * Occurrence * Detection

Input	Assessment of recognisability
Scale	1 – 3: 4 – 6; 7 - 10

Task:
Assess the **detectability** of the failure modes assigned to the selected ML tasks in the Excel table.

Causes	Mitigations	Occurrence	Detectability	RPN
Model architectures are too large or complex for target real-time environments. Frameworks do not leverage hardware acceleration efficiently (e.g., GPU, TPU, Edge devices). Pipeline overhead (e.g., preprocessing, postprocessing) adds excessive latency during inference.	Apply model optimization techniques such as pruning, quantization, or knowledge distillation. Ensure deployment-ready models are benchmarked and optimized for specific hardware accelerators. Optimize full pipeline latency by batching operations and reducing unnecessary transformations.	2	5	
Training and inference exceptions are not properly caught or logged. Losses, gradients, or other training dynamics are not systematically monitored for anomalies. Training and deployment pipelines lack comprehensive logging and metadata tracking.	Integrate structured exception handling and mandatory logging for all critical training and inference operations. Implement runtime monitors for numerical instabilities (e.g., NaNs, divergence) and alert on thresholds. Embed pipeline-wide metadata capture and systematic experiment logging into the training framework.	6	6	
Lack of deterministic/reproducible training settings due to parallel computation and floating-point arithmetic. Limited built-in validation or monitoring. No built-in numerical stability mechanisms.	Implement deterministic training pipelines with seed control and precision configuration to ensure repeatability. Integrate continuous performance monitoring during training with automatic validation checkpoints. Use frameworks that incorporate gradient clipping, normalization, and regularization for improved numerical robustness.	3	3	

FMEA Excel: evaluation of failure modes

AQI | Automotive Quality Institute

Risk priority indicator

RPN = Severity * Occurrence * Detection

Task:
Calculation of the risk priority number (RPI) for each fault mode.

Scale 1 – 29; 30 – 299; 300 -1000

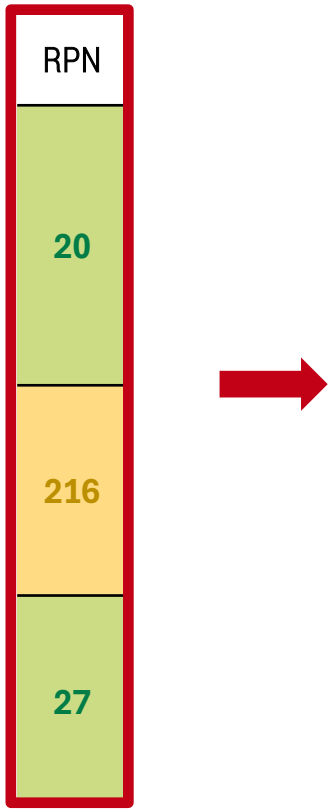
Compliance Area	Phase	Task	Failure Mode	Impact	Severity	Causes	Mitigations	Occurrence	Detectability	RPN
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Lack of Real-Time Processing	Trained models and supporting frameworks are unable to deliver low-latency, real-time inference performance required for critical applications such as autonomous systems.	2	Model architectures are too large or complex for target real-time environments. Frameworks do not leverage hardware acceleration efficiently (e.g., GPU, TPU, Edge devices). Pipeline overhead (e.g., preprocessing, postprocessing) adds excessive latency during inference.	Apply model optimization techniques such as pruning, quantization, or knowledge distillation. Ensure deployment-ready models are benchmarked and optimized for specific hardware accelerators. Optimize full pipeline latency by batching operations and reducing unnecessary transformations.	2	5	20
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Inadequate Error Handling & Logging	Frameworks and models lack robust error detection, recovery, and structured logging capabilities, making root cause analysis and system recovery difficult.	6	Training and inference exceptions are not properly caught or logged. Losses, gradients, or other training dynamics are not systematically monitored for anomalies. Training and deployment pipelines lack comprehensive logging and metadata tracking.	Integrate structured exception handling and mandatory logging for all critical training and inference operations. Implement runtime monitors for numerical instabilities (e.g., NaNs, divergence) and alert on thresholds. Embed pipeline-wide metadata capture and systematic experiment logging into the training framework.	6	6	216
Functional Safety	Training & Validation	Deep Learning & ML Frameworks	Model Instability & Poor Generalization	Inconsistent training leading to unreliable AI behaviour in critical applications.	3	Lack of deterministic/reproducible training settings due to parallel computation and floating-point arithmetic. Limited built-in validation or monitoring. No built-in numerical stability mechanisms.	Implement deterministic training pipelines with seed control and precision configuration to ensure repeatability. Integrate continuous performance monitoring during training with automatic validation checkpoints. Use frameworks that incorporate gradient clipping, normalization, and regularization for improved numerical robustness.	3	3	27

FMEA Excel: evaluation of failure modes

AQI Automotive Quality Institute

Risk priority indicator

RPN = Severity * Occurrence * Detection



Evaluation of the **risk priority number** (RPN):

- Low RPN values (RPN 1–30) indicate low criticality and no need for action
- Medium RPN values (RPN 31–299) indicate medium criticality and thus a need for action
- High values (RPN 300–1000) indicate high criticality and a greater need for action.

→ The determination of risks (RPN) in the development tasks (according to FMEA) focuses on the determination **per use case**

Risk-based tool qualification (details)

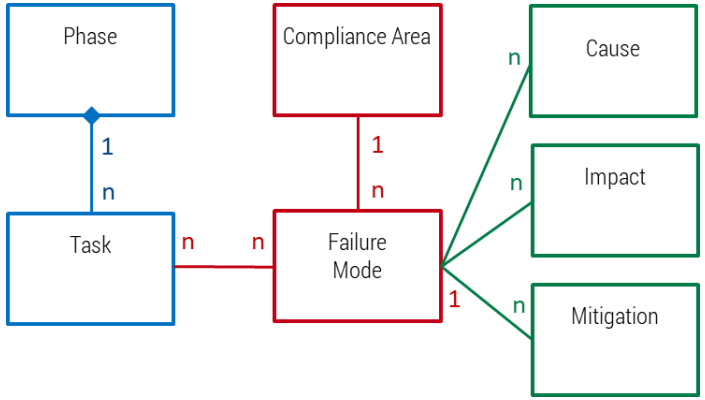
Two-stage process for identifying task-related risks and qualifying tools based on the risks

AQI | Automotive Quality Institute

1. Identification of risks in development tasks (according to FMEA)

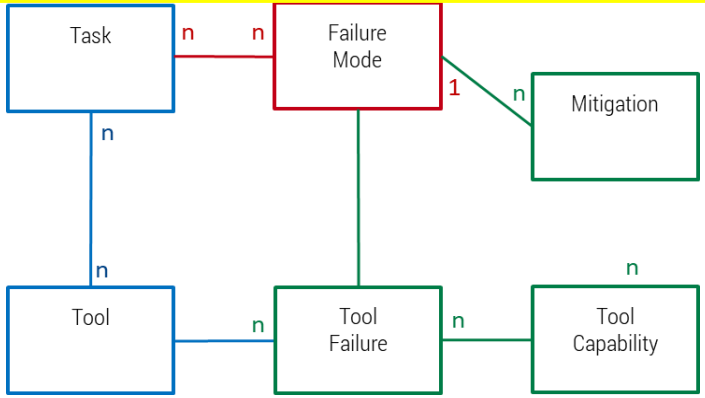
RPN = Severity * Occurrence * Detection

1.1 Selection of relevant tasks phases	1.2 Selection of area of compliance and applicable failure modes	1.3 Calculation of the RPN for selected failure modes in a task



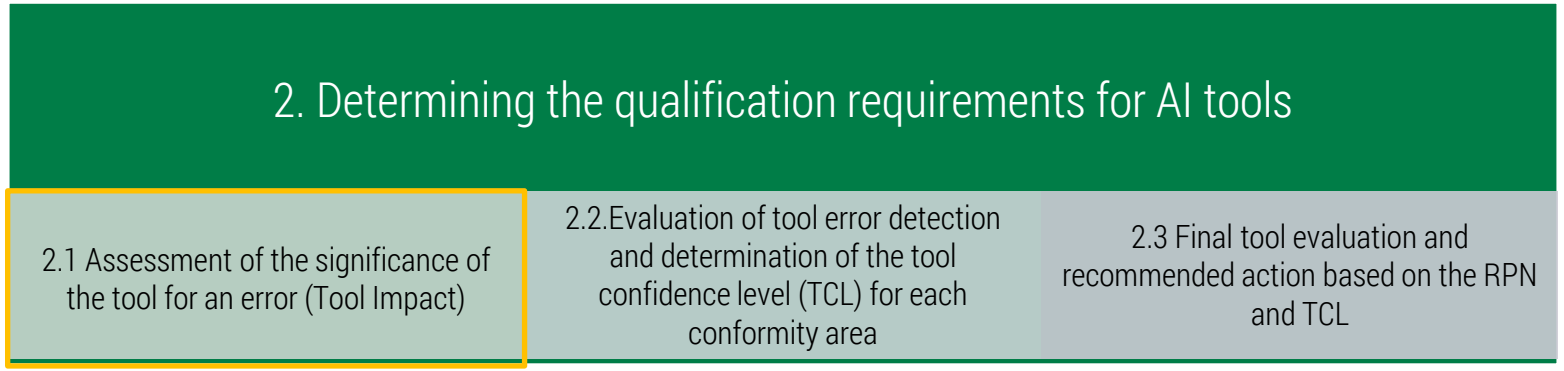
2. Identification of qualification requirements for AI tools

2.1 Assessment of the significance of the tool for an error (Tool Impact)	2.2 Evaluation of tool error detection and determination of the tool confidence level (TCL) for each conformity area	2.3 Qualification recommendation based on RPN and TCL



2. Determining the qualification requirements for AI tools

2.1 Assessment of the significance of the tool for an error (Tool Impact)

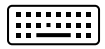


FMEA Excel: impact of tool error

AQI Automotive Quality Institute

Impact of the tool

Starting point: Does a potential error in the tool affect the functionality of safety-related systems?

	**Input**	Tool Impact Value - Assessment of the existence of safety-related effects
	**Scale**	1 (yes) , 2 (no)

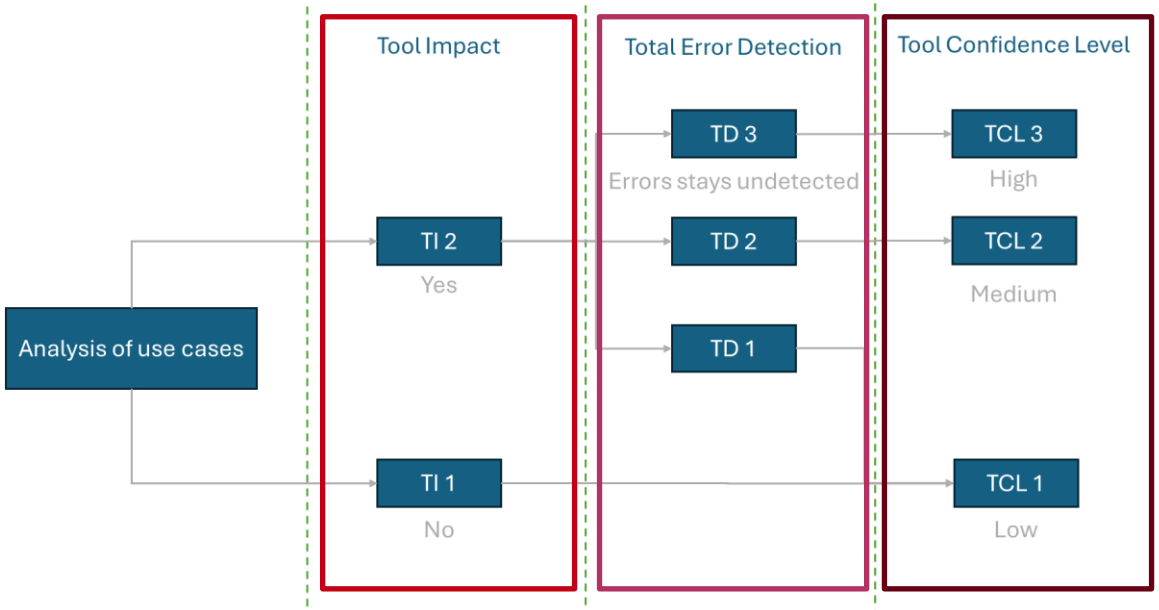
Potential Tool Error	Recommended Tool Capabilities	Tool Impact Value	Tool Error Detection	Tool Confidence Value
May fail to guarantee deterministic training execution and runtime validation, resulting in trained models that do not meet real-time safety-critical performance requirements.	Framework support for model optimization (e.g., TensorRT, ONNX Runtime optimizations). Support for hardware-specific runtime optimizations and deployment targeting (e.g., EdgeTPU, CUDA kernels). Framework support for pipeline fusion, minimal preprocessing APIs, and asynchronous inference.	1		
May fail to provide comprehensive error logging and runtime monitoring, resulting in trained models with limited traceability for failure analysis in safety-critical systems.	Framework support for structured error reporting, custom callbacks, and logging integration. Built-in hooks for anomaly detection during training and validation (gradient checkers, divergence detectors). ML lifecycle management tools (e.g., MLflow, Weights & Biases) integration for metadata and error tracking.	2		
May fail to enforce deterministic training procedures and runtime validation, resulting in trained models with unpredictable behavior and poor generalization in safety-critical applications.	Explicit deterministic training support with reproducibility settings and controlled parallelism. Built-in validation dashboards and metric tracking APIs. Numerical stability modules with configurable regularization options.	2		

Background: impact of tool error

Impact of the tool

The figure was developed based on section 11.4.5.2 of **ISO 26262:2018** and shows the process for determining the **Tool Confidence Level (TCL)** based on:

- **Tool Impact (TI)**, which assesses whether a tool error could be safety-relevant,
- **Tool Error Detection (TD)**, which assesses the probability that an error will be detected, and results in the
- **Tool Confidence Level (TCL)**, which is the required level of confidence in the tool (TCL1 to TCL3).

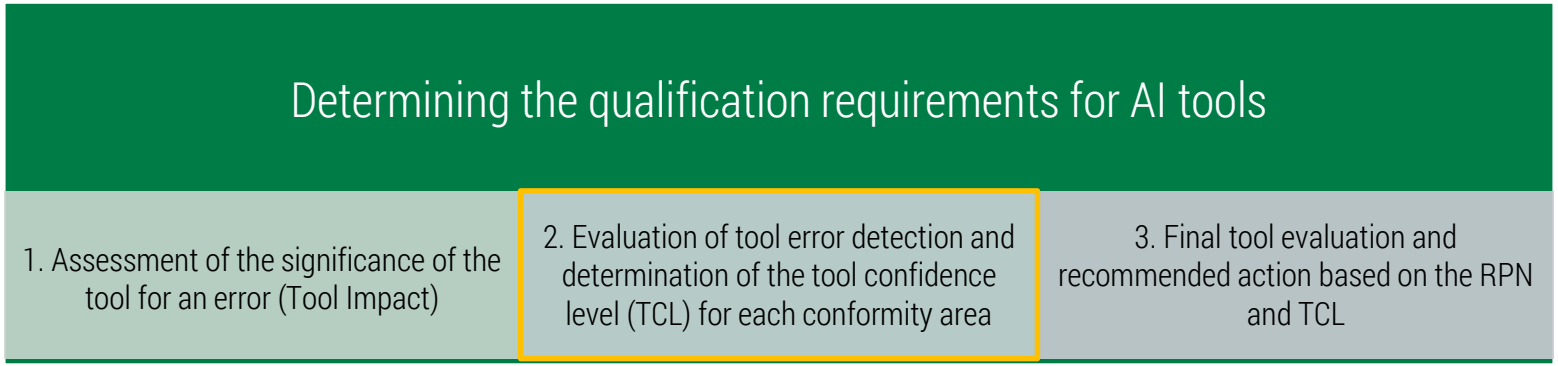


Source: Own representation based on ISO 26262:2018 Section 11.4.5.2

AQI Automotive Quality Institute

2. Determining the qualification requirements for AI tools

2.2. Evaluation of tool error detection and TCL calculation

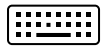


FMEA Excel: detectability of tool error

AQI | Automotive Quality Institute

Detectability of tool errors

Starting point: How well can errors caused by the tool be detected in the further development process?



Input Tool Error Detection – Assessment of the existence of safety-related effects

Scale Error detection (1 = detected, 2 = may remain undetected, 3 = remain undetected)

Potential Tool Error	Testable Tool Capabilities	Tool Impact Value	Tool Error Detection	Tool Confidence Value
May fail to guarantee deterministic training execution and runtime validation, resulting in trained models that do not meet real-time safety-critical performance requirements.	Framework support for model optimization (e.g., TensorRT, ONNX Runtime optimizations). Support for hardware-specific runtime optimizations and deployment targeting (e.g., EdgeTPU, CUDA kernels). Framework support for pipeline fusion, minimal preprocessing APIs, and asynchronous inference.	1	1	
May fail to provide comprehensive error logging and runtime monitoring, resulting in trained models with limited traceability for failure analysis in safety-critical systems.	Framework support for structured error reporting, custom callbacks, and logging integration. Built-in hooks for anomaly detection during training and validation (gradient checkers, divergence detectors). ML lifecycle management tools (e.g., MLflow, Weights & Biases) integration for metadata and error tracking.	2	2	
May fail to enforce deterministic training procedures and runtime validation, resulting in trained models with unpredictable behavior and poor generalization in safety-critical applications.	Explicit deterministic training support with reproducibility settings and controlled parallelism. Built-in validation dashboards and metric tracking APIs. Numerical stability modules with configurable regularization options.	2	3	

Background: calculation of TCL

AQI Automotive Quality Institute

Need for trust in tool usage

Starting point: How high must the level of trust in the tool be (or is tool qualification necessary)?

Table 3 — Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

Source: ISO 26262:2018, Section. 11.4.5.4

Aspect	Picture	Description
Target	***Assignment of the appropriate Tool Confidence Level (TCL)*** *based on Tool Impact (TI) and Tool Error Detection (TD)*	*Determine* ***how high the level of trust*** *in the tool must be (or whether tool qualification is necessary)*
TCL1 (low)	▪ Tool has no safety-related influence (TI1) or ▪ Errors are reliably detected (TI2 + TD1) → **No tool qualification necessary**	▪ **Low significance** for product quality ▪ No tool qualification necessary → Confidence in tool behaviour from an ISO 26262 perspective **not critical**
TCL2 (medium)	▪ Tool has safety-related influence (TI2) ▪ Error **not reliably detectable (TD2)**	▪ Tool **important for product quality** ▪ Tool qualification required ▪ Requirements depend on ASIL level
TCL3 (high)	▪ Requirements depend on ASIL level (TI2) ▪ Errors **remain undetected (TD3)**	▪ **High importance** for product quality ▪ **Tool qualification absolutely necessary**, high requirements ▪ Differences between TCL2 and TCL3 are methodologically relevant but not dramatic (depending on ASIL)

Background: calculation of TCL

AQI Automotive Quality Institute

Need for trust in tool usage

Starting point: How high must the level of trust in the tool be (or is tool qualification necessary)?

Table 3 — Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

Source: ISO 26262:2018, Section. 11.4.5.4

Aspect	Picture	Description
Target	***Assignment of the appropriate Tool Confidence Level (TCL)*** *based on Tool Impact (TI) and Tool Error Detection (TD)*	*Determine* ***how high the level of trust*** *in the tool must be (or whether tool qualification is necessary)*
TCL1 (low)	▪ Tool has no safety-related influence (TI1) or ▪ Errors are reliably detected (TI2 + TD1) → No tool qualification necessary	▪ **Low significance** for product quality ▪ No tool qualification necessary → Confidence in tool behaviour from an ISO 26262 perspective **not critical**
TCL2 (medium)	▪ Tool has safety-related influence (TI2) ▪ Error **not reliably detectable (TD2)**	▪ Tool **important for product quality** ▪ Tool qualification required ▪ Requirements depend on ASIL level
TCL3 (high)	▪ Requirements depend on ASIL level (TI2) ▪ Errors **remain undetected (TD3)**	▪ **High importance** for product quality ▪ **Tool qualification absolutely necessary**, high requirements ▪ Differences between TCL2 and TCL3 are methodologically relevant but not dramatic (depending on ASIL)

Background: calculation of TCL

AQI Automotive Quality Institute

Need for trust in tool usage

Starting point: How high must the level of trust in the tool be (or is tool qualification necessary)?

Table 3 — Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

Source: ISO 26262:2018, Section. 11.4.5.4

Aspect	Picture	Description
Target	***Assignment of the appropriate Tool Confidence Level (TCL)*** *based on Tool Impact (TI) and Tool Error Detection (TD)*	*Determine* ***how high the level of trust*** *in the tool must be (or whether tool qualification is necessary)*
TCL1 (low)	▪ Tool has no safety-related influence (TI1) or ▪ Errors are reliably detected (TI2 + TD1) → No tool qualification necessary	▪ Low significance for product quality ▪ No tool qualification necessary → Confidence in tool behaviour from an ISO 26262 perspective not critical
TCL2 (medium)	▪ Tool has safety-related influence (TI2) ▪ Error **not reliably detectable (TD2)**	▪ Tool **important for product quality** ▪ Tool qualification required ▪ Requirements depend on ASIL level
TCL3 (high)	▪ Requirements depend on ASIL level (TI2) ▪ Errors **remain undetected (TD3)**	▪ High importance for product quality ▪ **Tool qualification absolutely necessary**, high requirements ▪ Differences between TCL2 and TCL3 are methodologically relevant but not dramatic (depending on ASIL)

Background: calculation of TCL

AQI Automotive Quality Institute

Need for trust in tool usage

Starting point: How high must the level of trust in the tool be (or is tool qualification necessary)?

Table 3 — Determination of the Tool Confidence Level (TCL)

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

Source: ISO 26262:2018, Section. 11.4.5.4

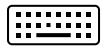
Aspect	Picture	Description
Target	***Assignment of the appropriate Tool Confidence Level (TCL)*** *based on Tool Impact (TI) and Tool Error Detection (TD)*	*Determine* ***how high the level of trust*** *in the tool must be (or whether tool qualification is necessary)*
TCL1 (low)	▪ Tool has no safety-related influence (TI1) or ▪ Errors are reliably detected (TI2 + TD1) → No tool qualification necessary	▪ Low significance for product quality ▪ No tool qualification necessary → Confidence in tool behaviour from an ISO 26262 perspective not critical
TCL2 (medium)	▪ Tool has safety-related influence (TI2) ▪ Error not reliably detectable (TD2)	▪ Tool important for product quality ▪ Tool qualification required ▪ Requirements depend on ASIL level
TCL3 (high)	▪ Requirements depend on ASIL level (TI2) ▪ Errors **remain undetected (TD3)**	▪ **High importance** for product quality ▪ **Tool qualification absolutely necessary**, high requirements ▪ Differences between TCL2 and TCL3 are methodologically relevant but not dramatic (depending on ASIL)

FMEA Excel: calculation of TCL

AQI Automotive Quality Institute

Need for trust in tool usage

Starting point: How high must the level of trust in the tool be (or is tool qualification necessary)?

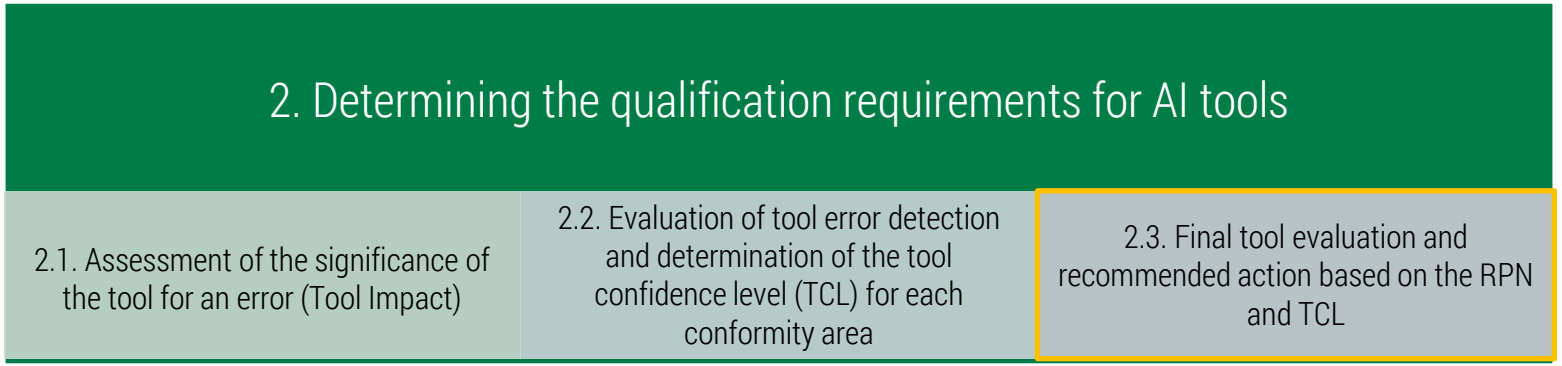
	**Input**	Tool Confidence Level - Assessment of the existence of trust
	**Scale**	Trust level (1 = low, 2 = medium, 3 = high)

Potential Tool Error	Testable Tool Capabilities	Tool Impact Value	Tool Error Detection	Tool Confidence Value
May fail to guarantee deterministic training execution and runtime validation, resulting in trained models that do not meet real-time safety-critical performance requirements.	Framework support for model optimization (e.g., TensorRT, ONNX Runtime optimizations). Support for hardware-specific runtime optimizations and deployment targeting (e.g., EdgeTPU, CUDA kernels). Framework support for pipeline fusion, minimal preprocessing APIs, and asynchronous inference.	1	1	TCL1
May fail to provide comprehensive error logging and runtime monitoring, resulting in trained models with limited traceability for failure analysis in safety-critical systems.	Framework support for structured error reporting, custom callbacks, and logging integration. Built-in hooks for anomaly detection during training and validation (gradient checkers, divergence detectors). ML lifecycle management tools (e.g., MLflow, Weights & Biases) integration for metadata and error tracking.	2	2	TCL2
May fail to enforce deterministic training procedures and runtime validation, resulting in trained models with unpredictable behavior and poor generalization in safety-critical applications.	Explicit deterministic training support with reproducibility settings and controlled parallelism. Built-in validation dashboards and metric tracking APIs. Numerical stability modules with configurable regularization options.	2	3	TCL3

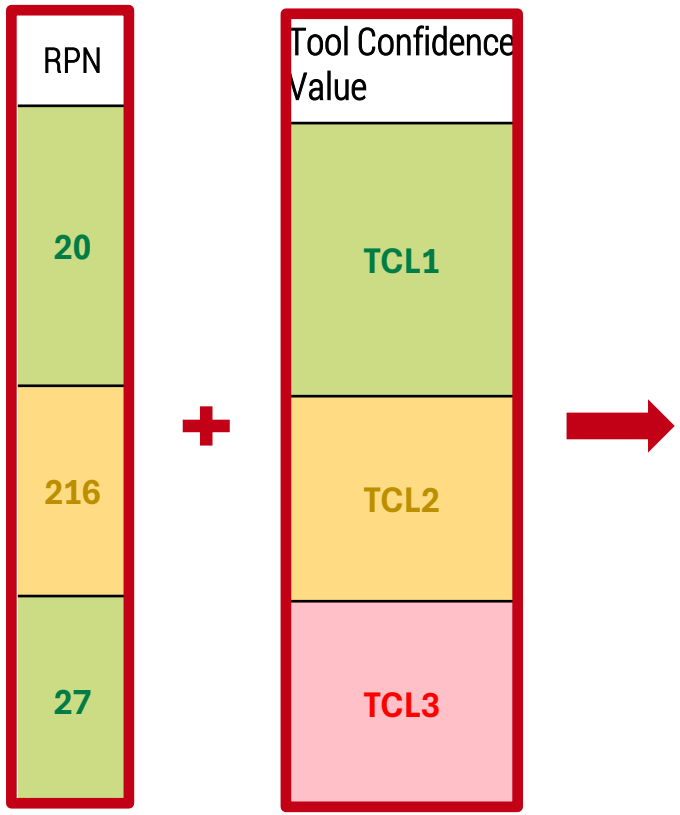
AQI Automotive Quality Institute

2. Determining the qualification requirements for AI tools

2.3. Tool assessment and recommended actions based on RPN and TCL



FEMA Excel: Summary and action



Qualification recommendation in accordance with ISO 26262:2018, Section 11.4.6

1. Step: Assessment of the trust requirement (TCL)

TCL = 1

No tool qualification necessary, as sufficient measures are already in place to detect errors!

TCL = 2 oder 3

Tool qualification necessary, as these have a **greater impact** on **functional safety**!

Continue to next slide

FEMA Excel: Summary and action

Use „Mitigations“ and „Testable Tool Capabilities“ for further mitigation activities and documentation.

Mitigations	Testable Tool Capabilities	Tool Confidence Value
✓ Integrate structured exception handling and mandatory logging for all critical training and inference operations. ✓ Implement runtime monitors for numerical instabilities (e.g., NaNs, divergence) and alert on thresholds. – Embed pipeline-wide metadata capture and systematic experiment logging into the training framework.	✓ Framework support for structured error reporting, custom callbacks, and logging integration. – Built-in hooks for anomaly detection during training and validation (gradient checkers, divergence detectors). ✓ ML lifecycle management tools (e.g., MLflow, Weights & Biases) integration for metadata and error tracking.	TCL2
✓ Implement deterministic training pipelines with seed control and precision configuration to ensure repeatability. ✓ Integrate continuous performance monitoring during training with automatic validation checkpoints. – Use frameworks that incorporate gradient clipping, normalization, and regularization for improved numerical robustness.	✓ Explicit deterministic training support with reproducibility settings and controlled parallelism. ✓ Built-in validation dashboards and metric tracking APIs. – Numerical stability modules with configurable regularization options.	TCL3

Please note: The Excel application is currently not supporting direct support for check marking and coloring the mitigations and testable tool capabilities as depicted above.