AQI Automotive Quality Institute

Projekt „Nachweisführung für sicherheitskritische KI/ML-Komponenten“

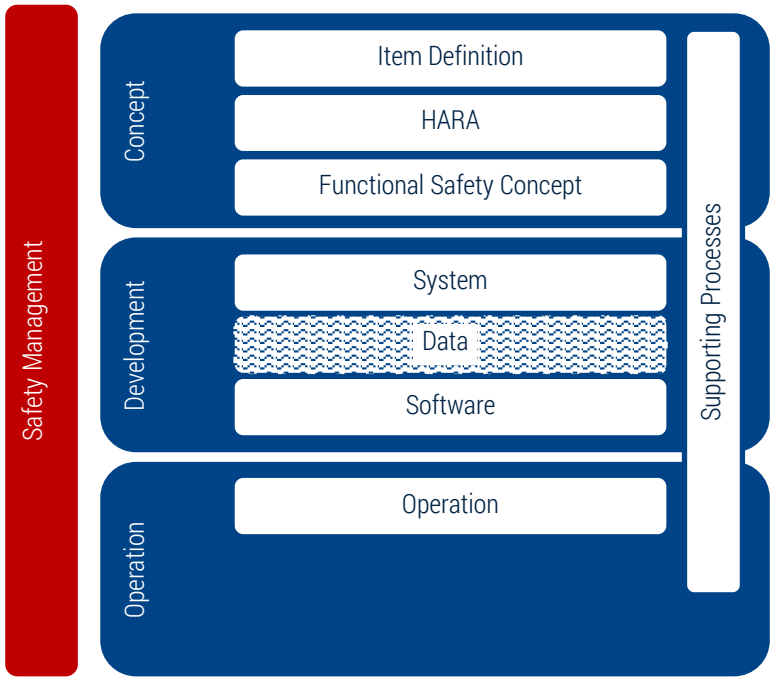
Leitfaden als Ergänzung zum Abschlussbericht

Safety Lifecycle als Anker für Analyse/Erweiterung

Analyse entlang der Phasen des Lebenszyklus und der Handlungsfelder

Automotive Quality Institute

Lebenszyklus-Phase

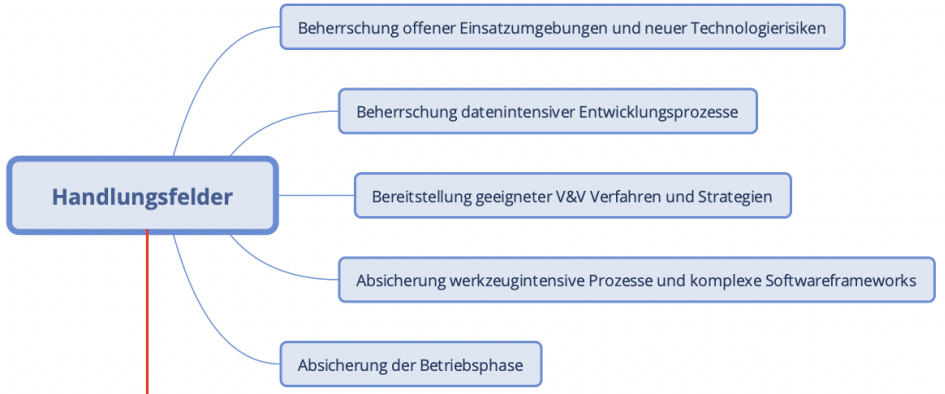


Herausforderungen mit Beispiel

Risiken

Maßnahmen / Handlungsempfehlungen

Erweiterte Dokumentations- und Nachweispflicht



Handlungsfeld: Beherrschung offener Einsatzumgebungen und neuer Technologierisiken

Herausforderung: Systeme oder Komponenten mit ML-Anteilen weisen **ML-spezifische Fehlermodi bzw. Risikofaktoren** auf (z.B. Verwundbarkeit durch Adversarial Examples), die ggfs. zu Systemausfällen, Systemfehlern oder anderen Unzulänglichkeiten mit sicherheitskritischen Implikationen führen können.

Beispiel: *Objekte lassen sich künstlich so manipulieren, dass sie von einem neuronalen Netzt nicht mehr als das, was sie sind, erkannt werden. Entsprechende Versuche mit einem Stoppschild haben gezeigt, dass die Möglichkeit besteht, durch* ***geschickt platzierte Aufkleber das Stoppschild beispielsweise als Schild zur Geschwindigkeitsbegrenzung*** *erscheinen zu lassen.*

Risiko: Werden ML-spezifische Fehlermodi bzw. Risikofaktoren nicht explizit im Sicherheitskonzept berücksichtigt, können keine geeigneten Gegenmaßnahmen entwickelt werden und die Systeme werden potenziell unsicher. **Existierende Kataloge, Best Practices und Empfehlungen** zur Durchführung von HARA decken **ML-spezifische Fehlermodi** und die damit verbundenen Risiken **nicht** ab.

Maßnahmen: Berücksichtigung **ML-spezifischer Fehlermodi** in der HARA. Ausgehend von den für die jeweils **verwendete Technologie** zu erwartenden ML-spezifischen Fehlermodi ist zu ermitteln, welche **Risiken für das Gesamtsystem** entstehen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass in der HARA auch die ML-spezifischen Fehlermodi berücksichtigt und in ihrer Bedeutung für die Systemrisiken erfasst wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

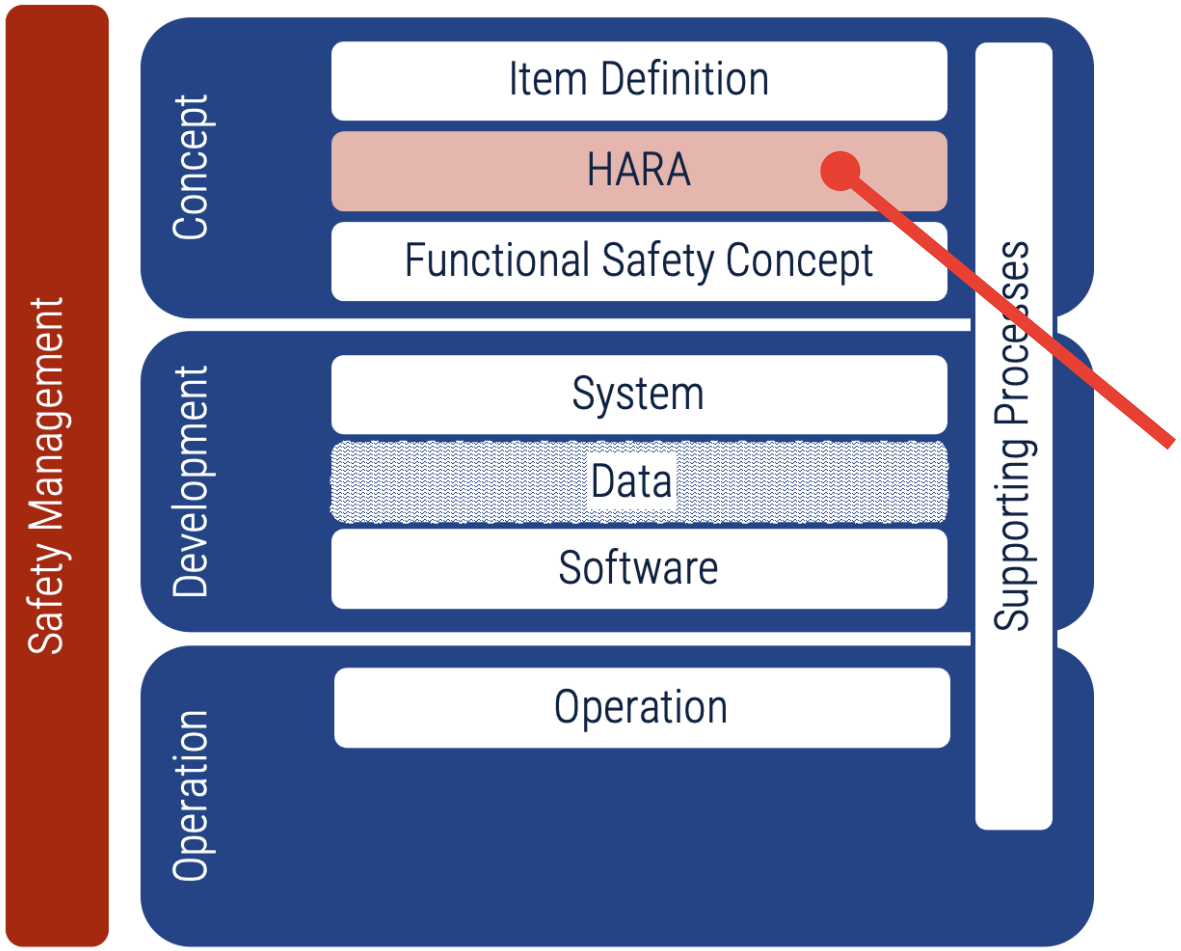
- 3.7.5.1 Hazard Analysis and risk assessment (ISO 26262)
- 6.6.1 Hazards at the vehicle level (SOTIF)
- 6.6.2 Risk evaluation of hazardous behaviors (SOTIF)
- 6.6.3 Acceptance criteria (SOTIF)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus

- *15-ML08 Risikoreport ML*

Safety Lifecycle als Anker für Analyse/Erweiterung

Herausforderungen/Konsequenz/Maßnahmen



Handlungsfeld: Beherrschung offener Einsatzumgebungen und neuer Technologierisiken

Herausforderung: Systeme oder Komponenten mit ML-Anteilen weisen **ML-spezifische Fehlermodi bzw. Risikofaktoren** auf (z.B. Verwundbarkeit durch Adversarial Examples), die ggfs. zu Systemausfällen, Systemfehlern oder anderen Unzulänglichkeiten mit sicherheitskritischen Implikationen führen können.

Beispiel: *Objekte lassen sich künstlich so manipulieren, dass sie von einem neuronalen Netzt nicht mehr als das, was sie sind, erkannt werden. Entsprechende Versuche mit einem Stoppschild haben gezeigt, dass die Möglichkeit besteht, durch* ***geschickt platzierte Aufkleber das Stoppschild beispielsweise als Schild zur Geschwindigkeitsbegrenzung*** *erscheinen zu lassen.*

Risiko: Werden ML-spezifische Fehlermodi bzw. Risikofaktoren nicht explizit im Sicherheitskonzept berücksichtigt, können keine geeigneten Gegenmaßnahmen entwickelt werden und die Systeme werden potenziell unsicher. **Existierende Kataloge, Best Practices und Empfehlungen** zur Durchführung von HARA decken **ML-spezifische Fehlermodi** und die damit verbundenen Risiken **nicht** ab.

Maßnahmen: Berücksichtigung **ML-spezifischer Fehlermodi** in der HARA. Ausgehend von den für die jeweils **verwendete Technologie** zu erwartenden ML-spezifischen Fehlermodi ist zu ermitteln, welche **Risiken für das Gesamtsystem** entstehen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass in der HARA auch die ML-spezifischen Fehlermodi berücksichtigt und in ihrer Bedeutung für die Systemrisiken erfasst wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 3.7.5.1 Hazard Analysis and risk assessment (ISO 26262)
- 6.6.1 Hazards at the vehicle level (SOTIF)
- 6.6.2 Risk evaluation of hazardous behaviors (SOTIF)
- 6.6.3 Acceptance criteria (SOTIF)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus

- *15-ML08 Risikoreport ML*

Dokumentations- und Nachweispflicht

Arbeitsprodukte, in denen die zusätzlichen Maßnahmen dokumentiert werden müssen

- Inhaltliche Definition der Dokumentations- und Nachweispflicht
- Verweis auf Arbeitsprodukte (work products) als potentielle Dokumentation des Nachweises
  - ISO 26262
  - ISO/DIS 21448:2021
- ML spezifische Arbeitsprodukte (work products), die im Vorgängerprojekt als Ergebnis des *Referenzprozess ML definiert wurden.*

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass in der HARA auch die ML-spezifischen Fehlermodi berücksichtigt und in ihrer Bedeutung für die Systemrisiken erfasst wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

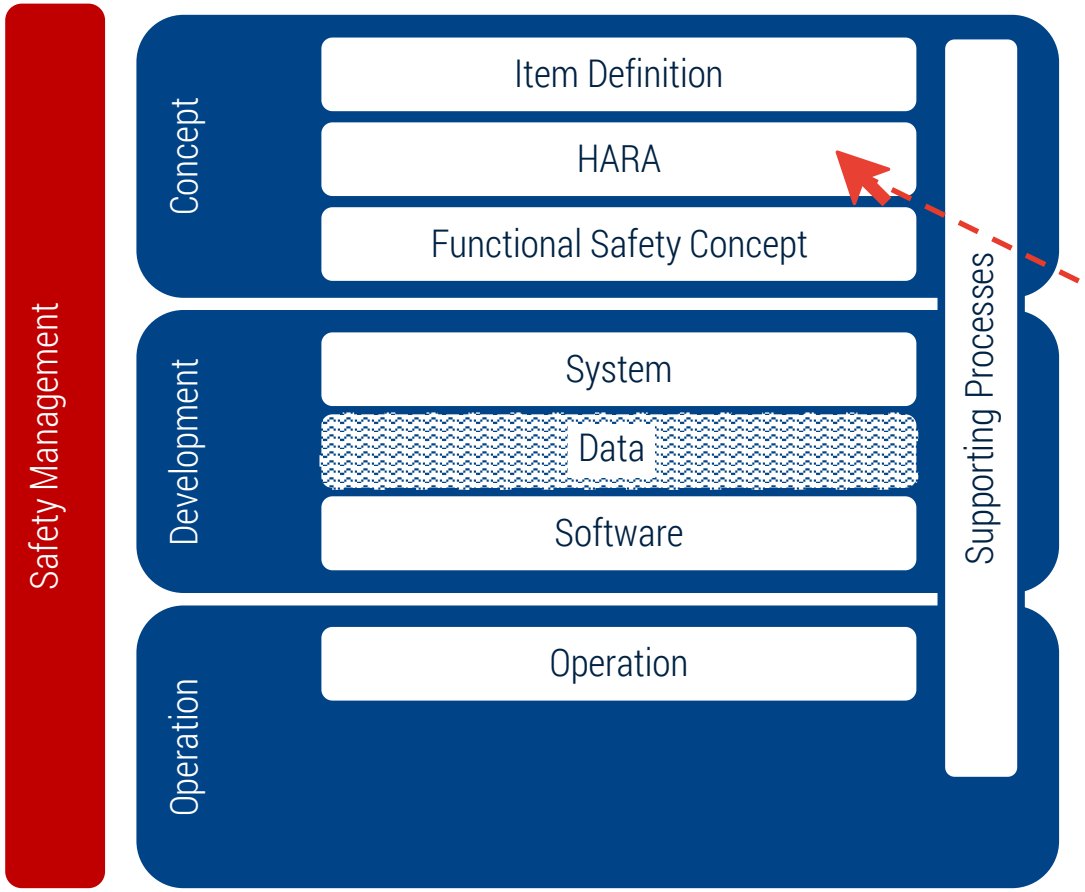
- 3.7.5.1 Hazard Analysis and risk assessment (ISO 26262)
- 6.6.1 Hazards at the vehicle level (SOTIF)
- 6.6.2 Risk evaluation of hazardous behaviors (SOTIF)
- 6.6.3 Acceptance criteria (SOTIF)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus

- *15-ML08 Risikoreport ML*

Navigation

Analyse entlang der Phasen des Lebenszyklus und der Handlungsfelder

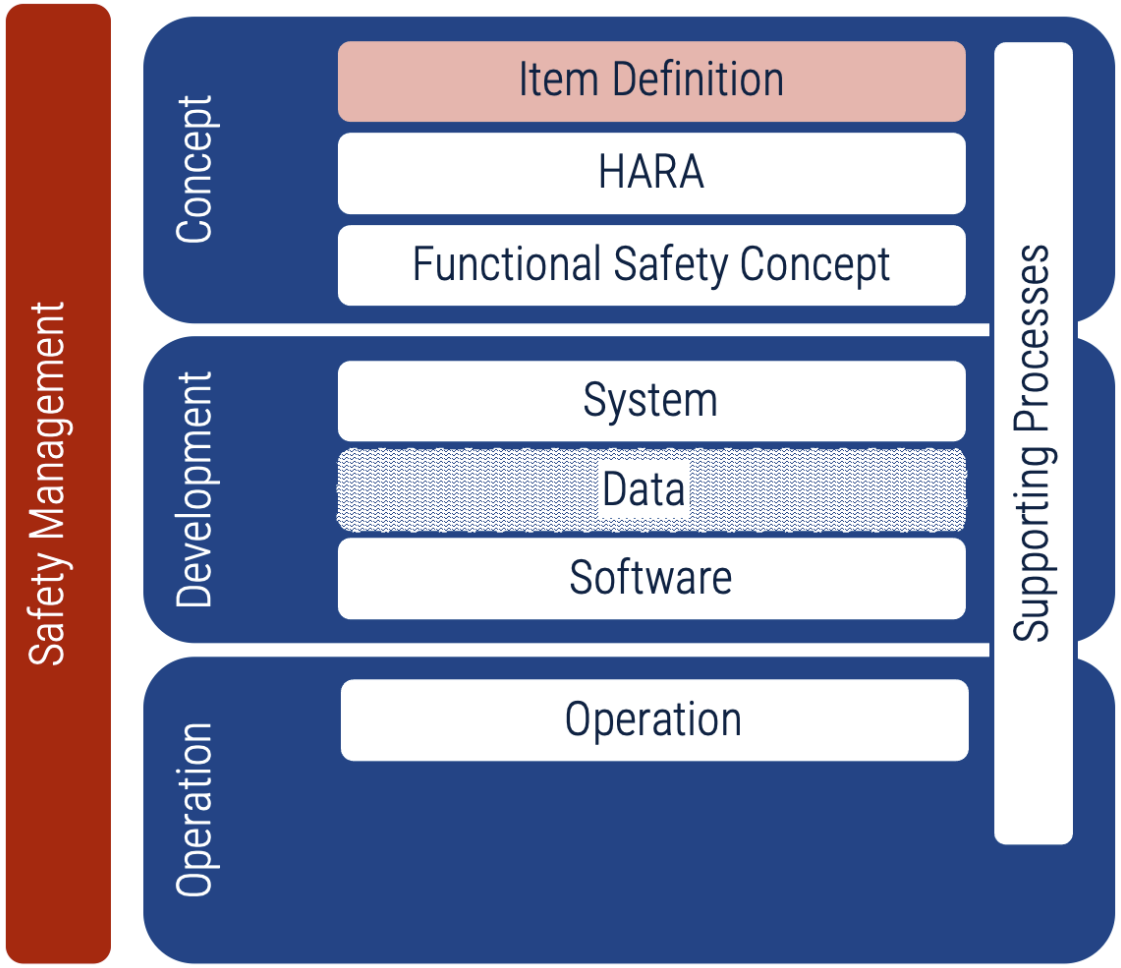


Durch Klicken auf die einzelnen Phasen des Lebenszyklus gelangen Sie auf den entsprechenden Abschnitt in den Folien.

Auf den Folien der Lebenszyklusphasen befindet sich ein Button, *zurück zur Navigation* über den Sie zurück zur Übersicht gelangen.

Item Definition

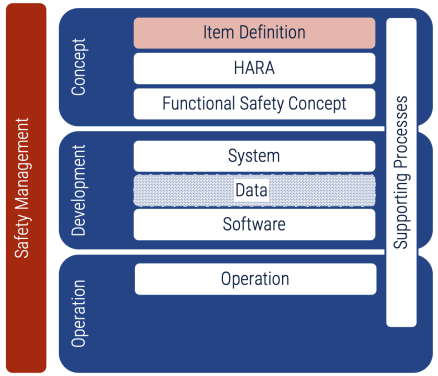
Aufgabe dieser Phase ist es, das System, seine Funktionalität, seine Abhängigkeiten und Wechselwirkung mit dem Fahrer, der Umgebung und anderen Systemen auf Fahrzeugebene zu definieren und zu beschreiben und ein angemessenes Verständnis des Systems zu schaffen, damit die Aktivitäten in den nachfolgenden Phasen durchgeführt werden können



zurück zur Navigation

Item Definition

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/2)



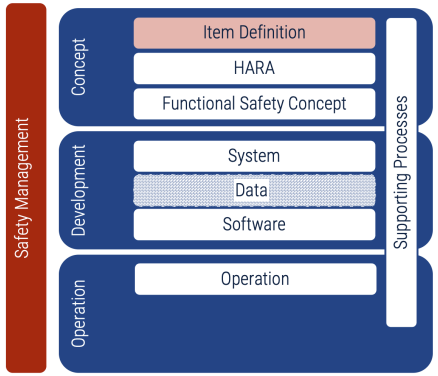
Herausforderung: ML wird insbesondere für die Realisierung von Anwendungen in komplexen Umgebungen eingesetzt, die sich mit den Methoden der klassischen Softwareentwicklung i.d.R. nicht mehr umsetzen lassen. Solche sog. offenen Umgebungen zeichnen sich dadurch aus, dass zum Zeitpunkt der Systemspezifikation nicht alle Konfigurationen bzw. Interaktionen in der Umgebung bekannt sind.

***Beispiel**: Bei der Realisierung einer KI/ML-basierten Software, die Objekte entlang potenzieller Trajektorien eines Fahrzeuges erkennen und klassifizieren können soll, ist es nicht möglich, alle Objektklassen und Szenarien umfassend zu definieren, die für das Fahrzeug und die Fahrzeugführung relevante Gefahren darstellen. Es muss grundsätzlich davon ausgegangen werden, dass Objekte relevant werden, die zum Spezifikationszeitpunkt nicht berücksichtigt wurden (z.B. ein als Litfaßsäule verkleideter Mensch am Straßenrand). Zusätzlich muss davon ausgegangen werden, dass die Software auch bekannte Objekte nicht immer richtig identifizieren und klassifizieren kann (z.B. ist für eine auch einfachen Kamerabildern basierende Objekterkennung ein auf einem LKW Container aufgedrucktes Haus evtl. eher ein Haus als ein LKW mit Container). Im Rahmen der Spezifikation müssen Kennzahlen definiert werden, mit der sich auch eine fehlerhafte Erkennungsleistung in Bezug auf die Sicherheit des Systems quantifizieren lässt.*

Risiko: Da Umgebungen und Interaktionen weder vollständig bekannt noch vollständig spezifizierbar sind, ist es i.d.R. nicht möglich eine vollständige, formale Spezifikation zu erstellen, mit der sich die Systemfunktionalität in allen Details umfassend beschreiben lässt.

Item Definition

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/2)



Maßnahmen:

- Einführung von Spezifikationsmethoden, die in der Lage sind, Unsicherheiten über die Systemumgebung sowie die Interaktion des Systems mit dieser Umgebung auszudrücken.
- Entwicklung des Systemverständnisses auf Basis des Wissens von Domänenexperten, die in den nachfolgenden Prozessschritten mit ihren Erfahrungen die Identifikation von Risiken und das Ableiten von Sicherheitsfunktionen unterstützen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Es ist nachzuweisen, dass das System in seiner Interaktion mit der Umgebung ausreichend verstanden und dokumentiert ist, um mit dem System zusammenhängende Gefährdungen identifizieren und reduzieren zu können. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

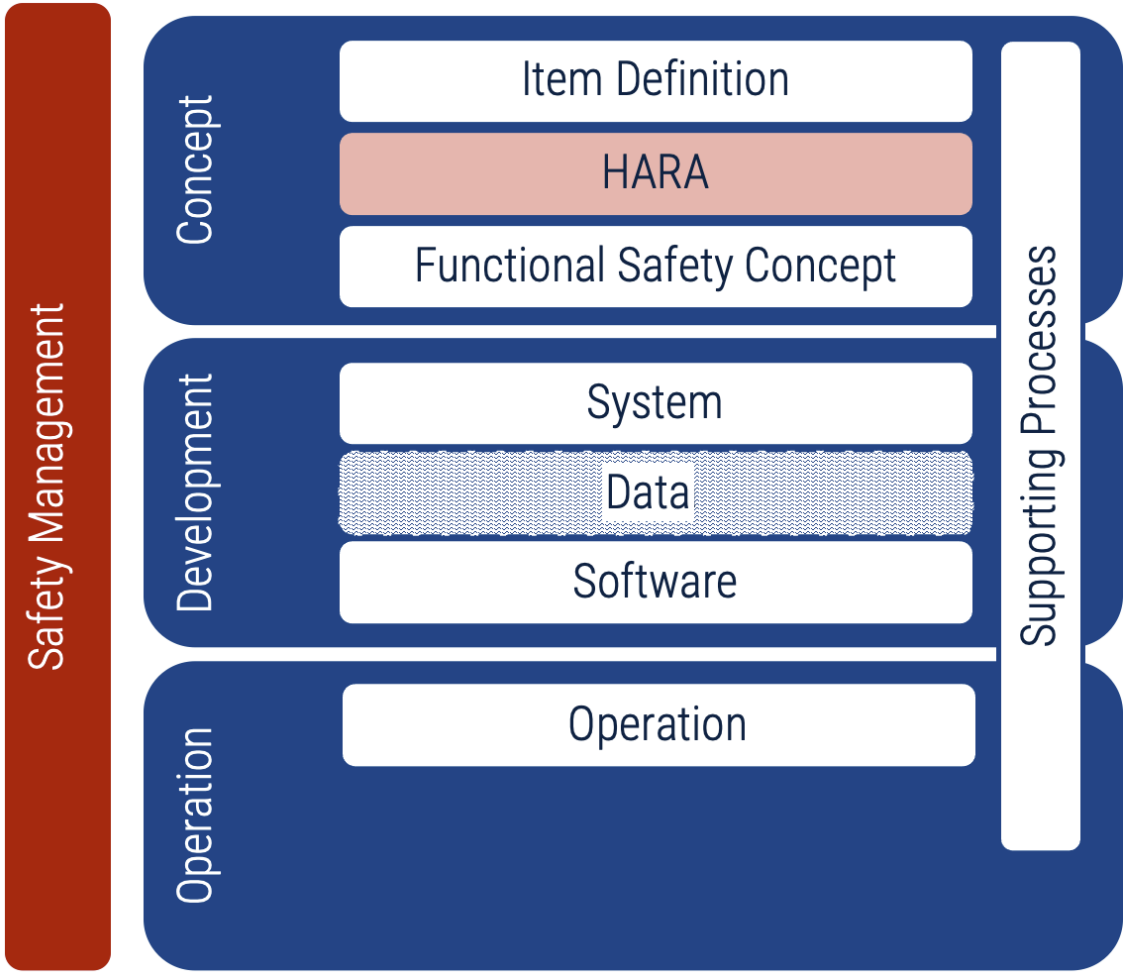
- 3.6.5.5 Item Definition (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus

- *04-ML01: Domain Model*
- *17-ML11: Anforderungsspezifikation ML-Modell und Daten*

Hazard and Risk Analysis (HARA)

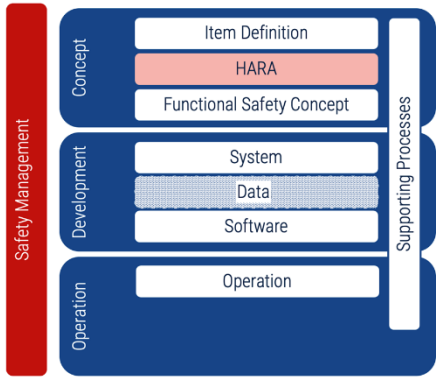
Aufgabe dieser Phase ist es, gefährliche Ereignisse, die durch Fehlfunktionen des Systems verursacht werden, zu identifizieren und zu klassifizieren und Sicherheitsziele mit den entsprechenden ASILs zu formulieren, die sich auf die Vermeidung oder Minderung der Gefährdungen beziehen, um ein unverhältnismäßiges Risiko zu vermeiden.



zurück zur Navigation

Hazard and Risk Analysis (HARA)

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/2)



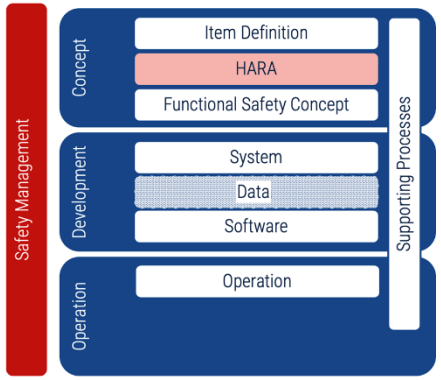
Herausforderung: In offenen Umgebungen sind i.d.R. nicht alle potenziell möglichen Konfigurationen und Ereignisse bekannt. Vor diesem Hintergrund ist davon auszugehen, dass es eine relevante Anzahl potenzieller gefährlicher Ereignisse gibt, die aufgrund der Komplexität der Aufgabe, der Umgebung und der vielfältigen Wechselwirkungen zwischen System und Umgebung schwer zu ermitteln bzw. vollständig unbekannt sind.

***Beispiel**: Ein autonomes Fahrzeug muss auch unter extremen Bedingungen ein sinnvolles Verhalten zeigen. Ist es beispielsweise die KI/ML-basierte Software eines solchen Fahrzeugs mit einem verunglückten Gefahrenguttransporter konfrontiert, aus dem Gefahrgut austritt, muss eine solche Situation in ihrer Bedeutung erkannt und anschließend sinnvolles Handeln abgeleitet werden. Sinnvoll Handeln könnte in diesem Fall sein, das Fahrzeug mit den Insassen schnellstmöglich aus der Gefahrenzone zu bringen, ohne erst am Gefahrguttransporter vorbeizufahren. Voraussetzung dafür ist, dass das Fahrzeug die Situation in ihrer gesamten Bedeutung (ein verunglückten Gefahrenguttransporter ist kein einfaches Hindernis) erkennt und in der Lage ist in einer Ausnahmesituation aus der Gefahrenzone zu navigieren.*

Risiko: Selten auftretende, aber potenziell gefährliche Ereignisse werden in der HARA ungenügend berücksichtigt und können entsprechend nicht im Sicherheitskonzept systematisch adressiert werden.

Hazard and Risk Analysis (HARA)

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/2)



Maßnahmen:

- Einführungen von HARA Methoden, die in der Lage sind, selten auftretende aber potenziell gefährliche Ereignisse und die damit verbundenen Risiken zu identifizieren und zu analysieren.
- Entwicklung eines Risikoverständnisses auf Basis des Wissens von Domänenexperten.

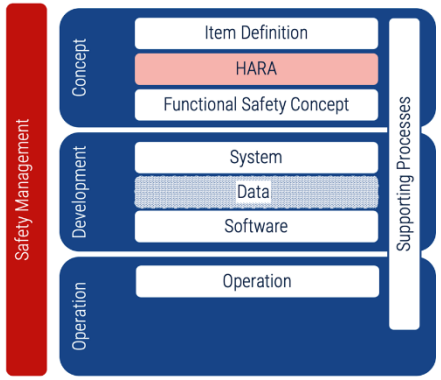
Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Es ist nachzuweisen, dass alle potenziell gefährlichen Ereignisse in der HARA berücksichtigt werden und dass entsprechendes Domänenwissen bei der Identifikation der Risiken berücksichtigt wurde. Dies kann u.a. durch Rückgriff auf offizielle Fehler und Unfalldatenbanken geschehen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 3.7.5.1 Hazard Analysis and risk assessment (ISO 26262)
- 6.6.1 Hazards at the vehicle level (SOTIF)
- 6.6.2 Risk evaluation of hazardous behaviors (SOTIF)
- 6.6.3 Acceptance criteria (SOTIF)

Hazard and Risk Analysis (HARA)

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/2)



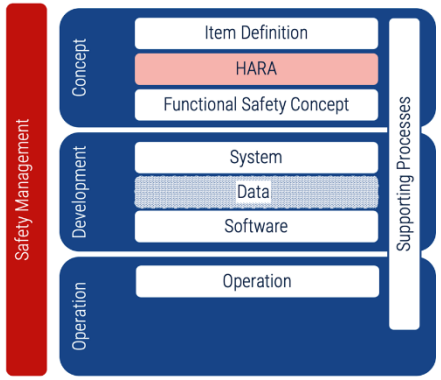
Herausforderung: Systeme oder Komponenten mit ML-Anteilen weisen ML-spezifische Fehlermodi bzw. Risikofaktoren auf (z.B. BIAS, Empfindlichkeit gegen Verteilungsverschiebungen in der Umgebung, Verwundbarkeit durch Adversarial Examples), die ggfs. zu Systemausfällen, Systemfehlern oder anderen Unzulänglichkeiten mit sicherheitskritischen Implikationen führen können.

Beispiel:: *Durch gezielte Manipulation an den Eingangsdaten eines neuronalen Netzes zur Klassifikation von Objekten kann erwirkt werden, dass eine ursprünglich korrekt klassifizierte Bild vom gleichen Netzwerk als etwas völlig anderes klassifiziert wird. Die Fehlklassifikation kann dabei vom Angreifer beliebig bestimmt werden. Untersuchungen haben gezeigt, dass dies nicht nur für die digitalen Daten gilt. Beliebige Objekte, d.h. auch Schilder und andere für die Verkehrssteuerung wichtige Objekte lassen, sich künstlich in der physikalischen Welt so manipulieren, dass sie von einem neuronalen Netzt nicht mehr als das, was sie sind, erkannt werden. Entsprechende Versuche mit einem Stoppschild haben gezeigt, dass die Möglichkeit besteht, durch geschickt platzierte Aufkleber das Stoppschild beispielsweise als Schild zur Geschwindigkeitsbegrenzung erscheinen zu lassen..*

Risiko: Werden ML-spezifische Fehlermodi bzw. Risikofaktoren nicht explizit im Sicherheitskonzept berücksichtigt, können keine geeigneten Gegenmaßnahmen entwickelt werden und die Systeme werden potenziell unsicher. Existierende Kataloge, Best Practices und Empfehlungen zur Durchführung von HARA decken ML-spezifische Fehlermodi und die damit verbundenen Risiken nicht ab.

Hazard and Risk Analysis (HARA)

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/2)



Maßnahmen: Berücksichtigung ML-spezifischer Fehlermodi in der HARA. Ausgehend von den für die jeweils verwendete Technologie zu erwartenden ML-spezifischen Fehlermodi ist zu ermitteln, welche Risiken für das Gesamtsystem entstehen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Es ist nachzuweisen, dass in der HARA auch die ML-spezifischen Fehlermodi (z.B. BIAS, Empfindlichkeit gegen Verteilungsverschiebungen in der Umgebung, Verwundbarkeit durch Adversarial Examples) berücksichtigt und in ihrer Bedeutung für die Systemrisiken erfasst wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 3.7.5.1 Hazard Analysis and risk assessment (ISO 26262)
- 6.6.1 Hazards at the vehicle level (SOTIF)
- 6.6.2 Risk evaluation of hazardous behaviors (SOTIF)
- 6.6.3 Acceptance criteria (SOTIF)

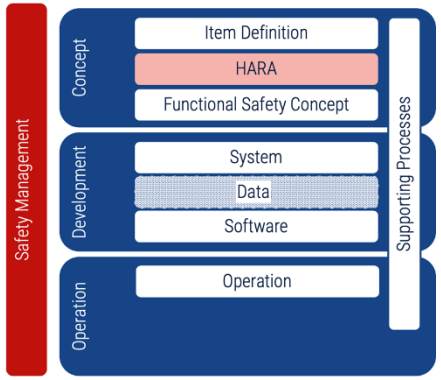
Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 15-ML08 Risikoreport ML

Hazard and Risk Analysis (HARA)

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/2)

AQI Automotive Quality Institute



Herausforderung: Werden Systeme oder Komponenten mit ML-Anteilen in Umgebungen mit anderen Systemen und Akteuren verwendet, resultieren daraus neue Risiken, die in den komplexen Interaktionen zwischen mehreren semi-intelligenten und intelligenten Agenten (autonomen Fahrzeugen, menschlichen Fahrer/innen und weiteren Verkehrsteilnehmer/innen) begründet sind.

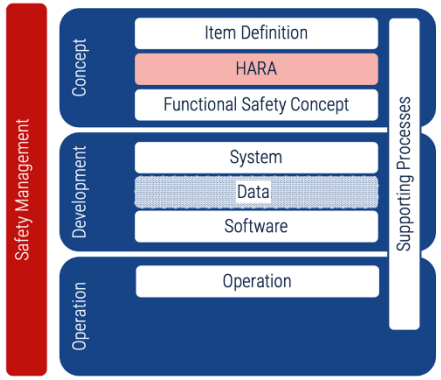
***Beispiel**: Interaktion zwischen autonomen Fahrzeugen und Fußgänger/innen können sich über einen längeren Betriebszeitraum dahingehend verändern, dass Fußgänger/innen unvorsichtiger agieren, da sie gelernt haben, dass automatisierte Fahrzeuge deutlich eher und effektiver Bremsen als menschliche Fahrer/innen.*

Risiko: Werden ML-spezifische Risiken, die in den komplexen Interaktionen zwischen mehreren semi-intelligenten und intelligenten Agenten begründet sind, nicht berücksichtigt, können keine geeigneten Gegenmaßnahmen entwickelt werden und die Systeme werden potenziell unsicher. Existierende Kataloge, Best Practices und Empfehlungen zur Durchführung von HARA decken (Multi)Agenten-spezifische Fehlermodi und die damit verbundenen Risiken nicht ab.

Hazard and Risk Analysis (HARA)

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/2)

AQI Automotive Quality Institute



Maßnahmen:

Einführungen von HARA Methoden, die in der Lage sind, (Multi)Agenten-spezifische Fehlermodi und die damit verbundenen Risiken zu adressieren.

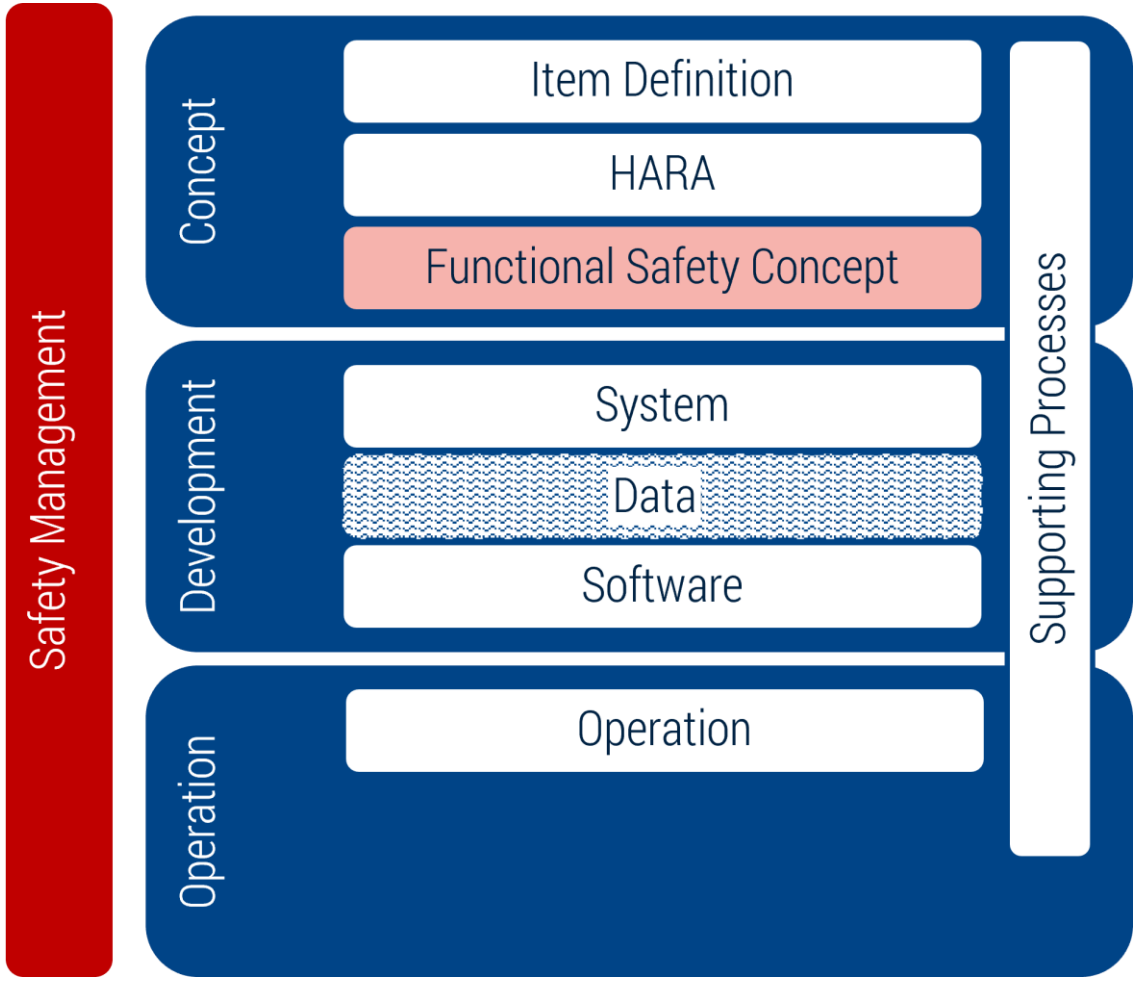
Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Bei Systemen, die in Umgebungen mit mehreren semi-intelligenten und intelligenten Agenten interagieren (z.B. autonome Fahrzeuge) ist nachzuweisen, dass in der HARA Risiken berücksichtigt werden, die durch die Interaktion mehrerer solcher Agenten begründet sind. Hierzu zählen beispielsweise Risiken, die dadurch entstehen, dass andere Verkehrsteilnehmer neue Fähigkeiten autonomer Fahrzeuge (schnelle Bremsmanöver) antizipieren und ihr Verhalten in der Interaktion mit autonomen Fahrzeugen anpassen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 3.7.5.1 Hazard Analysis and risk assessment (ISO 26262)
- 6.6.1 Hazards at the vehicle level (SOTIF)
- 6.6.2 Risk evaluation of hazardous behaviors (SOTIF)
- 6.6.3 Acceptance criteria (SOTIF)

Funktionales Sicherheitskonzept

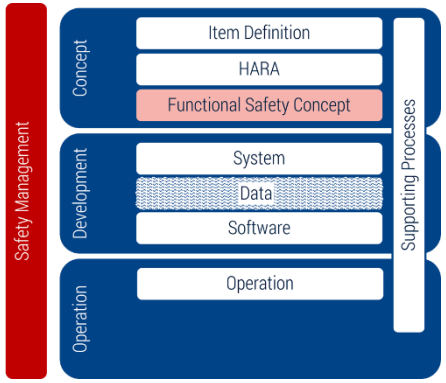
Ziel dieser Phase ist es, das gewünschte und das fehlerhafte Funktionsverhalten des Systems in Übereinstimmung mit den zuvor festgelegten Sicherheitszielen zu spezifizieren. Dabei werden die Einschränkungen hinsichtlich einer geeigneten und frühzeitigen Erkennung und Beherrschung von relevanten Fehlern in Übereinstimmung mit den Sicherheitszielen spezifiziert, die Strategien oder Maßnahmen auf Systemebene definiert, um die geforderte Fehlertoleranz zu erreichen oder die Auswirkungen relevanter Fehler durch das System selbst, durch den Fahrer oder durch externe Faktoren angemessen zu reduzieren. Weitere Ziele der Phase sind es, die Anforderungen an die funktionale Sicherheit, dem Systemarchitekturentwurf oder externen Maßnahmen zuzuordnen, das funktionale Sicherheitskonzept zu verifizieren und die sicherheitsrelevanten Bewertungskriterien festzulegen.



zurück zur Navigation

Funktionales Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/2)



Herausforderung: Fragestellungen beim ML werden i.d.R. so modelliert, dass die Antwort eine Wahrscheinlichkeit ist. Bei einer Klassifikation beispielsweise, liefert das System für jedes zu klassifizierende Objekt und jede Klasse einen Wahrscheinlichkeitswert zurück. Zur Weiterverarbeitung müssen diese Wahrscheinlichkeitswerte sinnvoll interpretiert werden. In einer solchen Interpretation werden potenzielle sicherheitskritische Entscheidungen gefällt (ab welchen Wahrscheinlichkeitswert ist ein Objekt als Gefährdung einzustufen). Sie ist nicht Teil des Modells und muss gesondert spezifiziert und realisiert (ggfs. durch eine SW Komponente in der Nachverarbeitung) werden.

***Beispiel**: Ein System zur Objekterkennung bietet in rascher Folge neue Interpretationen eines Sachverhalts, die jeweils unterschiedliche Konsequenzen für die Aktionen eines autonomen Fahrzeugs haben. Da in der Regel keine Konfidenzinformationen für die jeweilige Interpretation vorliegen, wechselt das Fahrzeug seine Handlungsstrategie entlang der vorliegenden Interpretationen. Im Fall des Uber Unfalls wird diesbezüglich festgestellt:*

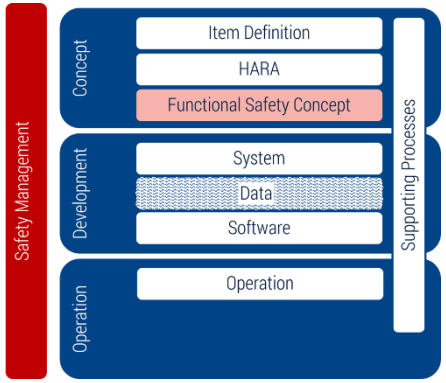
"The recorded telemetry showed the system had detected Herzberg six seconds before the crash, and classified her first as an unknown object, then as a vehicle, and finally as a bicycle, each of which had a different predicted path according to the autonomy logic." (Quelle Wikipedia)

Risiko: Es muss davon ausgegangen werden, dass ML-Komponenten nicht immer eindeutige Ergebnisse liefern. Dieser Umstand muss in der Nachverarbeitung berücksichtigt und in ein eindeutiges Entscheidungsschema überführt werden. Hierdurch entstehen vielfältige Herausforderungen, die in einem Sicherheitskonzept berücksichtigt werden müssen. Hierzu zählt:

- Identifizieren des Risikopotentials von unklaren Entscheidungssituationen und Spezifikation eines sicheren Vorgehens für solchen Situationen (Definition von Rückfallebenen),
- Einführung von Systemen, Architekturen und Komponenten, die unklare Entscheidungssituationen erkennen und behandeln können (heterogene Redundanz, Plausibilitätsprüfungen),
- Mechanismen zur Differenzierung von unklaren Entscheidungssituationen und fehlerhaften Funktionsverhalten.

Funktionales Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/2)



Maßnahmen: Definition von Maßnahmen, die geeignet sind, unklare Entscheidungssituationen zu erkennen und im Rahmen des Funktionskontexts sinnvoll aufzulösen. Hierzu zählt der Einsatz von Technologien, die eine Fehlerabschätzung möglich machen, die Definition heterogen, redundanter Systeme, die durch Vergleich die Identifikation von Entscheidungsunsicherheiten erlauben sowie die Definition geeigneter Betriebszustände, die auch in unklaren Entscheidungssituation einen sicheren Betrieb des Systems erlauben.

Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Im Rahmen des Sicherheitskonzepts ist nachzuweisen, in welcher Form ein System in der Lage ist, unklare Entscheidungssituationen zu erkennen und zu behandeln. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

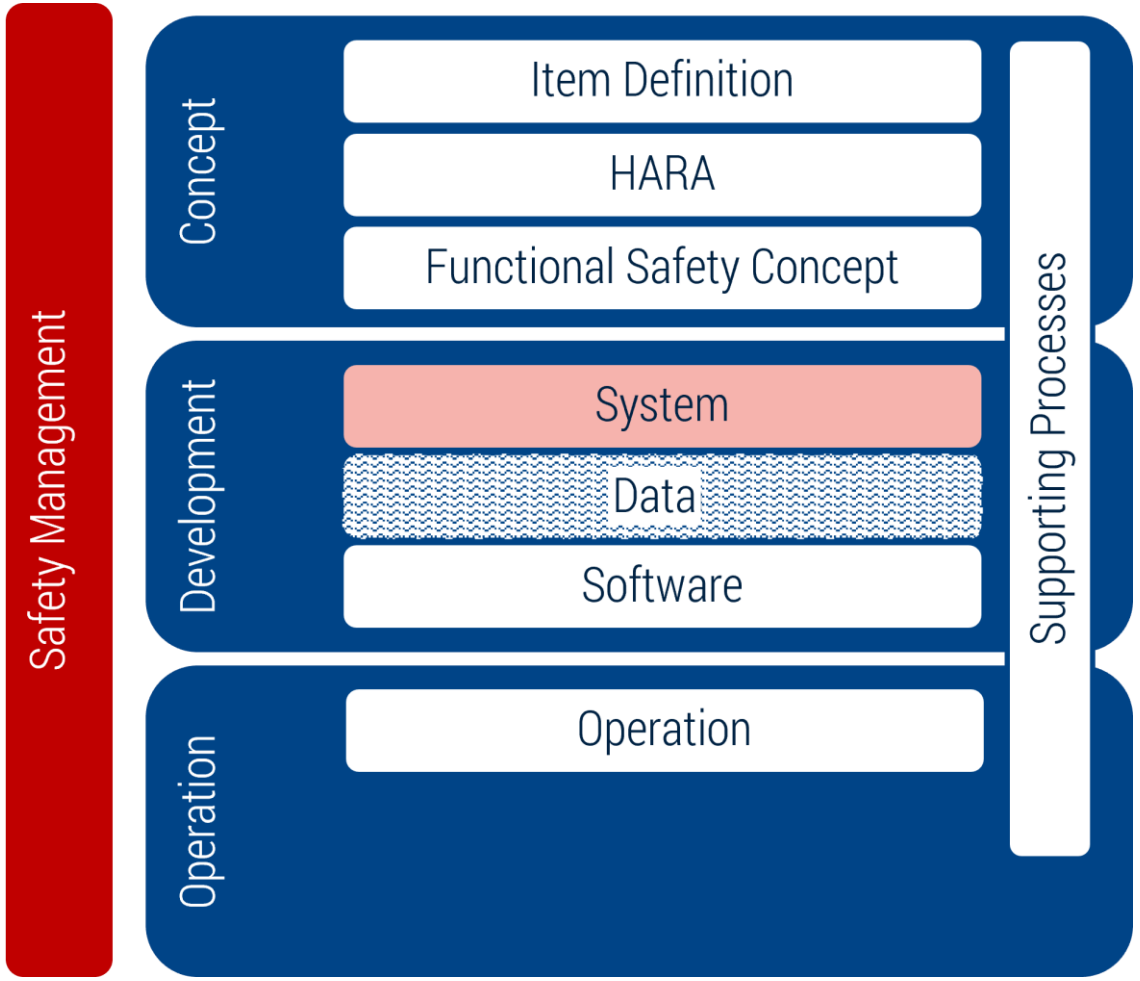
- 3.8.5.1 Functional Safety Concept (ISO 26262)
- 7.5.1 Identified potential insufficiencies of specification, performance limitations and triggering conditions (SOTIF)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus

- 17-ML08 Integrationsanforderungen ML-Modell
- 04-ML30 Initiale Lösungsarchitektur
- 15-ML08 Risikoreport ML

Technisches Sicherheitskonzept

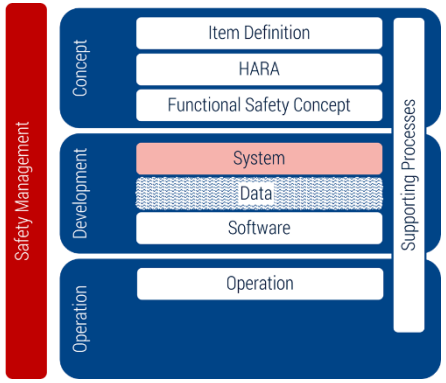
Ziel dieser Phase ist es, die sicherheitsrelevanten Anforderungen an die Funktionalität, Abhängigkeiten, Einschränkungen und Eigenschaften der Systemelemente und Schnittstellen, die für ihre Implementierung erforderlich sind, zu spezifizieren. Dabei werden sicherheitsrelevante Anforderungen hinsichtlich der in den Systemelementen und Schnittstellen zu implementierenden Sicherheitsmechanismen und die Anforderungen an die funktionale Sicherheit des Systems und seiner Komponenten während der Produktion, des Betriebs, der Wartung und der Stilllegung spezifiziert. Außerdem werden die technischen Sicherheitsanforderungen auf Eignung zum Erreichen der funktionalen Sicherheit auf Systemebene und auf Übereinstimmung mit den Anforderungen an die funktionale Sicherheit überprüft. Weitere Ziele sind die Entwicklung eines Systemarchitekturentwurfs und eines technischen Sicherheitskonzeptes, die die Sicherheitsanforderungen erfüllen und nicht im Widerspruch zu den nicht-sicherheitsbezogenen Anforderungen stehen. Weitere Aufgaben sind die Analyse des Systemarchitekturentwurfs zur Vermeidung von Fehlern, das Ableiten von notwendigen sicherheitstechnischen Besonderheiten für Produktion und Betrieb sowie die Verifizierung, dass der Systemarchitekturentwurf und das technische Sicherheitskonzept die Sicherheitsanforderungen entsprechend der jeweiligen ASIL erfüllen.



zurück zur Navigation

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/3)



Herausforderung: Eine unvollständige Spezifikation erschwert das Ableiten sowie die Verifikation von technischen Sicherheitsanforderungen.

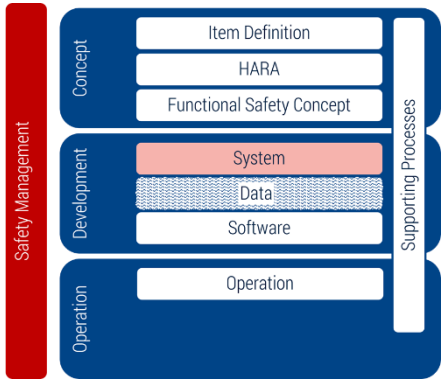
***Beispiel**: Bei der Objekterkennung eines autonomen Fahrzeugs ist zwar grundsätzlich bekannt, dass Objekte, beispielsweise bei ungünstigen Lichtverhältnissen, potenziell nicht erkannt werden. Die konkreten Situationen können aber dennoch nicht systematisch behandelt und mit technischen Gegenmaßnahmen unterlegt werden, da die Situationen im Einzelnen nicht bekannt sind.*

„Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of the impact caused the Model S to pass under the trailer, with the bottom of the trailer impacting the windshield of the Model S."

Risiko: Im Rahmen der Produktentwicklung müssen technische Sicherheitsanforderungen an das System, seine Produktion und sein Einsatz definiert werden. Das Ableiten von Sicherheitsanforderungen erfolgt in der Regel auf Basis der HARA und der funktionalen Systemspezifikation. Fehlende Vollständigkeit in der funktionalen Systemspezifikation sowie die Komplexität der Einsatzumgebung erschweren den Prozess der Ableitung und Verifikation der technischen Sicherheitsanforderungen.

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/3)

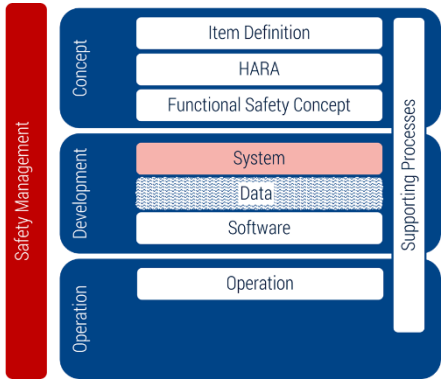


Maßnahmen: Auch wenn keine vollständige funktionale Spezifikation als Grundlage für die Definition von Sicherheitsanforderungen verfügbar ist, kann oft dennoch eine Teilspezifikation herangezogen werden. Diese abstrahiert in der Regel von Einzelfällen und trifft Allaussagen über eine potenziell unendliche Menge von Ereignissen. Eine solche Teilspezifikation lässt sich in einer formalen Sprache ausdrücken, mit der Einschränkung für alle Ein-/Ausgangspaare einer Funktion definiert werden können. Insbesondere handelt es sich um eine Art von Vorwissen, das an verschiedenen Stellen im Entwicklungs- und Modellierungsprozess verwendet werden kann, um Kriterien für die Auswahl von Daten und Modellen zu definieren, den Trainingsprozess zu steuern (Constraints als Teil der Loss Function), die Verifikation zu leiten oder im Betrieb die kontinuierliche Überwachung des Systems zu definieren (Saley et al.).

- Ableiten von Sicherheitsanforderungen an die Trainingsdaten formulieren (Qualitätskriterien wie Vollständigkeit, Relevanz, Genauigkeit etc.)
- Zusammenarbeit zwischen ML-Experten und Domänenexperten um Sicherheitsanforderungen in Bezug auf das Trainingsziel, die Art der Modellierung, die Trainingsmethode, sowie die Optimierungsziele und KPIs sinnvoll definieren zu können.

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Im Rahmen der Sicherheitsanalyse ist nachzuweisen, dass das System hinreichend genau spezifiziert ist, um sicherzustellen, dass alle identifizierten Risiken durch technische Maßnahmen ausreichend gemindert sind. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

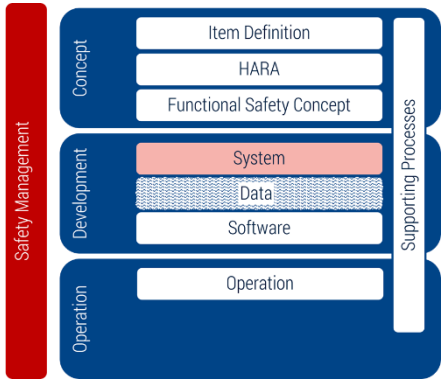
- 4.8.5.2. Safety validation report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 17-ML08 Integrationsanforderungen ML-Modell
- 17-ML02 Verifikations- und Validierungsanforderungen ML-Modell und Daten
- 04-ML30 Initiale Lösungsarchitektur

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/3)



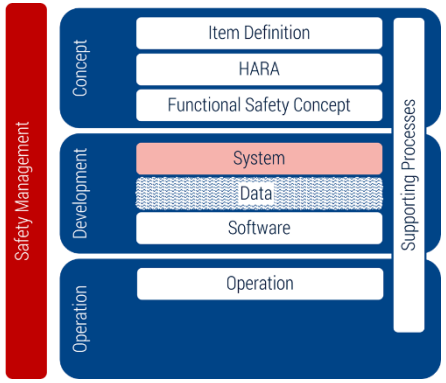
Herausforderung: Die Möglichkeit von Verteilungsverschiebungen, dem Auftauchen neuer bisher nicht entdeckter Sonderfälle führt dazu, dass sich für ein ML-basiertes System erst zur Laufzeit herausstellt, dass Grundannahmen zur Sicherheit nicht mehr erfüllt sind, sodass von einem bereits im Betrieb befindlichen System eine grundsätzliche Gefahr ausgeht. Diese Gefahren müssen zur Laufzeit identifiziert, kommuniziert und ggfs. aktiv gemindert werden können.

***Beispiel**: KI/ML-basierten Software* in a*utonomen Fahrzeugen wird darauf angewiesen sein, die Absicht eines Fußgängers aus der Analyse von Gesten und Bewegungsmustern vorhersagen zu können. Menschen nutzen solche Vorhersagen, um einschätzen zu können, ob ein Fußgänger auf dem Bürgersteig verbleibt oder auf die Straße läuft. Das Erkennen solcher Gesten und Bewegungsmuster ist ein komplexer Erkennungsprozess, der auf einer Vielzahl von unterschiedlichen Merkmalen basiert, die außerhalb der Kontrolle des Fahrzeugherstellers liegen und häufig gesellschaftlichen Einflüssen unterliegen. Änderungen in der Ausgestaltung der Merkmale kann einen entscheidenden Einfluss auf die Erkennungsgenauigkeit haben. So kann beispielsweise durch das vermehrte Tragen einer Maske, wie wir es aktuell in der Coronapandemie erleben, die Fähigkeit eines Vorhersagemechanismus dahingehend beschränkt werden, dass durch das Tragen der Masken die Blickrichtung einer Person nicht mehr richtig abgeleitet wird und die Absicht der Person deutlich schlechter prognostiziert werden kann. Die Kombination aus fehlender Transparenz in der Verwendung von Merkmalen sowie der Möglichkeit einer Verschiebung der Merkmalsrepräsentanz zur Laufzeit, führt dazu, das der OEM in die Lage versetzt werden muss, zu erkennen, dass sich die Umgebungsbedingungen für den Betrieb des Fahrzeuges geändert haben. .*

Risiko: Ohne Laufzeitkontrollen und ohne den aktiven Versuch nicht gelernte Szenarien gezielt zu erkennen, werden systematische Fehler zugelassen, die die Sicherheit des Systems gefährden.

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/3)



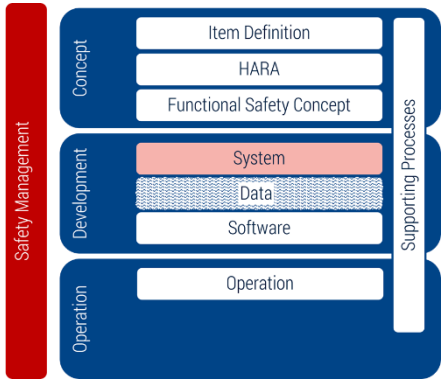
Maßnahmen:

Etablierung von Methoden und Systemen zum online Monitoring und zur online Überwachung von ML-Komponenten in Fahrzeugen mit dem Ziel, Auffälligkeiten und Leistungs-abweichungen in Bezug auf Verteilungsverschiebungen, unbekannte Eckfälle und anderen Risiken zu erkennen und zu beseitigen. Das Sicherheitskonzept muss die folgenden Aspekte berücksichtigen:

- Monitoring
- Fehlererkennung und evtl. Minderung der Auswirkungen im Betrieb
- Updatefähigkeit und -geschwindigkeit der Softwarekomponenten mit ML-Anteil.

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Im Rahmen der Erstellung des technischen Sicherheitskonzepts ist nachzuweisen, dass die Anforderungen an Monitoring und Fehlererkennung ausreichend sind, um ML-spezifische Auffälligkeiten und Leistungsabweichungen zur Betriebszeit erkennen zu können und ggfs. in Echtzeit darauf reagieren zu können. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

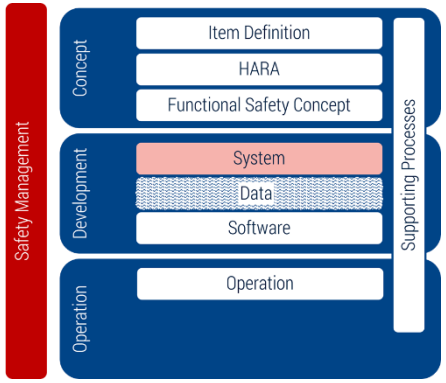
- 4.6.5.1. Technical safety requirements specification (ISO 26262)
- 4.7.5.1. Technical safety concept (ISO 26262)
- 4.7.5.2. System design specification (ISO 26262)
- 8.5.1 Specification of SOTIF measures (SOTIF)
- 4.6.5.5 Specification of requirements for production, operation, service and decommissioning (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- *04-ML30 Initiale Lösungsarchitektur*
- *15-ML08 Risikoreport ML*
- *08-ML13 Beobachtungs- und Wartungsplan*

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/3)



Herausforderung: Die Qualität der Trainingsdaten ist ein entscheidender Faktor für das Erreichen der Qualitätseigenschaften und damit auch für die Sicherheit einer ML-Komponente. Standards und Best Practices in der Softwareentwicklung und funktionaler Sicherheit adressieren aktuell weder Fragen nach der Datenqualität, noch werden Anforderungen an den Datenlebenszyklus formuliert.

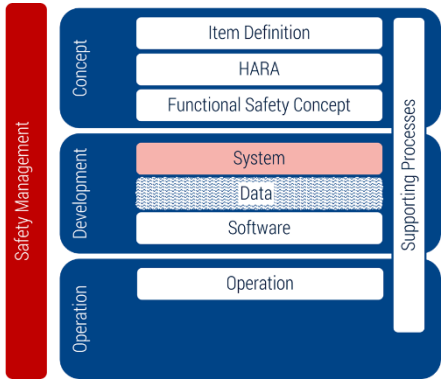
***Beispiel**: Weder ISO26262 noch die SOTIF beschreiben einen Datenqualitätslebenszyklus sowie dezidierte Datenqualitätsmaßnahmen, die als Grundlage für die Bewertung der Datenbereitstellung und -aufbereitung im Kontext sicherheitskritischer Systeme herangezogen werden kann. Insbesondere für das autonome Fahren müssen Datenbestände verwaltet, geprüft und qualitätsgesichert werden, die in ihrer Menge jedes bisher gekannte Maß übersteigen. Es kann davon ausgegangen werden, dass allein für das Training der KI/ML-basierten Software Datenmaterial vorhanden sein muss, welches einer gefahrenen Strecke von 6 Milliarden Kilometern entspricht. Nur ein kleiner Teil der Daten werden aufgezeichnet werden können. Ein Großteil wird synthetisch erzeugt werden müssen.*

Risiko: Es besteht die Gefahr, dass datenbezogene Fehler bzw. unvollständige, veraltete oder irrelevante Daten dazu führen, das sicherheitsrelevante Eigenschaften eines Systems (Bias-Freiheit, Robustheit, Generalisierungsfähigkeit, Genauigkeit etc.) nicht erreicht und nicht nachgewiesen werden können. Grundsätzlich ist sicherzustellen, dass ausreichend Daten zur Verfügung stehen, dass Sonderfälle und Ausnahmesituation ausreichend durch Daten abgedeckt sind, dass die Daten und die Datenaufbereitung in angemessener Qualität erfolgen und die verfügbare Datenverteilung eine gute Annäherung an die reale Welt darstellt.

Technisches Sicherheitskonzept

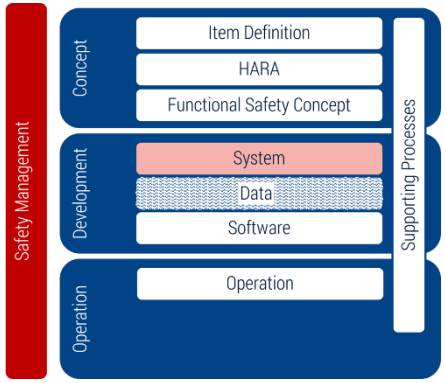
Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/3)

Maßnahmen: Integration des Datenlebenszyklus in den Sicherheitslebenszyklus inklusive der verschiedenen Aktivitäten wie Spezifikation von Datenanforderungen, dem Design der Daten und der Datenaufbereitung, der Festlegung datenbezogener Qualitäts- und Sicherheitsmetriken, der Verifikation und Validierung der Daten sowie der Datenpflege, der Datenversionierung und der Datenerweiterung.



Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: *Im Rahmen der Erstellung des technischen Sicherheitskonzeptes ist nachzuweisen, welche Rolle Daten in der Bereitstellung der sicherheitsrelevanten Funktionalität spielen und welche Maßnahmen erfolgen müssen, um eine ausreichende Datenqualität über den Lebenszyklus des Produkts sicherzustellen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:*

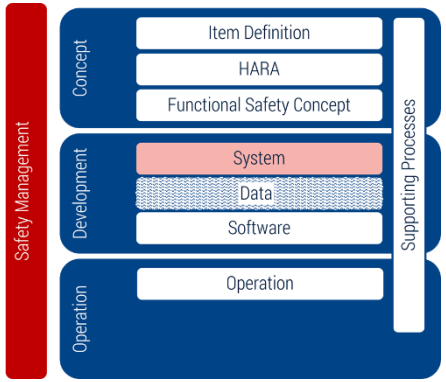
- *4.6.5.1. Technical safety requirements specification (ISO 26262)*
- *4.7.5.1. Technical safety concept (ISO 26262)*
- *4.7.5.2. System design specification (ISO 26262)*
- *8.5.1 Specification of SOTIF measures (SOTIF)*

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- *15-ML30 Report zur Datenauswahl*
- *17-ML30 Anforderungen an die Datenpipeline*
- *15-ML34 Report zur Normierung und Standardisierung der Daten*
- *15-ML31 Report zur Datenvalidierung und Bereinigung*
- *15-ML32 Report zur Datenauszeichnung und Evaluierung der Daten*
- *08-ML30 Plan zum Management der Trainingsdaten*

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/3)



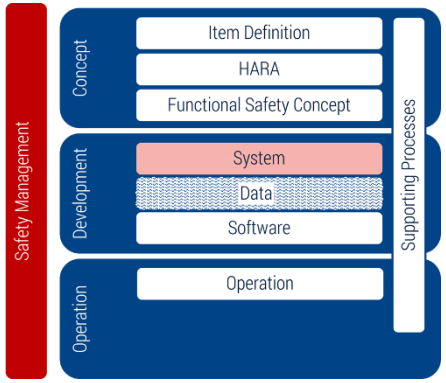
Herausforderung: Neuronale Netze, insbesondere DNNs, sind sog. Black-Box-Systeme, die sich in ihrer konkreten technischen Umsetzung durch menschliche Beobachter nur schwer verstehen lassen. Anstatt dass, wie im Fall der klassischen Programmierung, das Systemverhalten explizit kodiert wird, lernt beim ML der Algorithmus auf Basis einer großen Anzahl von Beispielen, die eine Abbildung der Eingabedaten auf die gewünschte Ausgabe darstellen. Transparenz und Interpretierbarkeit in der KI umfassen sowohl die Erklärbarkeit einer Entscheidung als auch das Vertrauen in das System und seine Entstehung [8][9]. Während die Interpretierbarkeit der Grad ist, in dem ein Mensch die Ursache einer Entscheidung verstehen kann [10], wird das Vertrauen in ein System durch das Verständnis des Systems selbst, seiner Einsatzumgebung sowie der Entwicklung des Systems gewonnen.

***Beispiel**: Auch wenn eine KI/ML-basierten Software zur Objekterkennung in der Lage ist, Fußgänger am Straßenrand i.d.R. richtig zu erkennen, ist häufig nicht nachzuvollziehen, auf Basis welcher Merkmale in den Daten, die Objekterkennung entschieden wurde. Wir können also i.d.R. nicht belegen, dass ein solches System letztendlich unser Verständnis dessen, was ein Fußgänger ist, tatsächlich auch realisiert. Beispielsweise haben Baker et. al. gezeigt, dass die Oberflächentextur bei der Klassifizierung durch tiefe Netzwerke eine größere Rolle spielen als bei der menschlichen Erkennung und dass dieselben Systeme die Gesamtform eines Objekts weniger stark verwenden als es ein Mensch tun würde. Wichtig wird ein solches Wissen aber genau dann, wenn sich Änderungen im Merkmalsraum ergeben und Vorhersagen dazu getroffen werden müssen, welche Bedeutung die Änderungen beispielweise für die Sicherheit des Systems haben.*

Risiko: Risiken bezüglich der Sicherheit ergeben sich hier aus der Abhängigkeit der Systemfunktionalität von einem algorithmischen Entscheidungssystem, das nicht einmal die Entwickler wirklich verstehen. Ein solches Verständnis ist aber notwendig, um einerseits unerwartetes Verhalten, dass zu sicherheitsrelevanten Fehlfunktionen führen kann, analysieren und systematisch eliminieren zu können und notwendig dafür, eine nachprüfbare und nachvollziehbare Sicherheitsargumentation liefern zu können. Dies ist insbesondere im Hinblick auf eine juristische Bewertung von Zulassungs- und Prüfprozessen wie auch bei der Bewertung von Schadensfällen eines algorithmischen Entscheidungssystems von großer Bedeutung und betrifft im Besonderen Verfahren des tiefen Lernens, die aufgrund ihres Black-Box-Charakters besonders intransparent sind. Eine detaillierte Kenntnis des Entscheidungsprozesses und seiner Parameter ist darüber hinaus notwendig, um Fehlerursachen und Verbesserungsmöglichkeiten des Systems untersuchen zu können.

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/3)

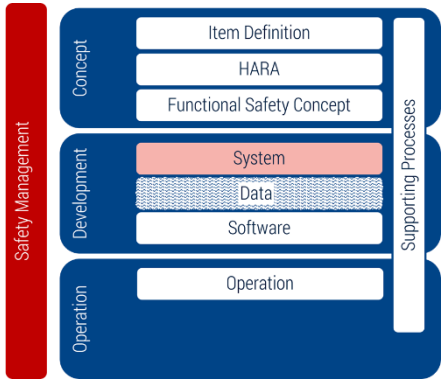


Maßnahmen: Integration von Methoden und Verfahren zur Interpretierbarkeit bzw. Erklärbarkeit von Modellen als Sicherheitsanforderungen in das Sicherheitskonzept. Die Schaffung von Interpretierbarkeit und Transparenz im tiefen Lernen ist ein heterogenes Forschungsfeld, mit unterschiedlichen Ansätzen mit dem übergreifenden Ziel den Black-Box-Charakter des tiefen Lernens zu überwinden. Grundsätzlich lassen sich die folgenden Ansätze unterscheiden:

- Verwendung von direkt interpretierbaren Modellen. Solche Modelle erlauben es einem menschlichen Benutzer, die Herkunft oder die Bedeutung einer Entscheidung direkt zu verstehen. Hierzu zählen nach Burton et al. [3] lineare oder logistische Regression, der k-nearest neighbor Algorithmus, ·Regelbasierte Verfahren, Bayesian Classifiers, SVM, Gaussian Processes, Markov-Entscheidungsprozess oder Modelle, wo ein menschlicher Akteur in die Entscheidungsfindung eingebunden ist.
- Post-hoc-Erklärungen erlauben es, Erklärungen für die Entscheidung eines Algorithmus im Nachgang zur Entscheidung liefern zu können. Wie genau dies geschieht, hängt von der jeweiligen Implementierung ab. Beispiele hierfür sind Attributions- und Saliency-Verfahren.
- Mit sog. beispielbasierte Erklärungen werden "Prototypen" für bestimmte Klassen von Entscheidungen bereitgestellt, die für den menschlichen Beobachter verständlich sind. Ziel ist es, Eigenschaften für die Ähnlichkeit von Daten und deren Bedeutung für den Entscheidungsprozess zu identifizieren und zu explizieren, sodass dem Beobachter eine oder mehrere "typische" Darstellungsinstanzen für eine bestimmte Klasse zur Verfügung gestellt werden können. Dadurch kann die Qualität und Vollständigkeit des Lernprozesses bewertet werden.
- Als sog. Surrogatmodelle werden interpretierbare Modelle bezeichnet, die so trainiert werden, dass sie dem eigentlichen Entscheidungsalgorithmus ähnlich sind und somit als Erklärungen für diese verwendet werden können.

Technisches Sicherheitskonzept

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Im technischen Sicherheitskonzept ist nachzuweisen, dass technische Maßnahmen ergriffen wurden, um die algorithmische Entscheidung ausreichend verstehen zu können, sodass eine nachvollziehbare Sicherheitsargumentation sowie die Analyse von potenziellen Betriebsfehler möglich wird. Eine entsprechende Dokumentation der Maßnahmen erfolgt im Rahmen der folgenden Arbeitsprodukte:

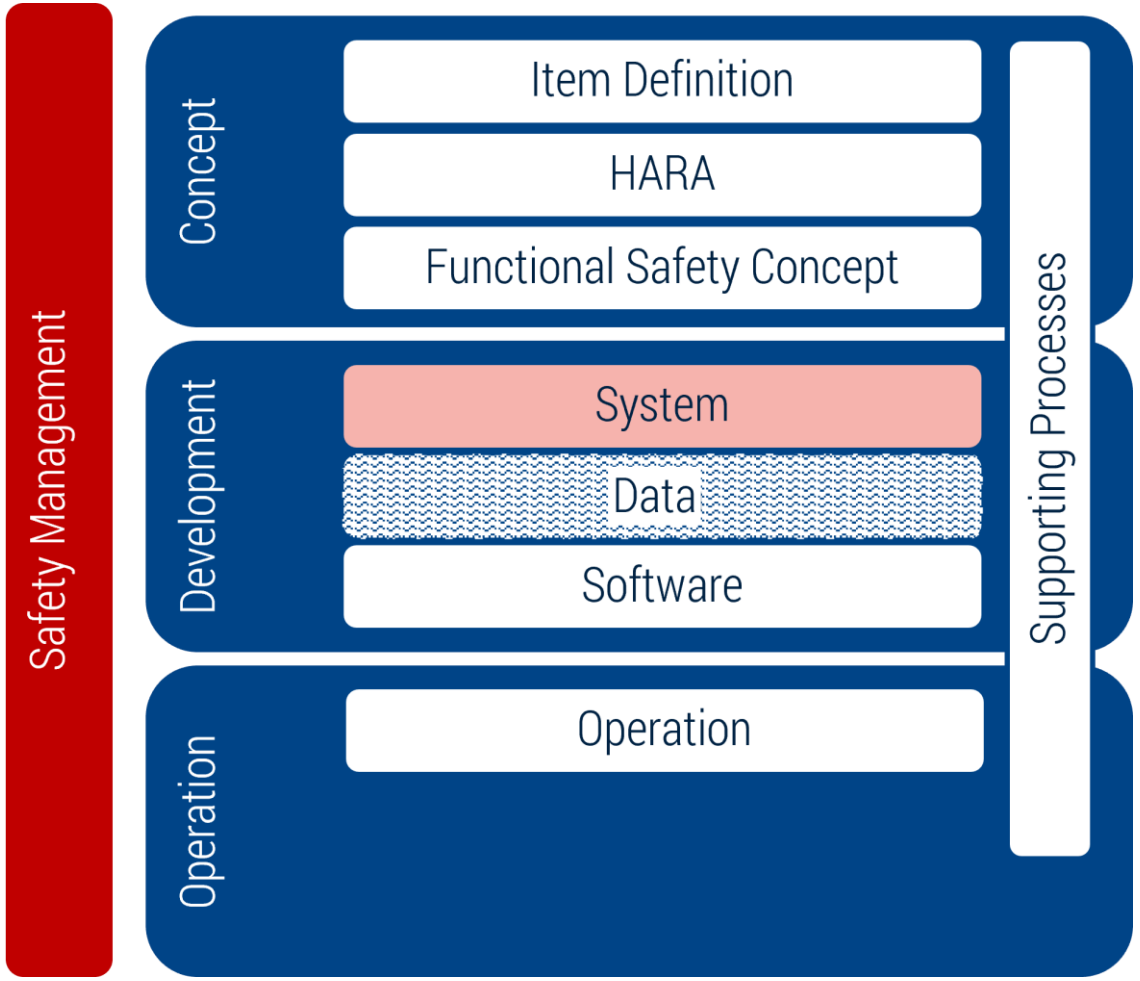
- *4.6.5.1. Technical safety requirements specification (ISO 26262)*
- *4.7.5.1. Technical safety concept (ISO 26262)*
- *4.7.5.2. System design specification (ISO 26262)*
- *8.5.1 Specification of SOTIF measures (SOTIF)*

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- *19-ML31 Modellierungs- und Trainingsstrategie*
- *15-ML51 Bericht zur Nachvollziehbarkeit der Entscheidungsfindung des ML-Modells*

System -und Objektintegration und -prüfung

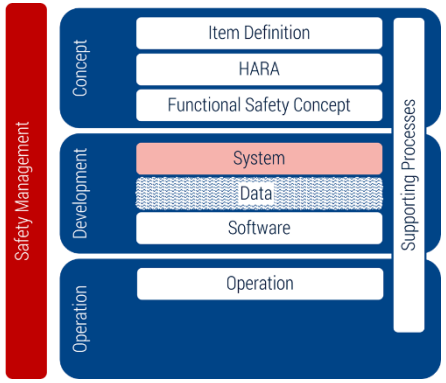
Aufgabe dieser Phase ist es, die Integrationsschritte zu definieren, die Systemelemente vollständig in das System zu integrieren und die definierten Sicherheitsmaßnahmen, die sich aus den Sicherheitsanalysen auf der Systemarchitekturebene ergeben, auf korrekte Umsetzung zu verifizieren. Außerdem wird der Nachweis erbracht, dass die integrierten Systemelemente ihre Sicherheitsanforderungen gemäß dem Systemarchitekturentwurf erfüllen.



zurück zur Navigation

System -und Objektintegration und -prüfung

Absicherung der Betriebsphase (1/3)



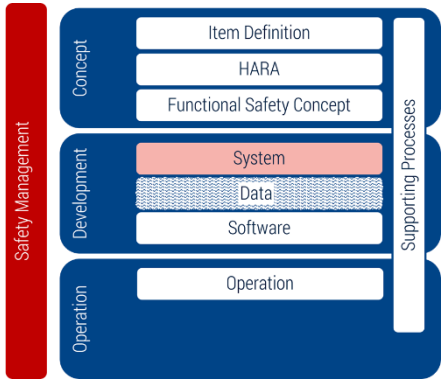
Herausforderung: Die Ergebnisse einer ML-Komponente sind abhängig von einer stabilen Betriebsumgebung (eine Betriebsumgebung, die der Umgebung entspricht, in der die Komponente trainiert wurde). Das bezieht sich sowohl auf die Umgebung des Gesamtsystems als auch auf die direkte technische Umgebung, d.h. der soft- und hardwaretechnischen Einbettung in das Gesamtsystem.

***Beispiel**: Toleranzen bei der Positionierung von Kamerasysteme in den Serienfahrzeugen führen dazu, dass die Leistung einer Objekterkennungssoftware nicht mehr den Erkennungsleistungen im Trainings- und Prüfprozess entspricht.*

Risiko: Abweichungen der Trainings- von der Betriebsumgebung können dazu führen, dass die Leistung einer ML-Komponente nicht den Annahmen aus dem Sicherheitskonzept entspricht. Zu möglichen Abweichungen gehören der Einsatz von Sensoren mit anderen Charakteristiken, Änderungen und Abweichungen in der Positionierung der Sensoren am Fahrzeug aber auch Änderungen in der Sensornachbearbeitung sowie der Interpretation der ML-Komponentenergebnisse.

System -und Objektintegration und -prüfung

Absicherung der Betriebsphase (2/3)

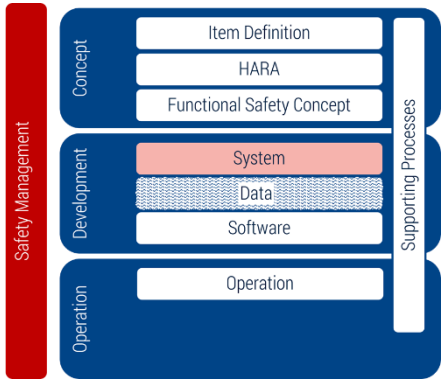


Maßnahmen:

- Identifikation der Stabilitätskriterien einer ML Anwendung bzgl. ihrer technischen Umgebung.
- Definition von Metriken und Toleranzschwellen für die Qualitätseigenschaften von Sensoren, ihrer Positionierung etc. im Hinblick auf die identifizierten Stabilitätskriterien einer ML Anwendung.
- Definition von Integrationsanforderungen in Bezug auf die Entscheidungsgüte einer algorithmischen Entscheidung.
- Verifikationsverfahren zur Identifikation von Abweichungen zwischen der realen Umgebung eines ML-Modells und der erwarteten Umgebung (ODD Monitoring, Distributional Shift Monitoring) sowie der Entscheidungsgüte eines algorithmischen Entscheidungssystems.

System -und Objektintegration und -prüfung

Absicherung der Betriebsphase (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Im Rahmen der Integration und sowie des Integrationstests ist nachzuweisen, dass die Varianz in Kalibrierung und Qualität der Hardware keine negativen Auswirkungen auf die sicherheitskritische Funktionalität eines algorithmischen Entscheidungssystems hat. Eine entsprechende Dokumentation der Prüfkriterien, des Prüfvorgehens sowie der Prüfergebnisse erfolgt im Rahmen der folgenden Arbeitsprodukte:

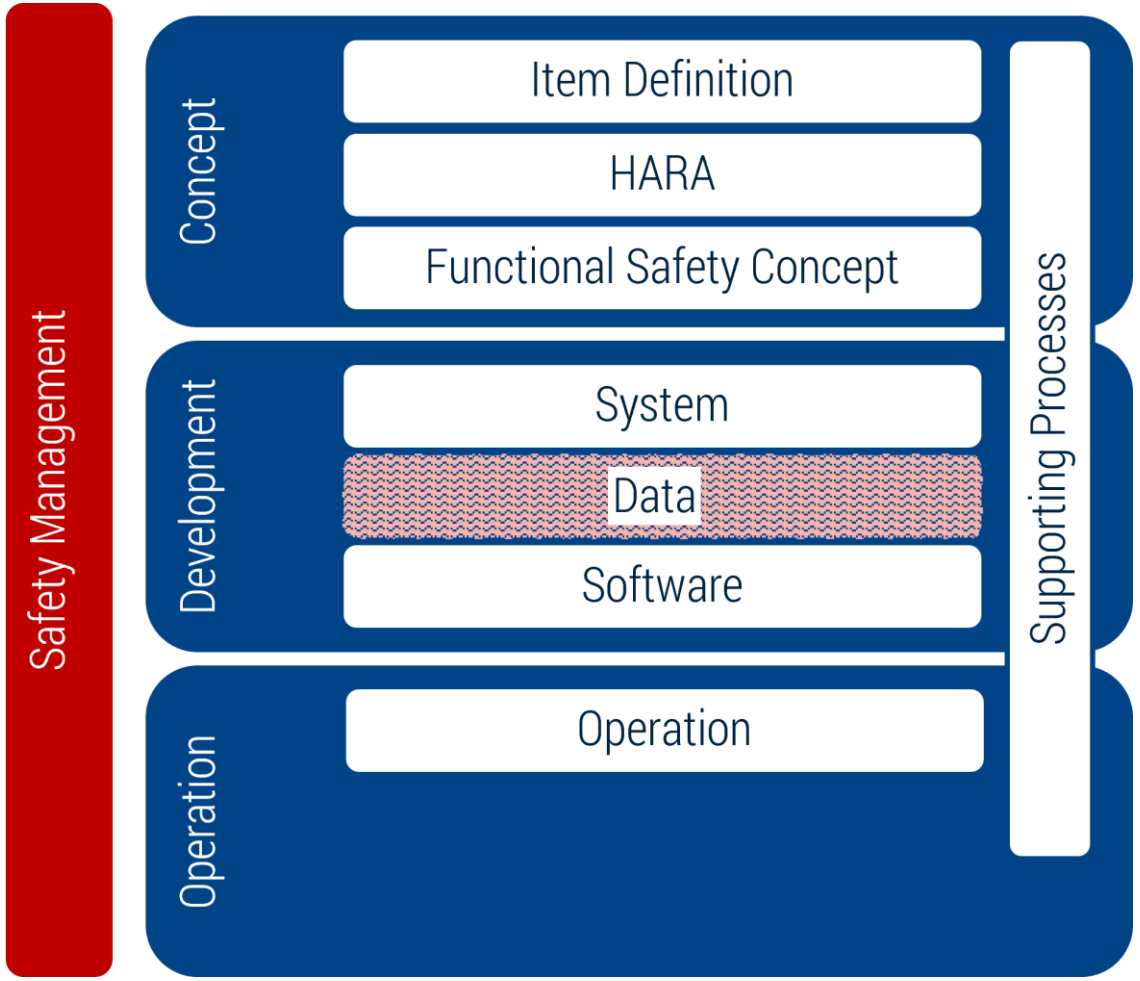
- *4.7.5.1 Integration and test strategy (ISO 26262)*
- *4.7.5.2 Integration and test report (ISO 26262)*

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- *08-ML52 Testplan für die Modellintegration*
- *08-ML50 Testspezifikation*
- *13-ML50 Testergebnisse und Messungen*
- *15-ML52 Test- und Verifikationsbericht in der Integration*

Produktentwicklung Daten

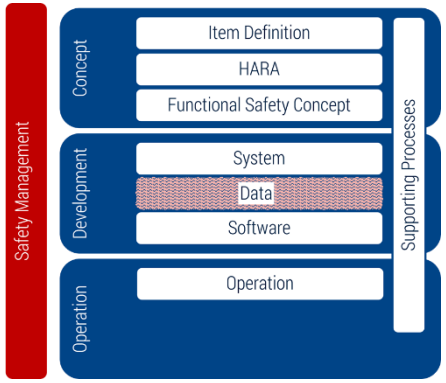
Ziel dieser Phase ist es, die aus dem technischen Sicherheitskonzept und dem Systemarchitekturentwurf abgeleiteten Daten-Sicherheitsanforderungen zu spezifizieren, die Infrastruktur der Datenaufbereitung abzusichern und Maßnahmen zur Qualitätssicherung der dem algorithmischen Entscheidungssystem zugrundeliegenden Daten zu spezifizieren und in ihrer Wirksamkeit zu verifizieren. Eine solche Phase gibt es aktuell weder in der ISO 26262 noch in anderen Vergleichbaren Sicherheitsstandards. Die besondere Bedeutung von Daten für das ML und die Auswirkungen der Datenqualität auf die Leistungsfähigkeit von ML-Systemen einerseits und die Komplexität, die hohen Aufwände sowie die prozessualen Besonderheiten der Bereitstellung und Pflege geeigneter Trainingsdaten andererseits, rechtfertigt unser Meinung nach aber die explizite Ausformulierung einer solchen Phase. Da es bisher keine Referenz zu einer solchen Phase in relevanten Sicherheitsstandards gibt, können wir in den Abschnitten „**Anforderungen bzgl. Dokumentations- und Nachweispflicht**“ nicht auf die Arbeitsprodukte normativer Standards verweisen.



zurück zur Navigation

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



Herausforderung: Ungeeignete Datenerfassungstechniken, nicht vertrauenswürdige Datenquellen oder unvollständige Daten.

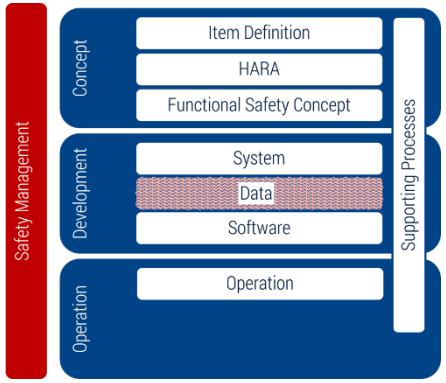
***Beispiel**: Wenn die Daten für das Training einer KI/ML-basierten Software zur Objekterkennung größtenteils aus Mitteleuropa stammen, wird das Fahren in einer vollständig verschneiten Umgebung nur unzureichend adressiert .*

Risiko: Die in ML verwendeten Daten müssen beschafft, erzeugt bzw. ausgewählt werden. Bereits im Auswahlprozess müssen grundlegende Anforderungen bzgl. der Qualität und Quantität der Daten berücksichtigt werden. In diesem Prozess werden die Daten erstmals genauer analysiert und es kann festgestellt werden, dass Daten fehlen, Messungen unzuverlässig waren oder die technische Infrastruktur zur Datenerfassung und -speicherung fehlerhaft ist. Es ist sinnvoll, bereits an dieser Stelle auf Qualitätsmängel zu reagieren und die Fehlerursachen in den vorangegangenen Prozessschritten zu beheben. Potenzielle Risiken ergeben sich wenn:

- Daten aus verschiedenen Quellen nicht korrekt abgeglichen werden.
- Daten nicht die notwendige Vielfallt und Repräsentativität besitzen, um den kompletten Ereignisraum abdecken zu können.
- Die Art und Anzahl der benötigten Merkmale in den Daten unzureichend repräsentiert sind.
- Daten nicht adäquat gespeichert und mit sinnvollen Metadaten versehen werden.
- Daten manipuliert wurden (data poisening)

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



Maßnahmen:

- Rückgriff auf vertrauenswürdige Datenquellen (z.B. zertifizierte Datenquellen)
- Verwendung bereits vordefinierter Datensätze (z.B. für Standardaufgaben bereits qualifizierter Datensätze)
- Anwendung von Verfahren zur Datenqualitätssicherung (z.B. Normalisierung, Data Cleaning, etc.)
- Verwendung von zuverlässigen und geprüften (qualifizierten) Datenmanagementplattformen

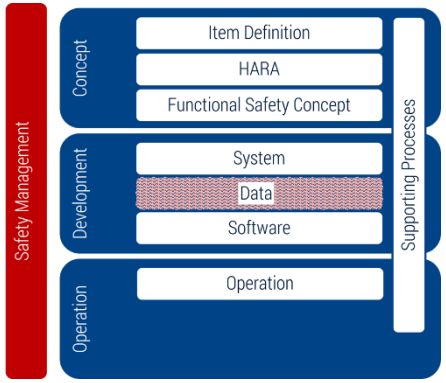
Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Es ist nachzuweisen, dass die Daten aus vertrauenswürdigen Quellen stammen, nicht manipuliert werden konnten und können und in der notwendigen Qualität und Repräsentanz über den gesamten Lebenszyklus einer ML-basierten Anwendung zur Verfügung stehen. Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 15-ML30 Report zur Datenauswahl
- 17-ML30 Anforderungen an die Datenpipeline
- 15-ML34 Report zur Normierung und Standardisierung der Daten
- 15-ML31 Report zur Datenvalidierung und Bereinigung

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



Herausforderung: Mangel an angemessenen Datenaufbereitungsprozessen/ ungeeignete Datenaufbereitung.

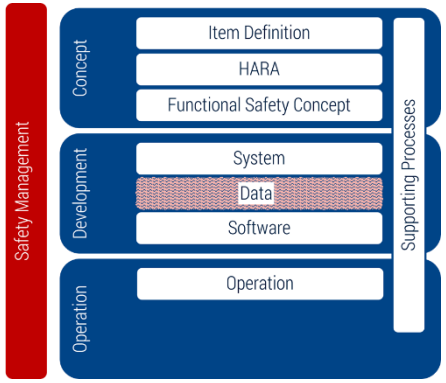
***Beispiel**: Die Daten für eine KI/ML-basierte Software zur Objekterkennung sind unzureichend ausgezeichnet, sodass das System Erwachsene und Kinder nicht sicher unterscheiden kann. Die fehlende Differenzierung führt dazu, dass das System beispielsweise das potentiell spontanere Verhalten von Kindern nur ungenügend berücksichtigen kann.*

Risiko: Im Anschluss an die Datenerfassung werden die Daten in einem iterativen Prozess bereinigt, verfeinert, ergänzt und ggfs. ausgezeichnet. Die Qualität der Daten muss entlang eines solchen Prozesses möglichst dahingehend verbessert werden, dass sie einerseits Kriterien wie Vollständigkeit, Korrektheit, Aktualität etc. aufweisen und andererseits in einer der weiteren Verarbeitung zuträglichen Form technisch aufbereitet, gespeichert und verwaltet werden. Mögliche Fehlerquellen mit Sicherheitsbezug sind:

- Datenaufbereitung ungenügend bzw. unvollständig, sodass datenbedingte Fehler nicht erkannt und ausgemerzt werden konnten.
- Ungeeignete oder falsche Beschriftung (Ungeeignete Ground Truth)

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



Maßnahmen:

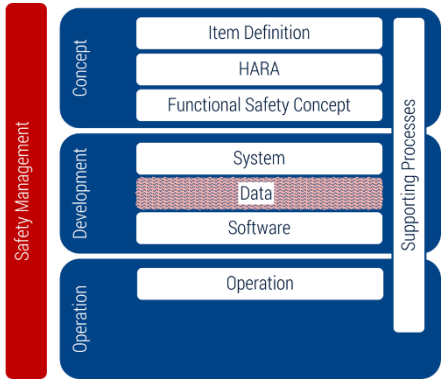
- Verwendung von zuverlässigen und geprüften (qualifizierten) Datenpipelines zur Datenaufbereitung.
- Systematische Anwendung von StoA Verfahren zur Datenqualitätssicherung und -verbesserung.
- Einführung von Prozessen zur Daten- und Metadaten Qualitätssicherung.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass im Zuge der Datenaufbereitung die Qualität der Daten und ihre Eignung für die Aufgabe sichergestellt werden. Hierzu zählt die Dokumentation der durch-geführten Datenaufbereitungsprozeduren (Datenbereinigung, Datenverbesserung) sowie die Dokumentation der Qualitätssicherungsprozeduren (Vollständigkeitsprüfungen, Korrektheitsprüfungen). Beim überwachten Lernen ist insbesondere der Prozess der Datenauszeichnung hinsichtlich der gewählten Label und ihrer Korrektheit zu dokumentieren. Es ist zu dokumentieren, dass alle Anforderungen an die Daten auch durch die bereitgestellten Daten realisiert werden (Traceability zwischen den Datenanforderungen, Datensätzen und den durchgeführten Prüfungsergebnissen). Grundlage für die Dokumentation bilden die folgen-den Arbeitsprodukte aus dem ML-Lebenszyklus:

- 17-ML11 Anforderungsspezifikation ML-Modell und Daten
- 5-ML34 Report zur Normierung und Standardisierung der Daten
- 15-ML31 Report zur Datenvalidierung und Bereinigung
- 17-ML31 Spezifikation der Datenauszeichnung
- 15-ML32 Report zur Datenauszeichnung und Evaluierung der Daten

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



Herausforderung: Algorithmische Entscheidungssysteme unterliegen i.d.R. einem Bias. Ursache eines solchen Bias ist i.d.R. eine durch gesellschaftliche Prozesse und/oder technische Prozesse entstandene Abweichung zwischen der Realwelt und der durch die Verteilung der Trainingsdaten repräsentierten Welt.

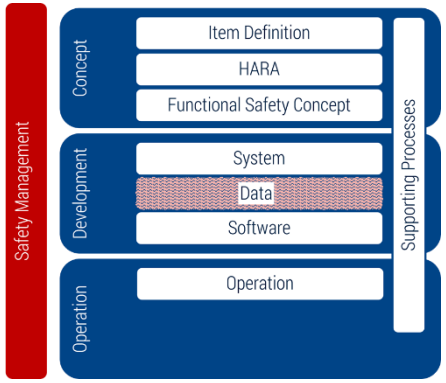
***Beispiel**: Menschen, die für ihre Fortbewegung auf Krücken angewiesen sind, werden beim Training nicht berücksichtigt und später im System nicht als Fußgänger wahrgenommen.*

Risiko: Bias führt zu einer unterschiedlichen Leistung eines ML-Algorithmus bzgl. objektiver Kriterien der Entscheidungsfindung, sodass bestimmte häufig vorkommende Ereignisse, Personen, Objekte besser erkannt bzw. anders behandelt werden als vermeintlich selten vorkommende Ereignisse, Personen und Objekte. Im Kontext Sicherheitskritischer Systeme kann eine solche variierende Leistungsfähigkeit dazu führen das:

- Die Sicherheit eines Systems nicht unter allen Betriebssituationen gegeben ist (siehe auch Verteilungsverschiebung).
- Bestimmte Bevölkerungsgruppen in kritischen Situationen einem höheren Sicherheitsrisiko ausgesetzt sind als andere.

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



Maßnahmen:

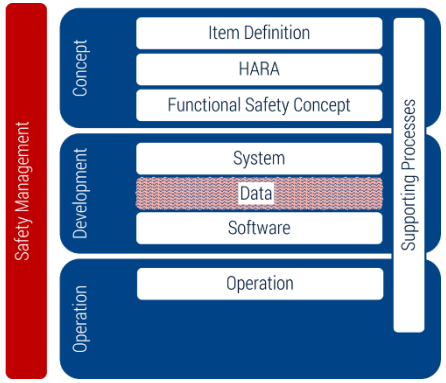
- Explizite Berücksichtigung der Qualitätseigenschaft „BIAS-Freiheit“ in der Sicherheitsbewertung algorithmischer Entscheidungssysteme.
- Aufnahme von Metriken und Anforderungen zur BIAS-Freiheit
- Verwendung expliziter Prüfverfahren, die sowohl den Trainingsdatensatz wie auch den algorithmischen Entscheider auf BIAS Freiheit prüfen.
- Im Rahmen der Forschung entwickelte Werkzeuge und Verfahren, um Diskriminierung und BIAS in maschinellen Lernmodellen während des gesamten Lebenszyklus von KI-Anwendungen zu untersuchen, zu melden und zu entschärfen wie beispielsweise das von IBM entwickelte Werkzeug AI-Fairness 360 werden aktuell im Be-reich von Finanz- und Medizinanwendungen erforscht, eine Anwendung im Bereich sicherheitskritischer Verkehrssysteme ist nicht bekannt.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass die für das Training ausgewählten Daten keinem BIAS unterliegen. Maßnahmen zur Prüfung und Vermeidung von BIAS in der Datenbereitstellung sind zu Dokumentieren. Grundlage für die Dokumentation bilden die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 15-ML30 Report zur Datenauswahl
- 15-ML32 Report zur Datenauszeichnung und Evaluierung der Daten
- 15-ML52 Bericht zum Nachweis der Fairness der Entscheidungsfindung des ML-Modells

Produktentwicklung Daten

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/2)



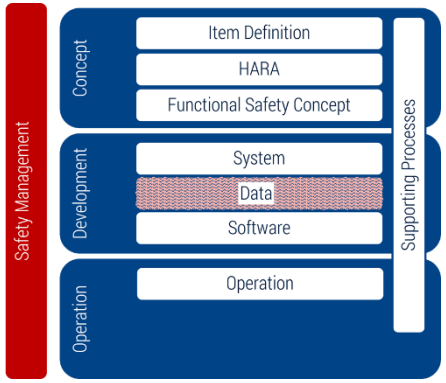
Herausforderung: Kritische Ereignisse sind in Trainingsdaten i. d. R. unterrepräsentiert. Es muss sichergestellt werden, dass diese Ereignisse während des Trainings und der Validierung ausreichend berücksichtigt und abgedeckt werden.

***Beispiel**: Beinahe Unfälle und andere kritische Verkehrssituationen sind in den Trainingsdaten für eine KI/ML-basierte Software potentiell unterrepräsentiert, da sie wesentlich seltener auftreten als Standardsituationen. Grundsätzlich kann davon ausgegangen werden, dass 80% der relevanten Szenarien einfach zu ermitteln und zu operationalisieren sind während für die fehlenden 20% ein extrem hoher Aufwand notwendig ist.*

Risiko: Ein ML-Modell kann Entscheidungen nur auf der Basis bereits "gesehener Daten" treffen. Sind kritische Szenarien in den Trainingsdaten nicht berücksichtigt, wird auch eine ML-Anwendung diese Szenarien nicht richtig verarbeiten können.

Produktentwicklung Daten

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/2)



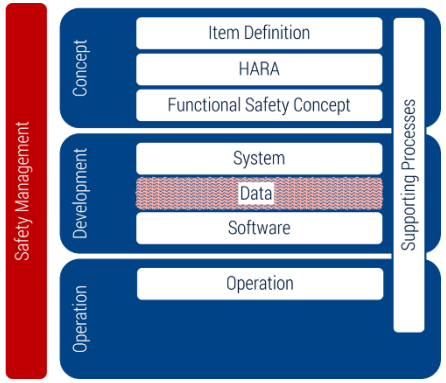
Maßnahmen: Einführung von Methoden und Strategien, die Vollständigkeit von Trainingsdaten zu messen und zu vervollständigen. Ontologien können verwendet werden, um den Aufbau und die Genese von Szenarien auf Basis einer formalisierten Wissensrepräsentation zu verstehen. Mit Hilfe von Ableitungsregeln können dann auf dieser Basis konkrete Szenarien abgeleitet werden [49]. Darüber hinaus schlagen Cheng et al. [50] Metriken vor, mit denen sich Vollständigkeitsanalysen auf Szenarien durchführen lassen. Da für realistische Spezifikationen von Betriebsbedingungen die Überprüfung der Abdeckung aller Szenarien aufgrund der kombinatorischen Explosion nicht durchführbar ist, schlagen Sie eine Szenario-Abdeckungsmetrik vor, die eng mit den bestehenden Arbeiten zu kombinatorischen Tests, abdeckenden Arrays und deren quantitativer Erweiterung verbunden ist.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass bereits durch die Datenauswahl die Abdeckung aller relevanten Szenarien durch geeignete Trainingsdaten abgedeckt werden. Die Abdeckung ist nachvollziehbar zu dokumentieren (z.B. Traceability zwischen der Spezifikation der relevanten Szenarien und den Trainingsdatensätzen). Grundlage für die Dokumentation bilden die folgen-den Arbeitsprodukte aus dem ML-Lebenszyklus:

- 04-ML01 Domain Model
- 17-ML11 Anforderungsspezifikation ML-Modell und Daten:
- 15-ML30 Report zur Datenauswahl
- 15-ML32 Report zur Datenauszeichnung und Evaluierung der Daten

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



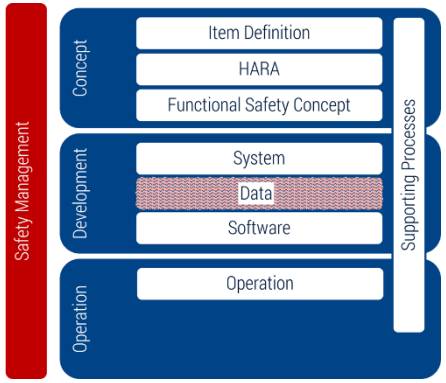
Herausforderung: Langfristige Sicherung der Datenqualität

***Beispiel**: Eine Objekterkennungssoftware basiert u.a. (und evtl. unbewusst) auf Merkmalen, deren Lebensdauer kürzer als die Betriebszeit des Fahrzeugs ist. Beispielsweise führt die Einführung neuer Fortbewegungsmittel (z.B. Hoover Board) dazu, dass sich Bewegungsmuster, die als Grundlage einer Fußgängererkennung verwendet werden, ergänzt werden müssen.*

Risiko: Daten können veralten und über die Zeit ihre Bedeutung verlieren. Um über den gesamten Lebenszyklus einer ML-Anwendung sicherzustellen, dass auch für Systemaktualisierungen Daten in ausreichender Qualität zur Verfügung stehen, müssen die Daten über einen längeren Zeitraum gepflegt, gewartet und aktualisiert werden.

Produktentwicklung Daten

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



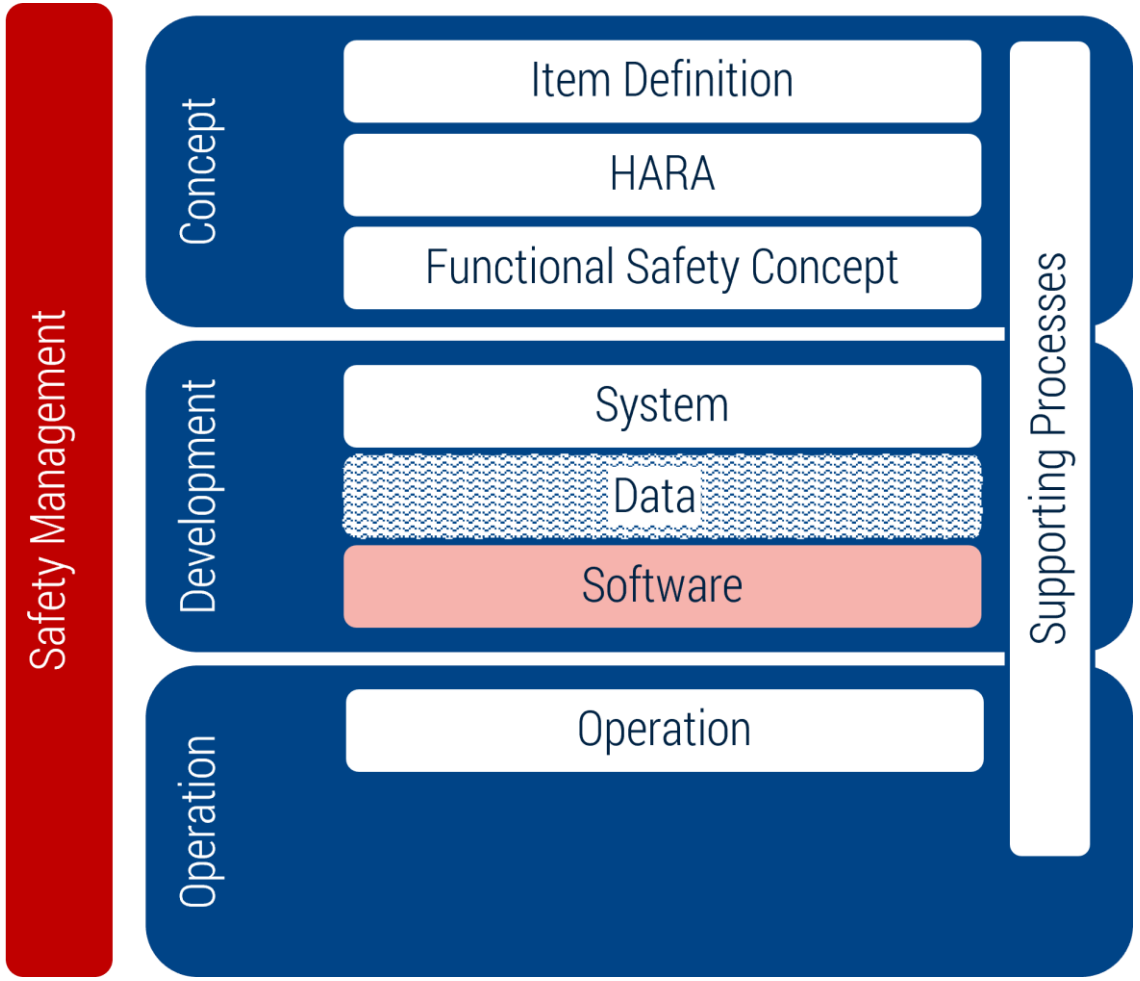
Maßnahmen: Eine der Möglichkeiten, eine gute Datenqualität zu erhalten, besteht darin, den Prozess der Datenerfassung und Datenaufbereitung mit Data Governance und einer rigiden Strategie zur Sicherung der Datenqualität zu untermauern.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass für die Trainingsdaten Prozesse etabliert worden sind, die eine längerfristige und aktive Pflege der Daten erlauben. Hierzu zählen u. a. Prozesse für ein kontinuierliches Change-Management und Versionsmanagement der Daten. Grundlage für die Dokumentation bilden die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML30 Plan zum Management der Trainingsdaten

Spezifikationen von Software-Sicherheitsanforderungen

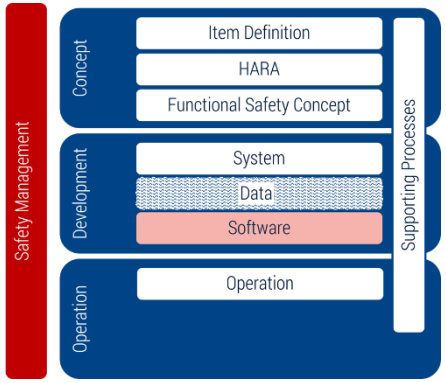
Ziel dieser Phase ist es, die aus dem technischen Sicherheitskonzept und dem Systemarchitekturentwurf abgeleiteten Software-Sicherheitsanforderungen zu spezifizieren und die Hardware-Schnittstelle (HSI) zu verfeinern, die in ISO 26262-4:2018, Klausel 6 festgelegt wurde. Des Weiteren wird überprüft, ob die Software-Sicherheitsanforderungen und die Anforderungen an die Hardware-Software-Schnittstellenspezifikation (HSI-Spezifikation) für die Softwareentwicklung geeignet sind und mit dem technischen Sicherheitskonzept und dem Systemarchitekturentwurf übereinstimmen.



zurück zur Navigation

Spezifikationen von Software-Sicherheitsanforderungen

Beherrschung datenintensiver Entwicklungsprozesse (1/3)



Herausforderung: ML ist ein hochgradig iterativer Optimierungsprozess, in dem Qualitäts- und Sicherheitsanforderungen aufeinander abgestimmt werden müssen. Aufgrund der komplexen Abhängigkeiten verschiedener Sicherheitsanforderungen ist davon auszugehen, dass

- widersprüchlicher Anforderungen existieren und in ihrer Realisierbarkeit gegeneinander abgewogen werden müssen.
- sich nicht alle Systemanforderungen in der Software umsetzen lassen.

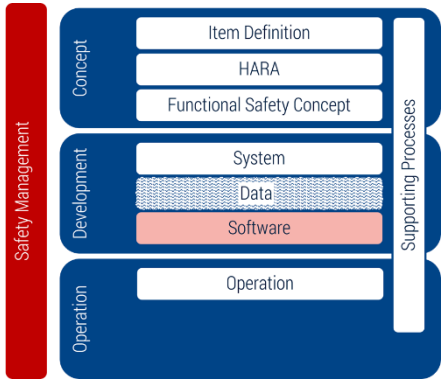
***Beispiel**: Genauigkeit und Robustheit eines neuronalen Netzes können konkurrierende Optimierungsziele sein, die Abhängig vom Anwendungskontext und unter Berücksichtigung potenzieller Risiken aufeinander abgestimmt werden müssen. Das bedeutet, dass im Trainingsprozess letztendlich entschieden werden muss, wie stark beispielsweise das Optimierungsziel Genauigkeit auf Kosten des Optimierungsziel Robustheit reduziert werden kann .*

Risiko: Iterative Produktverbesserung und Optimierung sind Prinzipien, die im Kontext sicherheitskritischer Systeme unüblich sind. Sicherheitsanforderungen und die darauf aufbauende Sicherheitsargumentation werden i.d.R. Top-Down abgeleitet und realisiert. Wünschenswert wäre die Möglichkeit, Sicherheitsanforderungen bereits als Optimierungsziele definieren zu können, sodass der Optimierungsansatz bereits bei der Definition der Softwareanforderungen berücksichtigt wird.

Ein iterativer Optimierungsansatz benötigt zusätzlich die Möglichkeit, System-Sicherheitsanforderungen auf Basis der Software-Sicherheitsanforderungen systematisch anpassen zu lassen. Erfolgt eine solche Bottom-Up Korrektur nicht in einer systematischen und regulierten Art und Weise, werden Sicherheitsargumentationen ungültig bzw. das System verletzt die Systemanforderungen.

Spezifikationen von Software-Sicherheitsanforderungen

Beherrschung datenintensiver Entwicklungsprozesse (2/3)

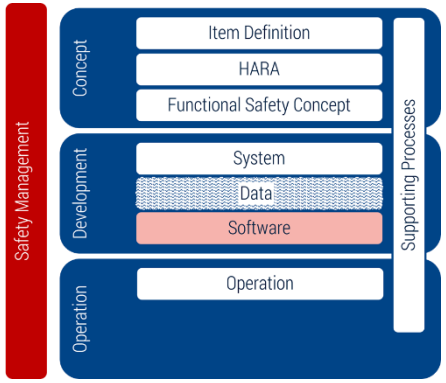


Maßnahmen:

- Zusammenarbeit zwischen ML-Experten und Domänenexperten, um System-Sicherheitsanforderungen in für die Formulierung des Trainingsziel, der Art der Modellierung, die Wahl der Algorithmen und Hyperparameter, sowie die Trainings-KPIs sinnvoll definieren zu können.
- Formulierung von technischen Software-Sicherheitsanforderungen als Optimierungsziele.
- Verfahren zur Kombination von Top-Down und Bottom-Up Sicherheitsargumentationen (siehe auch Ergebnisse des PEGASUS Projekts).

Spezifikationen von Software-Sicherheitsanforderungen

Beherrschung datenintensiver Entwicklungsprozesse (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es muss nachgewiesen werden, dass die Software-Sicherheitsanforderungen das grundsätzliche Vorgehen des ML berücksichtigen und das Trainingsziel, der Art der Modellierung, die Wahl der Algorithmen und Hyperparameter, sowie die Trainings-KPIs ausreichend definieren. Eine entsprechende Dokumentation der Maßnahmen erfolgt im Rahmen der folgenden Arbeitsprodukte:

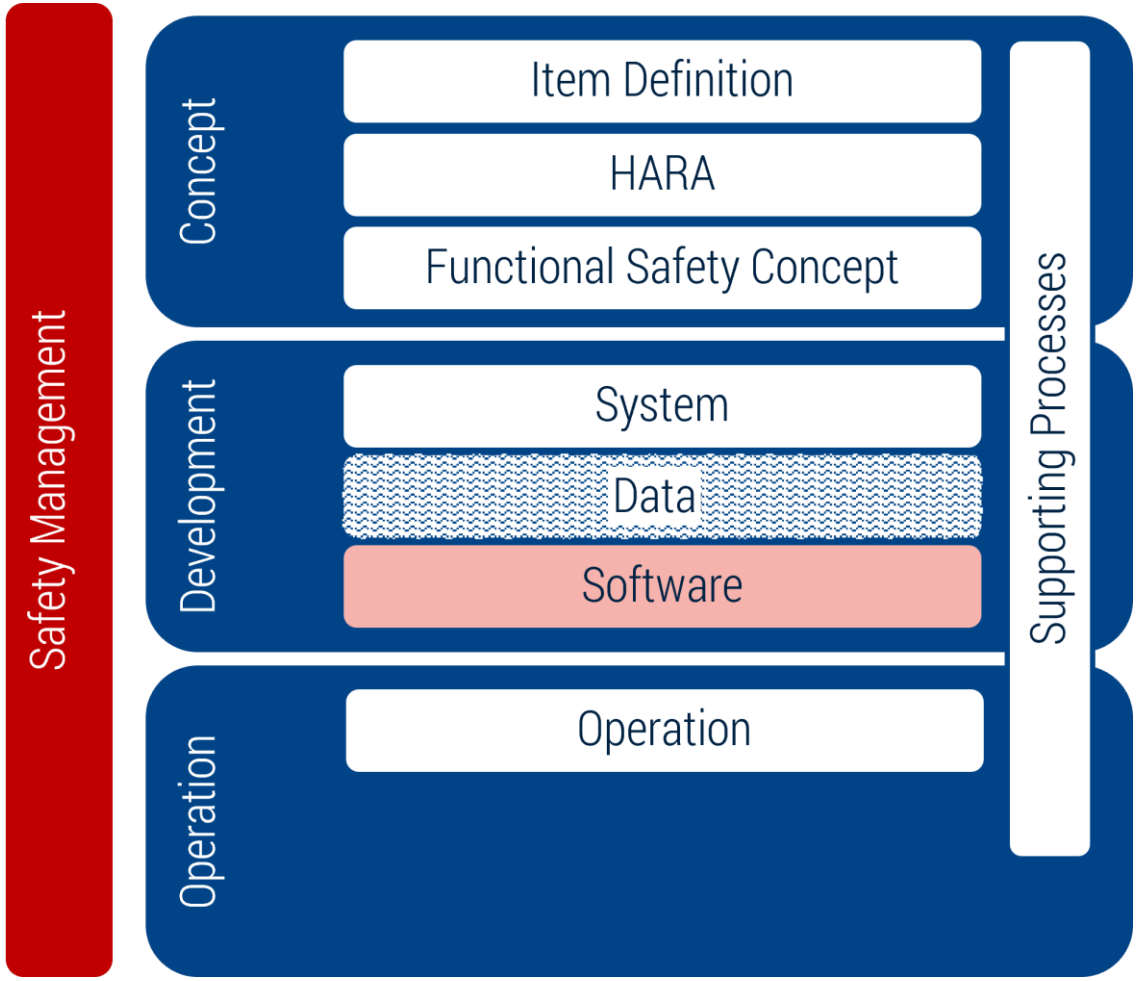
- 6.6.5.1 Software safety requirements specification (ISO 26262)
- 6.6.5.3 Software verification report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 17-ML02 Verifikations- und Validierungsanforderungen ML-Modell und Daten (Referenzprozess ML)
- 19-ML31 Modellierungs- und Trainingsstrategie (Referenzprozess ML)
- 17-ML33 Spezifikation des Modellbenchmarks und der Trainingsbaseline (Referenz-prozess ML)
- 03-ML34 Spezifikation der Modellvorlagen und Hyperparametersätze (Referenzprozess ML)
- 15-ML33 Bericht über die Modellauswahl (Referenzprozess ML)
- 08-ML30 ML-Modell Verifikations- und Validierungsplan (Referenzprozess ML)
- 08-ML50 Spezifikation der ML-Modell Verifikation und Validierung (Referenzprozess ML)
- 13-ML50 Verifikations- und Validierungsergebnisse (Referenzprozess ML)
- 15-ML50 Bericht der Modellvalidierung und -verifikation (Referenzprozess ML)

Software-Architektur-Design

Aufgabe dieser Phase ist es, ein Software-Architekturdesign zu entwickeln, das die Software-Sicherheitsanforderungen und die anderen Software-Anforderungen entsprechend der erforderlichen ASIL erfüllt. Es wird ebenfalls die Implementierung und Verifikation der Software unterstützt.

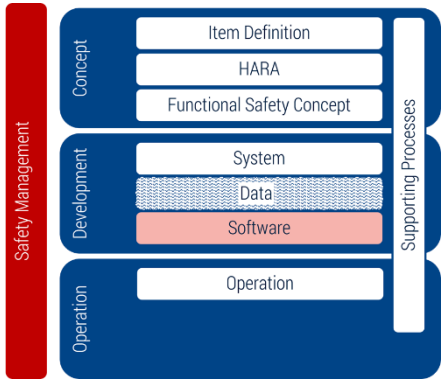


zurück zur Navigation

Software-Architektur-Design

Beherrschung datenintensiver Entwicklungsprozesse (1/3)

AQI | Automotive Quality Institute



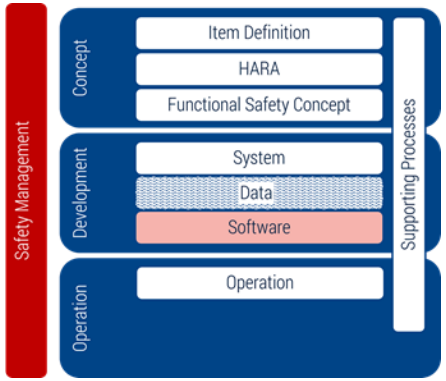
Herausforderung: Klassische Verfahren zur Komplexitätsreduktion (Modularisierung) sind bei der Erstellung von ML-Komponente in der Regel nicht anwendbar. D.h. eine ML-Komponente muss als Einheit spezifiziert, realisiert und verifiziert werden.

Beispiel:: *State of the Art Convolutional Neural Netzworks (CNN) sind komplexe Konstrukte mit mehreren Millionen Parametern und teilweise über hundert Ebenen. Das Fehlen einer sinnvollen Möglichkeit der Modularisierung führt dazu, dass ein solches Netzwerk nur als Ganzes validiert werden kann. Entsprechend intransparent und schwer nachvollziehbar ist die Bedeutung einzelner Parameter und damit möglicher Fehlerquellen im Hinblick auf das jeweilige Validierungsziel*

Risiko: Klassische Sicherheitsnachweise verlangen eine kompositionelle Softwarearchitektur, in der V&V auf verschiedenen Ebenen der Komposition/Dekomposition erfolgen kann. Lässt sich eine Anwendung/System oder Komponente nur schwer oder gar nicht in versteh- und analysierbare Einheiten dekomponieren, sind aufgrund der damit zusammenhängenden höheren Komplexität, Sicherheitsnachweise nur schwer realisierbar. Je größer der Funktionsumfang einer Funktionseinheit ist, desto schwieriger ist es einen nachvollziehbaren Sicherheits-nachweis zu erbringen.

Software-Architektur-Design

Beherrschung datenintensiver Entwicklungsprozesse (2/3)

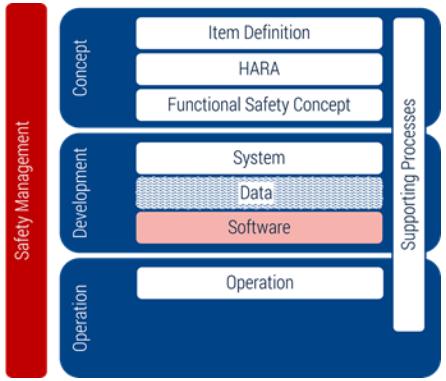


Maßnahmen:

- Kein Ende-zu-Ende-Lernen für automatisiertes Fahren.
- Solange eine klare Abgrenzung der Komponenten Wahrnehmung, Planung und Ausführung weiterbesteht und ML nur punktuell innerhalb einzelner Funktionsbausteine eingesetzt wird, erscheint das Testen und Absichern des Gesamtsystems aufgrund seiner modularen Zerlegbarkeit noch möglich und praktikabel, wenn auch sehr aufwendig (siehe auch Jungmann et al. [51])
- Anwendung von Verfahren zur Schaffung von Transparenz und Erklärbarkeit, um ein Verständnis der Entscheidungsfindung auch bei komplexeren Entscheidungssystemen zu ermöglichen (siehe Abschnitt zu Transparenz und Erklärbarkeit).

Software-Architektur-Design

Beherrschung datenintensiver Entwicklungsprozesse (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es muss nachgewiesen werden, dass die Software-Architektur auch bei Verwendung von ML-Modellen das grundsätzlich notwendige Maß an Dekomposition zulässt, um das System ausreichend verstehen, analysieren und prüfen zu können. Eine entsprechende Dokumentation der Maßnahmen erfolgt im Rahmen der folgenden Arbeitsprodukte:

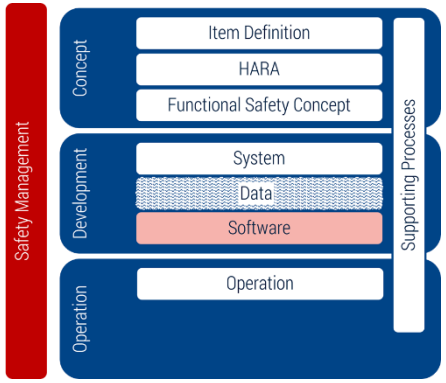
- 6.7.5.1 Software architectural design specification (ISO 26262)
- 6.7.5.2 Safety analysis report (ISO 26262)
- 6.7.5.3 Dependent failures analysis report (ISO 26262)
- 6.7.5.4 Software verification report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 17-ML02 Verifikations- und Validierungsanforderungen ML-Modell und Daten (Referenzprozess ML)
- 19-ML31 Modellierungs- und Trainingsstrategie (Referenzprozess ML)
- 17-ML33 Spezifikation des Modellbenchmarks und der Trainingsbaseline (Referenzprozess ML)
- 03-ML34 Spezifikation der Modellvorlagen und Hyperparametersätze (Referenzprozess ML)
- 15-ML33 Bericht über die Modellauswahl (Referenzprozess ML)
- 15-ML51 Bericht zur Nachvollziehbarkeit der Entscheidungsfindung des ML-Modells (Referenzprozess ML)

Software-Architektur-Design

Absicherung werkzeugintensive Prozesse und komplexe Softwareframeworks (1/3)



Herausforderung: Für die sichere Umsetzung von ML im Kontext sicherheitskritischer Software werden neuartige Sicherheitsarchitekturen und -konzepte notwendig werden, die ergänzend aber zugeschnitten auf die Aufgaben eines algorithmischen Entscheiders, diesen überwachen, korrigieren und notfalls das Gesamtsystem in einen sicheren Zustand überführen (fail safe/fail operational).

***Beispiel**: In einem autonomen Fahrzeug kann ein sog. Sicherheitskäfig dazu verwendet werden, die Grenzen der Fahrzeugbeschleunigung auf der Grundlage des relativen Abstands zum vorrausschauenden Fahrzeug sowie der aktuellen Ego-Geschwindigkeit zu begrenzen. Die KI/ML-basierte Software würde dann nur dazu verwendet werden in dem erlaubten Beschleunigungsbereich zu agieren und diesen abgestimmt auf die Situation zu optimieren. Sicherheitskäfige werden speziell in Bereichen eingesetzt, in denen eine vollständige Offline-Sicherheitsvalidierung nicht möglich ist und können technologiebedingte Risiken minimieren, ohne vollständig auf die Technologie verzichten zu müssen.*

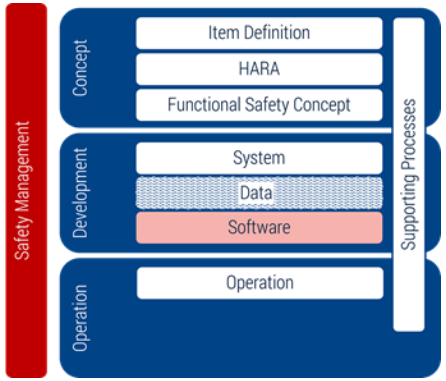
Risiko: Es ist davon auszugehen, dass algorithmische Entscheidungssysteme nicht in ausreichender Robustheit realisiert werden können, sodass sie ohne zusätzliche aktive Überwachungs- und Absicherungssysteme ausreichend sicher betrieben werden können. Solche Systeme sind bisher nur unzureichend in der Automobilindustrie etabliert und standardisiert. Im Rahmen der Architekturspezifikation und der Systemrealisierung müssen solche Systeme identifiziert, spezifiziert und in Abstimmung mit dem ODD und des algorithmischen Entscheiders realisiert werden.

Software-Architektur-Design

Absicherung werkzeugintensive Prozesse und komplexe Softwareframeworks (2/3)

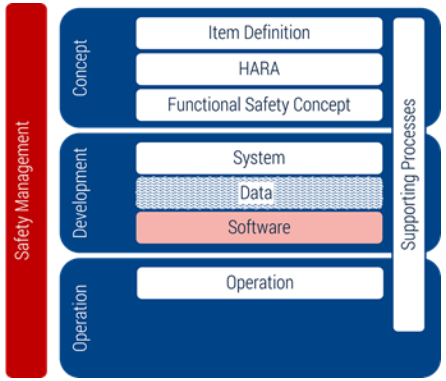
Maßnahmen: Eine Methode zur Laufzeitabsicherung, Laufzeitverifikation und – Qualitätssicherung besteht in der Realisierung eines „Sicherheitskäfig"-Architekturmodells, mit dem sich Eingangsdaten zur Laufzeit dahingehend analysieren lassen, sodass sichere und unsichere Szenarien identifiziert werden und unterschiedlich behandelt werden können. So können unsichere Szenarien beispielsweise nur durch klassische Softwarelösungen umgesetzt oder durch solche gesondert abgesichert werden.

Darüber hinaus ist zu bestimmen, für welche Anwendungen redundante Lösungen notwendig sind und welche Form der Redundanz sinnvoll eingesetzt werden kann.



Software-Architektur-Design

Absicherung werkzeugintensive Prozesse und komplexe Softwareframeworks (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es muss nachgewiesen werden, dass die Software-Architektur ausreichend Sicherheitsmaßnahmen vorsieht, um Schwachstellen eines algorithmischen Entscheidungssystems sinnvoll abzusichern. Eine entsprechende Dokumentation der Architekturelement und ihrer Wirksamkeit erfolgt im Rahmen der folgenden Arbeitsprodukte:

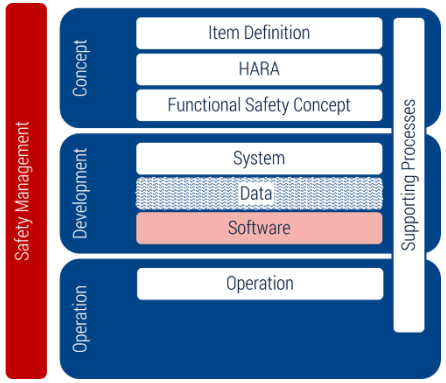
- 6.7.5.1 Software architectural design specification (ISO 26262)
- 6.7.5.2 Safety analysis report (ISO 26262)
- 6.7.5.3 Dependent failures analysis report (ISO 26262)
- 6.7.5.4 Software verification report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML52 Testplan für die Modellintegration (Referenzprozess ML)
- 08-ML50 Testspezifikation (Referenzprozess ML)
- 13-ML50 Testergebnisse und Messungen (Referenzprozess ML)
- 15-ML52 Test- und Verifikationsbericht in der Integration (Referenzprozess ML)

Software-Architektur-Design

Absicherung werkzeugintensive Prozesse und komplexe Softwareframeworks (1/2)



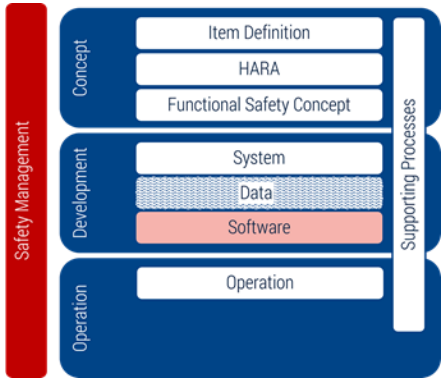
Herausforderung: Abhängigkeit der Systemsicherheit von der gesamten Prozesskette inklusive Datenvorverarbeitung, dem algorithmischen Entscheider und der Ergebnisinterpretation.

***Beispiel**:* Da jedes datengesteuerte Modell in hohem Maße von den bereitgestellten Trainingsdaten abhängt, können böswillige Nutzer versuchen, die Datenqualität zu verändern, um die Systeme potenziell verwundbar zu machen (sog. Data Poisening). Eine der Möglichkeiten besteht darin, den Trainingsdatensatz so zu verändern, dass sich die Vorhersagequalität des Modells verringert. Eine andere Möglichkeit besteht darin, dass in den Trainingsdatensatz Beispiele eingespeist werden, die es den Angreifern ermöglichen, das trainierte und eingesetzte Modell später mit einem bestimmten Verhalten zu steuern. Solche Mechanismen werden als Hintertüren bezeichnet.

Risiko: Algorithmische Entscheidungssysteme sind in eine softwaretechnische Prozesskette eingebunden, die neben dem ML-Modell aus Softwarekomponenten besteht, die eingehende Daten vorverarbeiten und die Entscheidungen des ML-Systems interpretieren. Relevante Aktivitäten im Safety Prozess (Definition technischer Sicherheitsanforderungen, Architektur, Verifikation) müssen die komplexen Abhängigkeiten zwischen ML-Modell und den Softwarekomponenten in der Vorverarbeitung und Ergebnisinterpretation systematisch berücksichtigen.

Software-Architektur-Design

Absicherung werkzeugintensive Prozesse und komplexe Softwareframeworks (2/2)



Maßnahmen: Vermeidung von direkten Abhängigkeiten zwischen Ergebnisinterpretation und ML-Modell durch starke architektonische und logische Trennung der Komponenten (z.B. keine Modell-spezifischen Fehlerkorrekturen in der Ergebnisinterpretation).

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es muss nachgewiesen werden, dass die einzelnen Elemente der Software-Architektur funktional so gekapselt sind, dass das algorithmischen Entscheidungssystems keine unbeabsichtigten Abhängigkeiten zu anderen Komponenten in der Prozesskette besitzt. Eine entsprechende Dokumentation der Architekturelemente sowie die Analyse potenzieller Fehlerketten erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 6.7.5.1 Software architectural design specification (ISO 26262)
- 6.7.5.2 Safety analysis report (ISO 26262)
- 6.7.5.3 Dependent failures analysis report (ISO 26262)
- 6.7.5.4 Software verification report (ISO 26262)

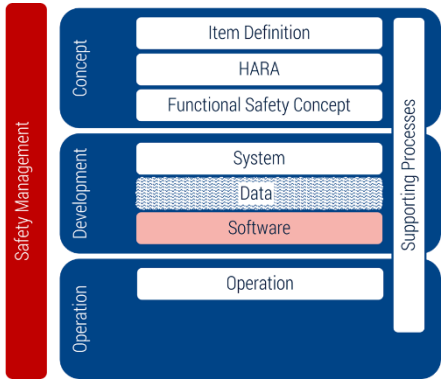
Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML52 Testplan für die Modellintegration (Referenzprozess ML)
- 08-ML50 Testspezifikation (Referenzprozess ML)
- 13-ML50 Testergebnisse und Messungen (Referenzprozess ML)
- 15-ML52 Test- und Verifikationsbericht in der Integration (Referenzprozess ML)

Software-Architektur-Design

Beherrschung datenintensiver Entwicklungsprozesse (1/2)

AQI Automotive Quality Institute



Herausforderung: Abhängigkeit der ML-Sicherheit von komplexen ML-Frameworks wie Tensorflow, pyTorch etc.

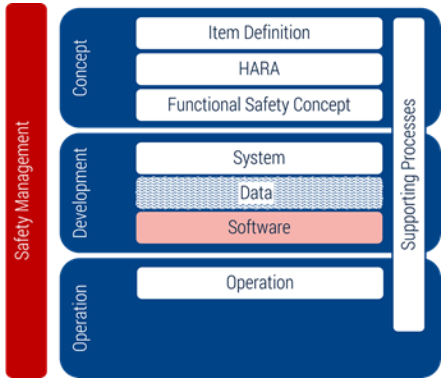
***Beispiel**: TensorFlow ist ein Framework zur Erstellung und zum Ausführen von ML-Modellen. Das Framework wurde von Google initiiert, als Open Source veröffentlicht und von 3.581 Mitwirkenden erarbeitet. Es gilt als State of the Art derzeit verfügbarer ML- Frameworks. Die Codebasis umfasst 2.959.635 Zeilen C++ die über 117.673 Commits in den letzten 5 Jahren in das Framework eingebracht wurden. Der geschätzte Arbeitsumfang beträgt 871 Personenjahre.*

Es kann davon ausgegangen werden, dass industrielle Software nach ihrem Deployment noch zwischen 15 – 50 Fehler per 1000 Zeilen Quellcode aufweist. Wird dieses Maß auf TensorFlow angewandt, kann befürchtet werden, dass das gesamte Framework noch mehr als 450.000 Softwarefehler enthält. Diese Fehler sind mit Sicherheit unterschiedlich relevant aber die hohe Anzahl spricht dafür, dass beim Einsatz im Kontext sicherheitskritischer Anwendungen zusätzliche Absicherungsmaßnahmen speziell auch für die benutzten Frameworks definiert werden müssen.

Risiko: Für die Umsetzung des ML werden i.d.R. komplexe Software-Frameworks verwendet, die ursprünglich nicht für den Einsatz in sicherheitskritischen Systemen vorgesehen wurden und entsprechend nicht den aktuellen regulatorischen Vorschriften und Best Practices der Automobilindustrie entsprechen.

Software-Architektur-Design

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



Maßnahmen: Vermeidung von direkten Abhängigkeiten zwischen Ergebnisinterpretation und ML-Modell durch starke architektonische und logische Trennung der Komponenten (z.B. keine Modell-spezifischen Fehlerkorrekturen in der Ergebnisinterpretation).

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass Fremdsoftware, insbesondere für das ML verwendete Laufzeitumgebungen und Frameworks, frei von sicherheitskritischen Fehlern sind. Eine entsprechende Dokumentation der Sicherheitsnachweise erfolgt im Rahmen der folgenden Arbeitsprodukte:

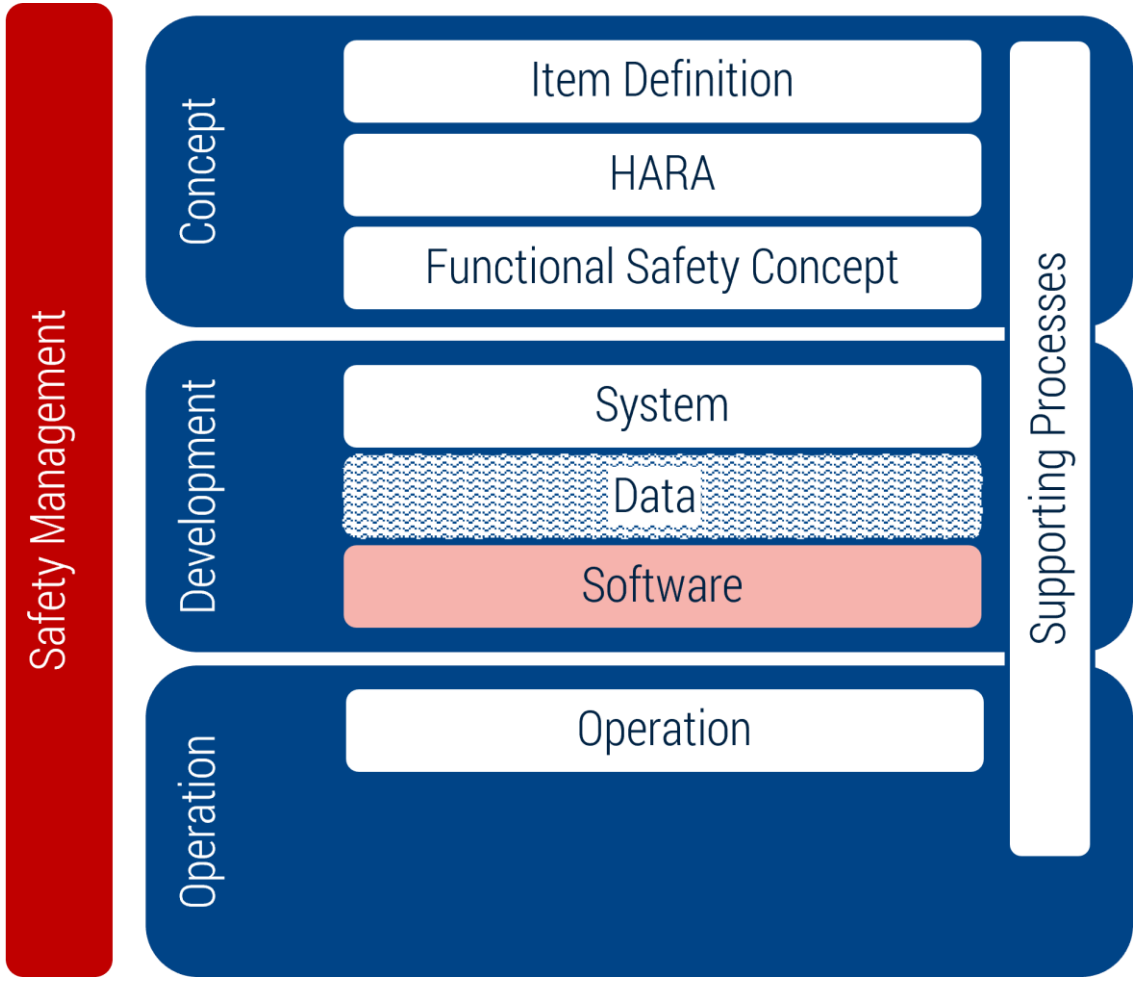
- 6.7.5.1 Software architectural design specification (ISO 26262)
- 6.7.5.2 Safety analysis report (ISO 26262)
- 6.7.5.3 Dependent failures analysis report (ISO 26262)
- 6.7.5.4 Software verification report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 04-ML31 Spezifikation des Rahmenwerks für Modellierung und Training (Referenz-prozess ML)

Entwurf und Implementierung von Software Unit

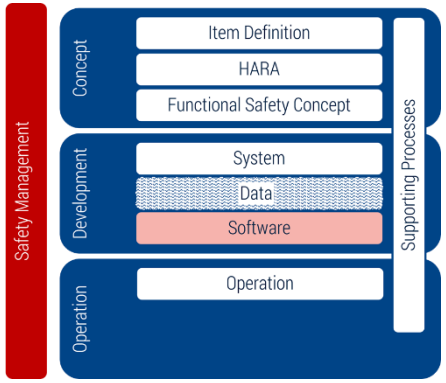
Ziel dieser Phase ist die Entwicklung eines Software Unit Design in Übereinstimmung mit dem Softwarearchitekturdesign, den Designkriterien und den zugewiesenen Softwareanforderungen, die die Implementierung und Verifizierung der Software Unit unterstützen sowie die Implementierung dieser Software Unit.



zurück zur Navigation

Entwurf und Implementierung von Software Unit

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



Herausforderung: Einfluss von Hyperparametern, Algorithmen und Netzwerkarchitekturen auf die Sicherheit des Systems berücksichtigen. Die Optimierung von Hyperparametern ist beim maschinellen Lernen allgegenwärtig und je komplexer die Netzwerke werden, umso anspruchsvoll wir das Ermitteln der optimalen Hyperparametern. Zum einen stellt bereits die große Anzahl von Hyperparametern eine Herausforderung dar. Darüber hinaus beeinflussen sich die Hyperparametern und ihr Einfluss auf Fehler gegenseitig, sodass es schwierig ist, allgemeine Empfehlungen und dauerhaft Hyperparameterwerte für die einzelnen ML-Probleme zu identifizieren.

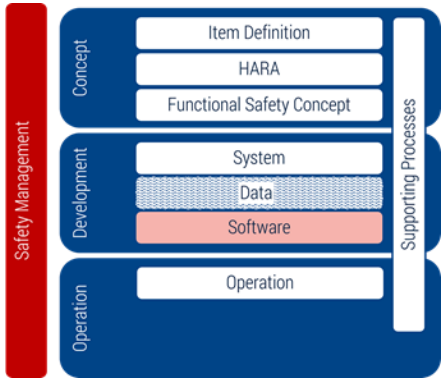
***Beispiel**:: Einer der wichtigsten Hyperparameter für das Training eines neuronalen Netzes mit Backpropagation und Gradientenabstieg, ist die Lernrate. Die Lernrate beschreibt die Größe der Gewichtsaktualisierungen über die die Verlustfunktion des Netzes minimiert werden soll. Ist die Lernrate zu klein gewählt, ist das Training sehr langsam und rechenintensiv, ist die Lernrate jedoch zu hoch, wird das Minimum im Gradientabstieg nicht gefunden. Die optimale Lernrate hängt selber wiederum von anderen Parametern ab, wie u.a. von der Topologie der Verlustfunktion, die wiederum selber von der Modellarchitektur wie auch von den Daten abhängig ist.*

Risiko: Die Funktionalität eines Modells und damit Sicherheit einer ML-basierten Anwendung hängt u.a. von der Auswahl geeigneter Hyperparametern, Algorithmen und Netzwerkarchitekturen ab. Die entsprechenden Auswahlverfahren, -prozesse und -kriterien sind nicht mit den Prozessen und Verfahren des klassischen Softwareengineering vergleichbar.

Entwurf und Implementierung von Software Unit

Beherrschung datenintensiver Entwicklungsprozesse (2/2)

AQI | Automotive Quality Institute



Maßnahmen:

- Definition von Referenzprozesse und -verfahren zur Begleitung des Trainingsprozesses.
- Rückgriff auf vortrainierte Modelle, Datensätze oder Parameter

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, in welcher Form die Auswahl von Hyperparametern, Algorithmen und Netzwerkarchitekturen einen Einfluss auf die Systemsicherheit haben und welche Kriterien bei der Auswahl der Parameter, Algorithmen und Architekturen berücksichtigt wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 6.8.5.1 Software unit design specification (ISO 26262)

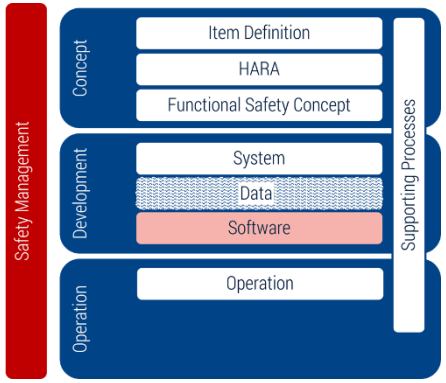
Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 03-ML34 Spezifikation der Modellvorlagen und Hyperparametersätze (Referenzprozess ML)

Entwurf und Implementierung von Software Unit

Beherrschung datenintensiver Entwicklungsprozesse (1/2)

AQI | Automotive Quality Institute



Herausforderung: Vermeidung klassischer Software- und Konversionsfehler bei der softwaretechnischen Realisierung der Neuronalen Netze. Neuronale Netze werden mittels Standardsoftware erstellt und unterliegen denselben Problemen bzgl. Implementierungsfehlern und Datenumwandlungsfehler (beispielsweise von der Fließkommaarithmetik auf Festkommaarithmetik), die bereits von klassischer Software bekannt sind. Bisher gibt es jedoch wenig Erfahrungen dazu, welche Fehler häufig auftreten und wie in neuronalen Netzen systematisch gegen diese Fehler getestet werden kann.

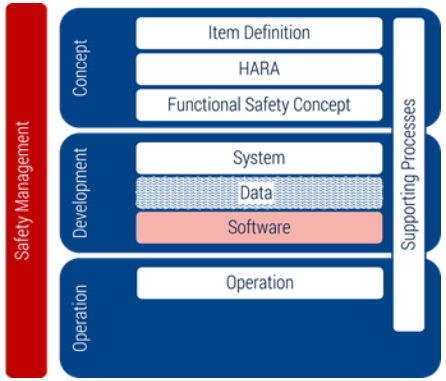
Beispiel:: Neuronale Netze nutzen gängige Computerarithmetikfunktionen mit allen in diesem Kontext bekannten Problemen. Eine Division durch Null wird von diesen Funktionen beispielsweise häufig nicht durch einen Programmabbruch verhindert, sondern mit dem Ergebnis NaN gekennzeichnet. Wird der Wert NaN als Ergebnis im Rahmen des Trainingsprozesse erzeugt, kann er im Model verbleiben und in der Inferenzphase zu Fehlfunktionen führen. Durch gezieltes Testen mit geeigneten Werkzeugen können solche Fehler gefunden und beseitigt werden.

Risiko: Neuronale Netze werde als Software, häufig auf Basis komplexer Frameworks wie Tesorflow, pyTorch etc., entwickelt. Obwohl die Qualität und Fehlerfreiheit des gesamten Softwarestacks (Implementierung des neuronalen Netzes sowie ML-Frameworks) allein nicht hinreichend für eine Sicherheitsargumentation ist, ist die Absicherung der Implementierung jedoch notwendiges Kriterium für die Sicherheit des Systems. Aktuell gibt es jedoch keine Richtlinien und Best Practices, die speziell auf die Implementierungs- und Absicherungsaktivitäten im Rahmen des ML zugeschnitten sind. Bestehende Richtlinien und Best Practices sind nur teilweise übertragbar, weil diese nicht auf die Spezifika des ML (iterative Prozesse, Einbindung in den Trainingsprozess, Höhere Programmiersprachen) abgestimmt sind.

Entwurf und Implementierung von Software Unit

Beherrschung datenintensiver Entwicklungsprozesse (2/2)

AQI Automotive Quality Institute



Maßnahmen:

Definition von Prozessen, Methoden und Best Practices zur Qualitätssicherung der software-technischen Qualitätsaspekte bei der Entwicklung von ML-Modellen (Softwarefunktionalität, die nicht durch den Trainingsprozess determiniert wird.)

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass die softwaretechnische Realisierung der verwendeten Neuronalen Netze frei von systematischen Fehlern ist. Nachweise können durch Proof in Use (ML-Frameworks, die einen hohen Grad an Wiederverwendung besitzen) oder durch einen systematischen V&V Ansatz erfolgen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 6.8.5.1 Software unit design specification (ISO 26262)
- 6.8.5.2 Software unit implementation (ISO 26262)

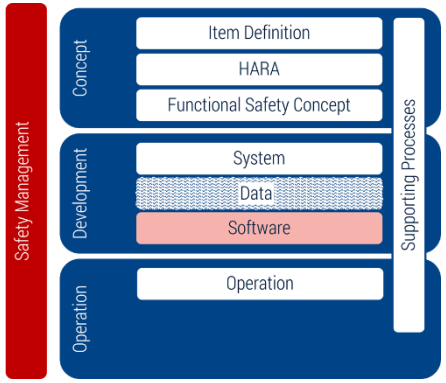
Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 04-ML31 Spezifikation des Rahmenwerks für Modellierung und Training (Referenz-prozess ML)
- 13-ML51 Testergebnisse des softwaretechnischen Tests der Modellierungs- und Trainingsinfrastruktur und des Modells (Referenzprozess ML)

Entwurf und Implementierung von Software Unit

Beherrschung datenintensiver Entwicklungsprozesse (1/2)

AQI | Automotive Quality Institute



Herausforderung: ML ist ein Optimierungsprozess, in dem potenziell widersprüchliche Qualitätsattribute maximiert werden sollen. Der Prozess an sich ist hochiterativ und benötigt klar definierte Vor-gaben für den Optimierungsprozess. Hierzu zählt u.a. die Definition sicherheitsrelevanter KPIs und Optimierungskriterien sowie klarer Vorgaben bezüglich Robustheit und zur Definition der Trainingsbaseline, der Modellauswahl sowie der Modellevaluation.

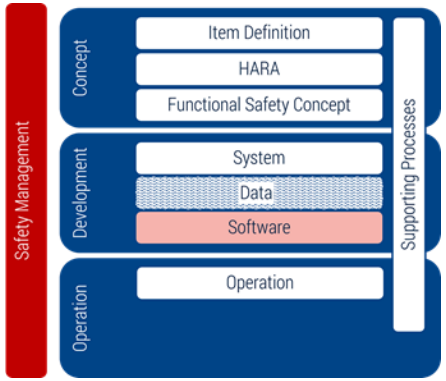
***Beispiel**: Es gibt die Hypothese, dass es ein inhärentes Spannungsverhältnis zwischen der Robustheit eines neuronalen Netzes gegen Adversarial Examples seiner Generalisierungsfähigkeit besteht. Insbesondere kann das Training robuster Modelle nicht nur ressourcenintensiver sein, sondern auch zu einer Verringerung der Genauigkeit führen (siehe Tsipras et al)*

Risiko: Gängige Praxis im ML ist die Optimierung eins Modells in Hinblick auf seine Leistungsfähigkeit. Diese wird i.d.R. als Kompromiss zwischen der Genauigkeit und der Generalisierungsfähigkeit des Modells definiert und durch die Variation über qualitätsbestimmende Faktoren (Daten, Selektionen, Sampling, Hyperparameter, Algorithmen, Modellarchitekturen) im Zuge der Trainingsaktivtäte optimiert. Für die Sicherheit eines Systems sind neben der Genauigkeit und der Generalisierungsfähigkeit zusätzliche Eigenschaften relevant, die häufig unberücksichtigt bleiben. Hierzu zählen u.a. die Robustheit und Stabilität des Modells sowie die Möglichkeit, Entscheidungen nachvollziehen zu können.

Darüber hinaus ist sicherzustellen, dass die Leistungsfähigkeit des Modells nicht nur für Standardfälle ermittelt wird, sondern auch für Rand- und Sonderfälle der ODD sichergestellt ist

Entwurf und Implementierung von Software Unit

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



Maßnahmen:

- Definition notwendiger Qualitätsmerkmale und -eigenschaften von Modellen für verschiedene Rand- und Sonderfälle der ODD.
- Definition geeigneter Metriken und KPIs zur Modellbewertung und zum Vergleich von Modellen.
- Definition eines Prozessmodells für den Trainingsprozess, sodass sichergestellt werden kann, dass alle notwendigen Aktivitäten durchgeführt wurden.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass durch die Auswahl des ML-Modells als Ergebnis des Optimierungsprozesses, die ideale Kombination von Sicherheitseigenschaften erreicht wurde. Hierzu sind die sicherheitsrelevanten Anforderungen und KPIs mit den erzielten Anforderungen und KPIs in Beziehung zu setzen und zu bewerten. Die Entscheidungskriterien für die Auswahl eines Modells als das optimale Modell ist explizit zu dokumentieren. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

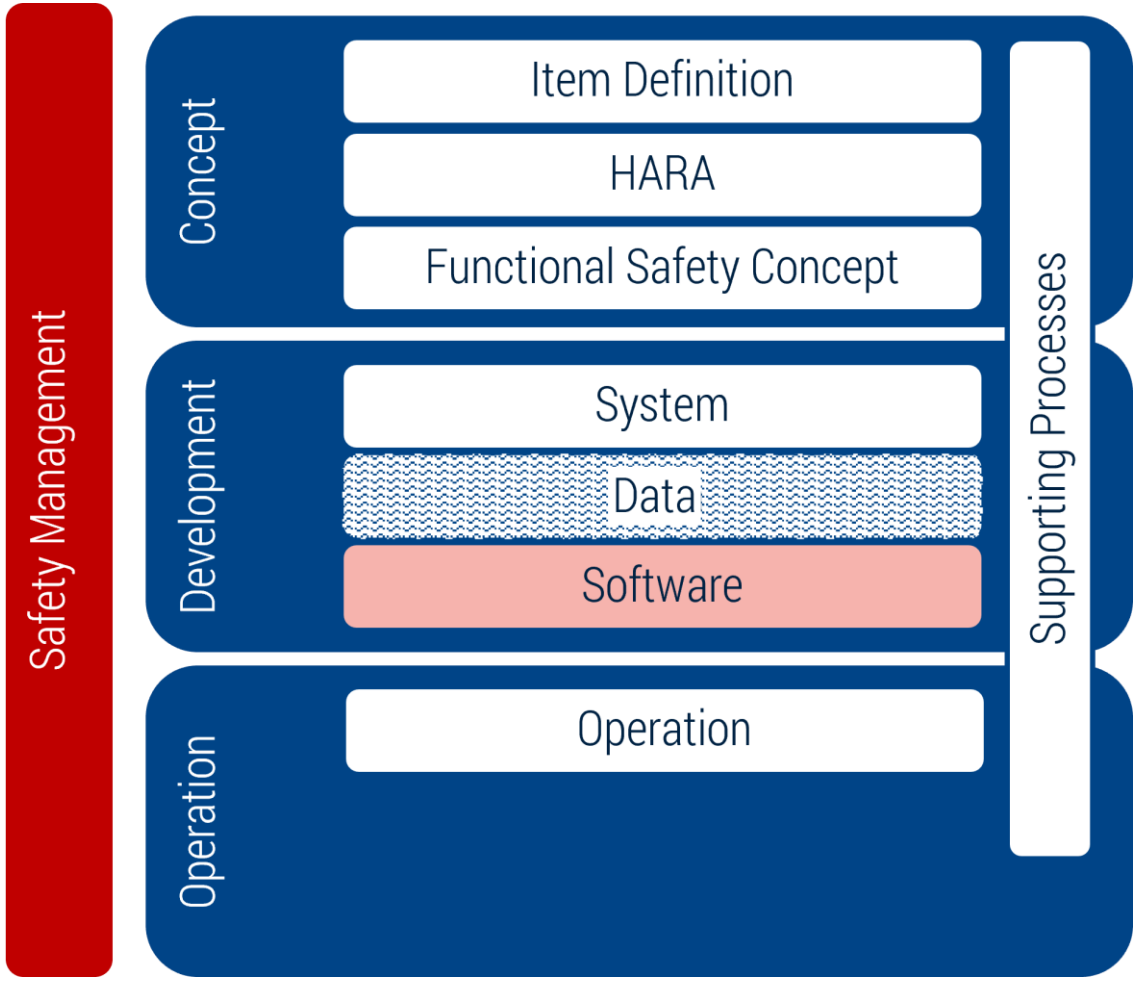
- 6.8.5.2 Software unit implementation (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 17-ML11 Anforderungsspezifikation ML-Modell und Daten (Referenzprozess ML)
- 17-ML33 Spezifikation des Modellbenchmarks und der Training-Baseline (Referenzprozess ML)
- 15-ML33 Bericht über die Modellauswahl (Referenzprozess ML)

Software Unit Verifizierung

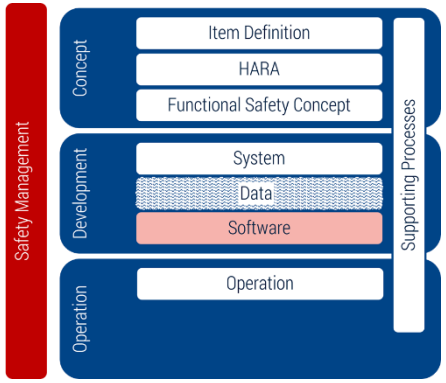
Ziel dieser Phase ist es, den Nachweis zu erbringen, dass der Entwurf der Software Unit die zugewiesenen Softwareanforderungen erfüllt und für die Implementierung geeignet ist. Es soll belegt werden, dass die definierten Sicherheitsmaßnahmen, die sich aus den sicherheitsbezogenen Analysen nach 7.4.10 und 7.4.11 ergeben, ordnungsgemäß umgesetzt werden und dass die implementierte Software Unit mit dem Unit Design übereinstimmt und die zugewiesenen Softwareanforderungen mit dem erforderlichen ASIL erfüllt sind. Zudem soll sichergestellt werden, dass die Software Unit weder unerwünschte Funktionalitäten noch unzulässige Eigenschaften hinsichtlich der funktionalen Sicherheit aufweist.



zurück zur Navigation

Software Unit Verifizierung

Bereitstellung geeigneter V&V Verfahren und Strategien (1/3)



Herausforderung: Model-Checking- und Theorem-Proving-Techniken sind im Allgemeinen bei der Verifikation unvollständiger Spezifikationen nur eingeschränkt nutzbar. Darüber hinaus führt die Komplexität eines Neuronalen Netzes (Operationen in hochdimensionalen Vektorräumen, nicht-lineare Abbildungen) zu einer Explosion der zu verifizierenden Zustände bzw. nicht-konvexen Lösungsräumen. So ist in den meisten Fällen eine formale Verifikation eines Neuronalen Netzes nicht oder nur schwer realisierbar.

Erläuterung: Ein neuronales Netz besteht aus einem (i.d.R. gerichteten) Grafen, in dem der Wert eines jeden Knotens durch die Berechnung einer Linearkombination von Werten aus den Vorgängerknoten sowie der anschließenden Anwendung einer Aktivierungsfunktion auf das Ergebnis der Linearkombination berechnet wird. Die Aktivierungsfunktionen sind i.d.R. nichtlinear, um bessere Approximationseigenschaften umsetzen zu können.

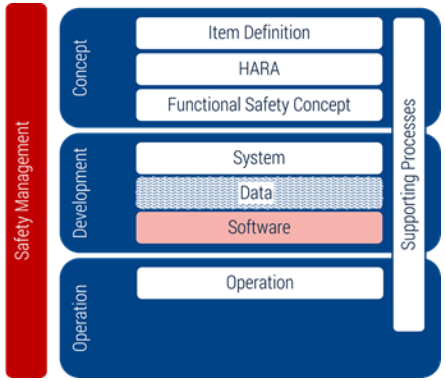
***Beispiel**:: Die Robustheit eines neuronalen Netzes wird dadurch belegt, dass für ein konkretes Eingabedatum X gezeigt wird, dass alle Eingabedaten, die in der näheren Umgebung von X liegen dasselbe Ergebnis liefern wie X. Das lässt sich auch für einfache Anwendungen und für einzelne Punkte des Eingaberaums beweisen. Für komplexere Anwendungen, wie z.B. tiefe neuronale Netze, die im Rahmen der Objekterkennung für das autonome Fahren relevant sind, ist das leider nicht mehr möglich.*

Wir haben keine Möglichkeit, alle Punkte X erschöpfend aufzuzählen in deren Umgebung der das Ergebnis einer Objekterkennung annähernd konstant sein sollte, da der Eingaberaum zu groß ist und wir nicht in der Lage sind alle zukünftigen natürlich vorkommenden Eingaben zu identifizieren. Ein weiteres Problem besteht darin, dass es verschiedene Normen und Metriken gibt, die in den hochkomplexen Eingaberäumen eine „nähere Umgebung" definieren. Alle verwendeten Metriken und Normen stellen eine Idealisierung des Problems der Umgebungsdefinition dar und haben einen starken Einfluss auf das Verifikationsergebnis.

Risiko: ISO 26262 empfiehlt dringend, für ASIL C- und D-klassifizierte Systeme unter anderem semiformale und formale Verifikation zu verwenden, um Design und Implementierung von Softwarekomponenten zu verifizieren.

Software Unit Verifizierung

Bereitstellung geeigneter V&V Verfahren und Strategien (2/3)

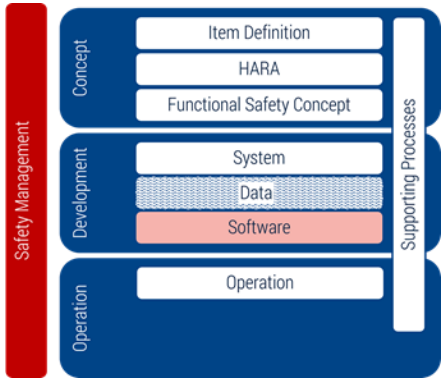


Maßnahmen:

- Entwicklung und Auswahl geeigneter formaler Verifikationsverfahren, um formale Beweise für Teilaspekte der in Neuronalen Netzen realisierten Funktionalität bereit-stellen zu können.
- In Huang et. al. werden existierende Ansätze zur Verifikation von neuronalen Netzen zusammenfassend beschrieben.
- [Huang et al., 2017b] entwickeln einen automatisierten Verifikationsansatz für neuronale Netze auf Basis der Satisfiability Modulo Theory (SMT).
- [Weng et al., 2018] definieren mit CLEVER eine Metrik, um die Lipschitz-Konstante schätzen zu können und damit eine sichere untere Schranke für die Robustheit einer Entscheidung definieren zu können, indem über die Gradienten eine Grenzwertverteilung mithilfe der Extremwerttheorie ermittelt werden wird.
- [Katz et al., 2017] und [Ehlers, 2017] entwickeln mit sog. Reluplex-Solvern ein SMT Verfahren, um neuronale Netze auf Eigenschaften zu verifizieren, die mit SMT-Constraints ausdrückbar sind.

Software Unit Verifizierung

Bereitstellung geeigneter V&V Verfahren und Strategien (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: Da derzeit noch keine Standards existieren, die als Referenz für den Stand der Technik beim Einsatz semiformale und formale Verifikationstechniken zur Absicherung sicherheitskritischer Systeme herangezogen werden können, ist nachzuweisen, dass die verwendeten semi-formale und formale Verifikationstechniken geeignet sind, die Nachweise für Korrektheit, Robustheit und Fehlerfreiheit der geprüften Software zu belegen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

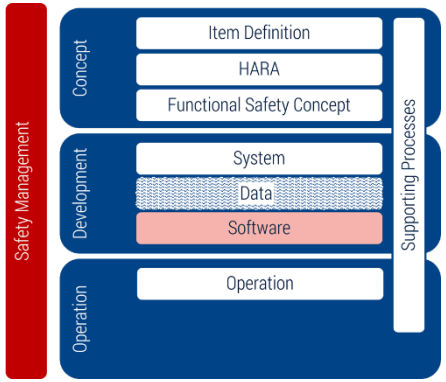
- 6.9.5.1 Software verification specification (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML30 ML-Modell Verifikations- und Validierungsplan (Referenzprozess ML)
- 08-ML50 Spezifikation der ML-Modell Verifikation und Validierung (Referenzprozess ML)

Software Unit Verifizierung

Bereitstellung geeigneter V&V Verfahren und Strategien (1/2)



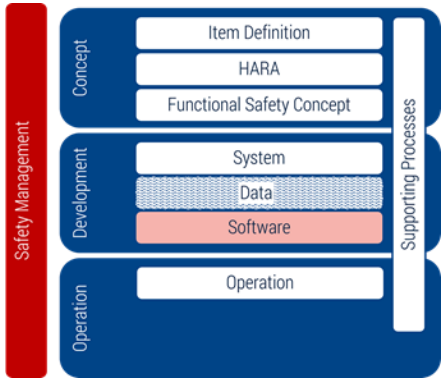
Herausforderung: Das Testen ist ein Verifikationsverfahren, das versucht, Fehler in den Eigenschaften eines Produktes anhand geeigneter Stichproben zu finden. Aufgrund der fehlenden präzisen Systemspezifikation (fehlende Anforderungen aufgrund der Komplexität und Komplexität der Einsatzumgebung) und völlig neuartiger Entscheidungsstrukturen im Vergleich zu klassischen Programmen (neuronales Netz vs. Kontrollflussgraph) sind bereits in der klassischen Softwareentwicklung bewährte und etablierte Testansätze nicht eins zu eins auf ML-basierte Software anwendbar. Es ist derzeit noch unklar, wie das systematische Testen einer ML-Komponente erfolgen und wie der Nachweis der Testvollständigkeit erbracht werden kann.

***Beispiel**:: Bei der Verifikation klassischer Software werden White-box Tests, d.h. Tests, die sich an der Struktur der Software orientieren, durchgeführt, um beispielsweise Logikfehler im Programm zu entdecken. Das macht insbesondere deshalb Sinn, weil die wichtigen Aspekte der Funktionalität über den Code und seine Struktur determiniert sind. Bei der traditionellen Softwareentwicklung liegt die Komplexität also im Code. Beim maschinellen Lernen hingegen sind die grundsätzlich zu erstellenden Algorithmen und deren Umsetzung im Prinzip einfach. Die die Komplexität und die wichtigen Aspekte der gewünschten Funktionalität liegt in den Daten und Parametern. Der White-box Test (im klassischen Sinn) macht zwar vor diesem Hintergrund auch für ML-Modelle und Sinn, beschränkt sich aber allein auf das Finden von Fehlern, die mit der Programmierung und halt nicht auf solche, die mit dem Training assoziiert sind.*

Risiko: Funktionale Tests sind in hohem Maße von der Existenz präziser Anforderungen abhängig. ML-basierte Anwendungen adressieren in der Regel Anwendungsbereiche, die nicht auf die übliche Weise spezifiziert werden können. Klassisches anforderungsbasiertes Testen ist daher schwer zu realisieren. Testen ist ein Verifikationsprozess, der versucht, Fehler in den Eigenschaften eines Produkts mit Hilfe geeigneter Stichproben zu finden. Sowohl für die Auswahl der Stichproben als auch für die Menge der Stichproben werden geeignete Abdeckungsmaße benötigt, die Aussagen über die Vollständigkeit des Testverfahrens erlauben. Für ML-basierte Anwendungen gibt es derzeit keine solche Maße. Während das Testen des ML-Modells (als Teil des Trainingsprozesses) typischerweise Eigenschaften wie Genauigkeit und Generalisierung adressiert, müssen implementierungs- und integrationsbezogene Tests sowie Qualitätseigenschaften wie Robustheit, Sicherheit und Datenschutz in getrennten Prozessen adressiert werden.

Software Unit Verifizierung

Bereitstellung geeigneter V&V Verfahren und Strategien (2/2)



Maßnahmen:

- Forschung, die sich der Definition von sinnvollen strukturellen Abdeckungskriterien für ML widmet (zusätzlich zu den oben skizzierten Ansätzen).
- Definition von Abdeckungskriterien basierend auf dem Verständnis der Eingabedomäne (z.B. Risikoabdeckung, Abdeckung von Sicherheitsszenarien).
- Definition einer Reihe von Referenzprozessen, die die Testaktivitäten im Zusammenhang mit der Modellimplementierung, dem Modelltraining und der Modellintegration umreißen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Da derzeit noch keine Standards existieren, die als Referenz für den Stand der Technik für Testverfahren zur Absicherung sicherheitskritischer ML-Systeme herangezogen werden können, ist nachzuweisen, dass die verwendeten Testverfahren geeignet sind, die Nachweise für Korrektheit, Robustheit und Fehlerfreiheit der geprüften Software zu belegen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

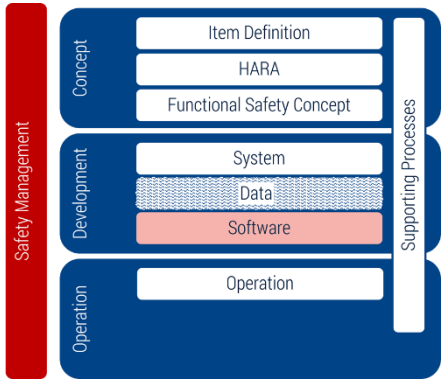
- 6.9.5.1 Software verification specification (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML30 ML-Modell Verifikations- und Validierungsplan (Referenzprozess ML)
- 08-ML50 Spezifikation der ML-Modell Verifikation und Validierung (Referenzprozess ML)

Software Unit Verifizierung

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (1/3)



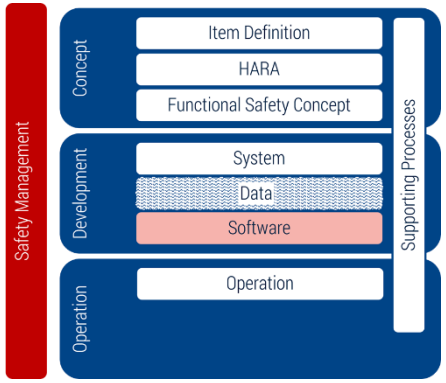
Herausforderung: Trainingsprozess ist i.d.R. nicht-deterministisch. Durch die Verwendung von zufällig gewählten Parametern sowie stochastischen Verfahren im Rahmen der Optimierung (mini-batch, stochastic gradient decent), muss davon ausgegangen werden, dass jeder Trainings-prozess andere Modelle erzeugt.

Nicht-Determinismus im Trainingsprozess (https://determined.ai/blog/reproducibility-in-ml/) hat die folgenden Ursachen:

- Zufällige Initialisierung der Ebenengewichte: Viele ML-Modelle setzen die initialen Gewichtswerte durch zufällige Initialisierungswerte. Es hat sich gezeigt, dass dies die Konvergenzgeschwindigkeit gegenüber der Initialisierung aller Gewichte mit Nullen erhöht [1, 2].
- Zufällige Auswahl von Trainingsdatensätzen: Der Trainingsdatensatz wird bei der Initialisierung oft zufällig gemischt (mini batch sampling).
- Bewusst eingesetztes Rauschen: Bestimmte Modellarchitekturen können Schichten enthalten, die während des Trainings mit inhärenter Zufälligkeit arbeiten. Beispielsweise schließen Dropout-Techniken im Training den Beitrag eines bestimmten Eingabeknotens mit einer vordefinierten Wahrscheinlichkeit aus. Ziel ist es, eine Überanpassung zu verhindern mit der Konsequenz, dass dieselbe Eingabeprobe bei jeder Iteration unterschiedliche Ebenenaktivierungen erzeugen kann.
- Änderungen in ML-Frameworks: Aktualisierungen von ML-Bibliotheken können zu subtil unterschiedlichem Verhalten bei der Nutzung solcher Frameworks führen, während die Migration eines Modells (von einem Framework zu einem anderen) zu einer noch größeren Diskrepanz führen kann.
- Nicht-deterministische Berechnung auf CPU/GPU: Bestimmte Funktionen in Deep Neural Network Bibliotheken für GPUs, garantieren standardmäßig keine Reproduzierbarkeit über verschiedene Trainingsläufe hinweg. Ähnliche Effekte entstehen durch Parallelverarbeitung (CPU-Multithreading) beispielsweise bei der Nutzung von Tesorflow.

Software Unit Verifizierung

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken (2/3)



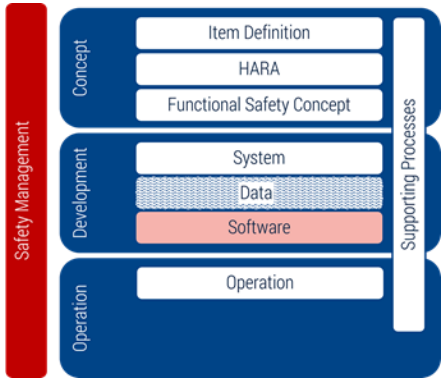
Beispiel:: *Das trainieren eines Modells ist ein komplexer Optimierungsprozess, in dessen Verlauf eine Vielzahl von Hyperparameter angepasst werden müssen. Für jede Anpassung wird jeweils ein neues Kandidatenmodell erzeugt, dass in seiner Leistungsfähigkeit mit dem den zuvor erzeugten Modellen verglichen wird, um so schlussendlich die beste Hyperparameterbelegung zu finden. Die Tatsache, dass der Trainingsprozess nicht-deterministisch ist, macht den Vergleich der Kandidatenmodelle deutlich komplexer, da Unterschiede im Modellveralten nicht nur auf die Parameteränderungen sondern auch auf den Nichtdeterminismus zurückgeführt werden müssen.*

Risiko: Der Nicht-determinismus in den Ergebnissen des Trainingsprozesses führt dazu, dass die Ergebnisse verschiedener Trainingsläufe nicht 1 zu 1 miteinander verglichen werden können, was Konsequenzen für etablierte Qualitätssicherungsverfahren haben kann (z.B. beim Back-to-Back Test). In der Konsequenz führen nicht-deterministische Trainingsprozesse zu Ergebnissen die:

- nur mit statistischen Methoden miteinander vergleichbar sind.
- nur bedingt wiederholbar sind (es gibt keine Garantie, dass ein erneutes Trainieren unter den gleichen technischen Bedingungen zu gleichen Ergebnissen führt).

Software Unit Verifizierung

Bereitstellung geeigneter V&V Verfahren und Strategien (3/3)



Maßnahmen:

- Entwicklung und/oder Bereitstellung von statistischen Ansätzen zur Qualitätsbewertung und Absicherung von Softwarekomponenten.
- Vermeidung von Nicht-determinismus im Trainingsprozess (wenn dieser technisch vertretbar ist und nicht zu anderweitigen Einschränkungen/Qualitätsverlusten führt.)

Anforderungen bzgl. Dokumentations- und Nachweispflicht:

Es ist nachzuweisen, dass der Nicht-determinismus im Trainingsprozess keine Auswirkungen auf die Systemsicherheit hat. Es ist zu dokumentieren, welche Konsequenzen der Nicht-determinismus für die Absicherung hat und, dass die verwendeten Absicherungs- und Prüfverfahren geeignet sind, mit dem vorhandenen Nicht-determinismus umzugehen.

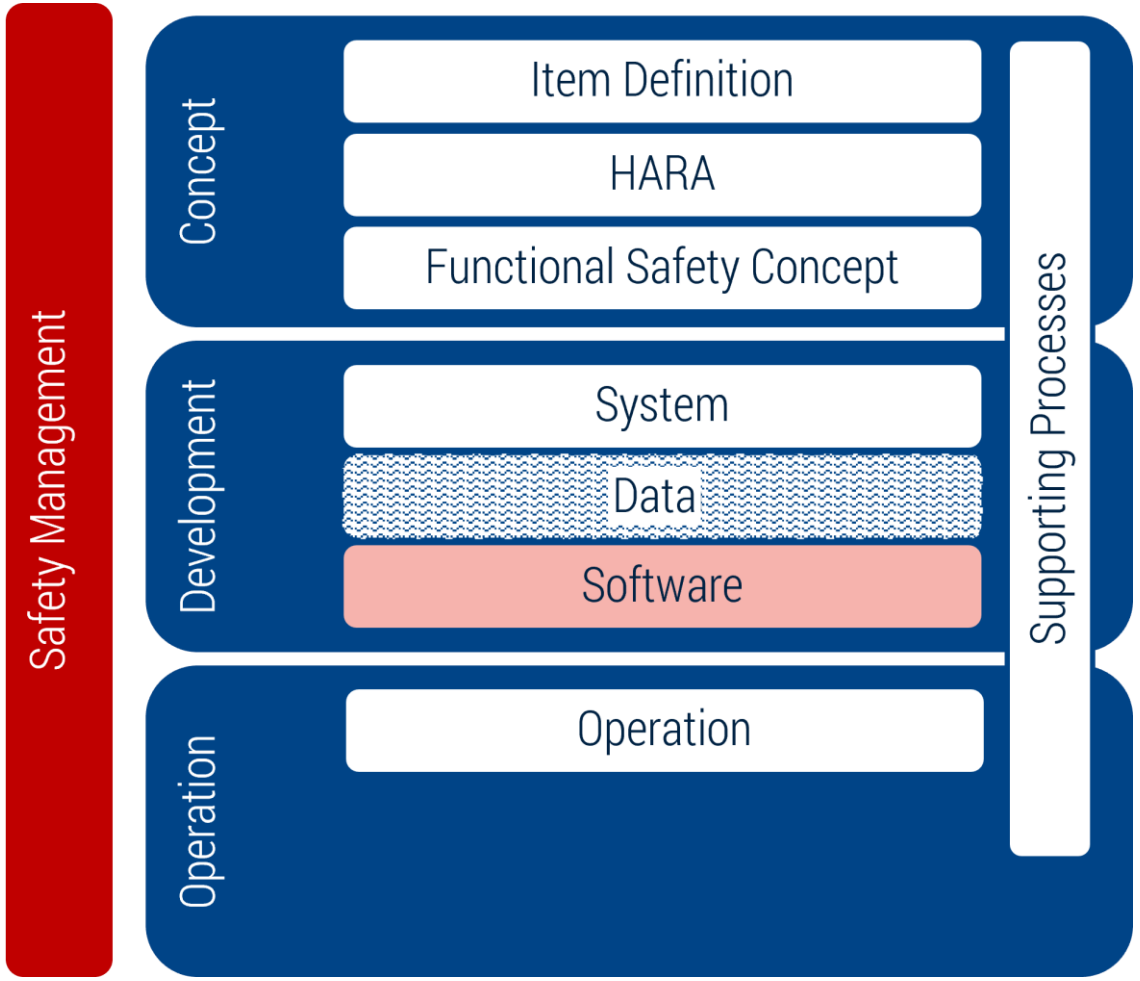
- 6.9.5.1 Software verification specification

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML30 ML-Modell Verifikations- und Validierungsplan (Referenzprozess ML)
- 08-ML50 Spezifikation der ML-Modell Verifikation und Validierung (Referenzprozess ML)

Software-Integration und Verifizierung

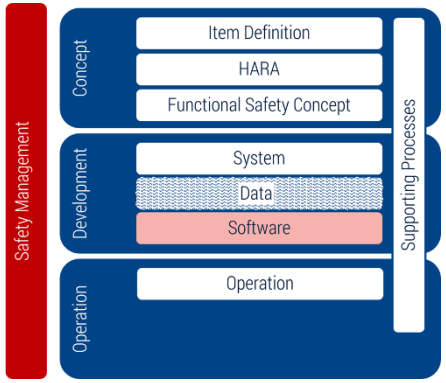
Ziel dieser Phase ist es, die Integrationsschritte zu definieren, die Softwareelemente bis zur vollständigen Software zu integrieren und sicherzustellen, dass die integrierten Software Units und Software-Komponenten ihre Anforderungen gemäß dem Software-Architekturdesign erfüllen und keine unerwünschten Funktionalitäten oder Eigenschaften hinsichtlich der funktionalen Sicherheit auftreten.



zurück zur Navigation

Software-Integration und Verifizierung

Absicherung werkzeugintensiver Prozesse und komplexer Softwareframeworks (1/2)



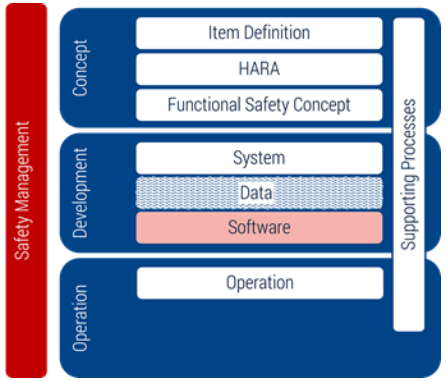
Herausforderung: Sowohl der Trainingsprozess als auch der spätere Betrieb eines Modells setzt eine Vorverarbeitung der prozessierten Daten sowie die Interpretation der Modellergebnisse durch die weiterverarbeitenden Komponenten voraus. Der Integrationsprozess kann jedoch mit erheblichen Problemen einhergehen, insbesondere dann, wenn Regeln für eine strikte Modularisierung und Aufgabenteilung nicht eingehalten wurden und konnten. Der Integrationsprozess hat sicherzustellen, dass durch die Integration der Komponenten keine Fehler entstehen. Insbesondere die Integration in der Verarbeitungsketten der für einen algorithmischen Entscheider notwendigen Datenflüsse und Ergebnisinterpretation sind komplex und weisen häufig implizite Abhängigkeiten auf (z.B. Vorverarbeitungsschritte in der Aufbereitung der Bilddaten auf der Strecke vom Sensor zum Modell, Ergebnisinterpretation und Unsicherheitsabschätzung). Hinzu kommt die Etablierung schlechter Designpattern, insbesondere durch schwer zu pflegendem Verbbindungscode, versteckten Abhängigkeiten zwischen Komponenten, Feedback-Schleifen usw. (siehe Amershi et al [54]).

***Beispiel**:: Zur Umsetzung einer autonomen Fahrfunktion sollen mehrere ML-Modelle miteinander interagieren. Ein vorgelagertes Objekterkennungssystem identifiziert einzelne Objekte und ein nachgelagertes Prädiktionssystem ordnet den Objekten Trajektorien zu und berechnet deren Wahrscheinlichkeiten. In beiden Systemen werden ML-Modelle verwendet. Auch wenn beide Systemen klar definierte Schnittstellen besitzen, können sie nicht einfach als fertige Systeme in der Integration miteinander gekoppelt werden. Es muss davon ausgegangen werden, dass das zweite System auf die Ausgaben des ersten Systems trainiert werden muss, um zuverlässig funktionieren zu können. Das heißt, dass der Integrationsprozess nicht erst die fertigen Teilsysteme betrachten kann, sondern bereits die Trainingsphase umfassen muss. Amershi et al. [54] stellen beispielsweise fest, dass, es nicht möglich ist Sprachmodelle für das Bestellen einer Pizza mit einem Sprachmodell für die englische Sprache einfach miteinander zu koppeln und zu erwarten, dass Pizza in korrektem Englisch bestellt wird.*

Risiko: Durch die expliziten und impliziten Abhängigkeiten zwischen Datenvorverarbeitung, Vor-hersage und Ergebnisinterpretation besteht ein großes Potenzial für Fehlerfälle, die nur in der Integration auftreten. Diese Fehlerfälle müssen durch systematische Tests identifiziert und anschließend beseitigt werden. Geeignete Abdeckungsmaße sind zu identifizieren, mit denen der Nachweis erbracht werden kann, dass die Integration des algorithmischen Entscheiders systematisch getestet wurde.

Software-Integration und Verifizierung

Absicherung werkzeugintensiver Prozesse und komplexer Softwareframeworks (2/2)



Maßnahmen:

- Einführung von Referenzarchitekturen für die Integration von ML-basierten Anwendungen in die Softwarearchitektur von Fahrzeugen.
- Entwicklung von Testverfahren und Abdeckungsmaßen zum systematischen Nach-weis der Güte des algorithmischen Entscheiders in der Integration mit Datenvorverarbeitung und Ergebnisinterpretation.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Da derzeit noch keine Standards existieren, die als Referenz für den Stand der Technik für Testverfahren zur Absicherung der Integration sicherheitskritischer ML-Systeme herangezogen werden können, ist nachzuweisen, dass die verwendeten Testverfahren und Abdeckungsmaße geeignet sind, Integrationsfehler in der geprüften Software zu belegen. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

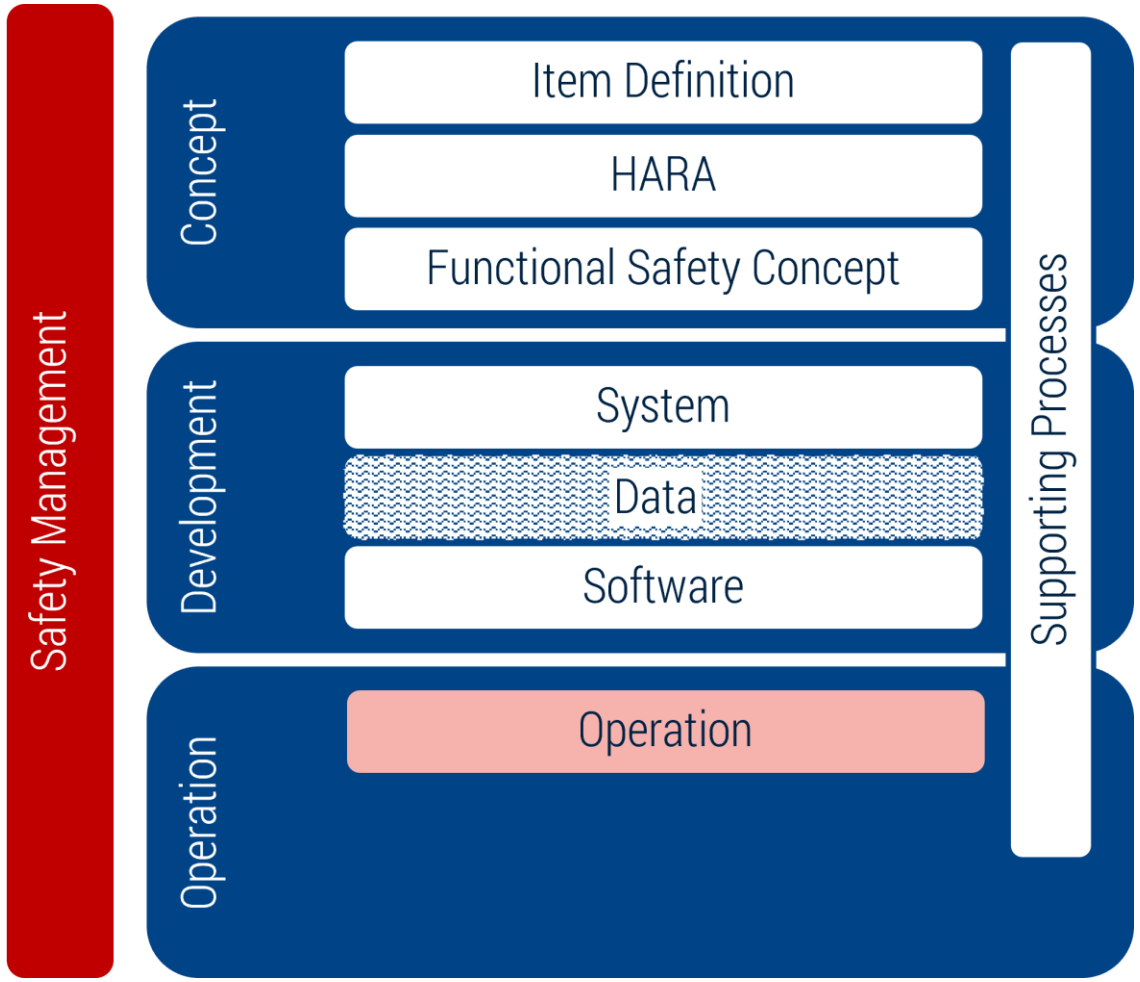
- 6.10.5.1 Software verification specification (ISO 26262)
- 6.10.5.3 Software verification report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML52 Testplan für die Modellintegration (Referenzprozess ML)
- 08-ML50 Testspezifikation (Referenzprozess ML)
- 13-ML50 Testergebnisse und Messungen (Referenzprozess ML)
- 15-ML52 Test- und Verifikationsbericht in der Integration (Referenzprozess ML)

Betrieb, Service und Außerbetriebnahme

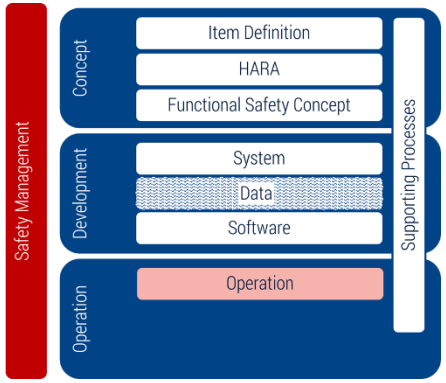
Aufgabe dieser Phase ist das Sicherstellen der funktionalen Sicherheit während der Teilphasen Betrieb, Wartung und Reparatur sowie Außerbetriebnahme des Fahrzeuglebenszyklus.



zurück zur Navigation

Betrieb, Service und Außerbetriebnahme

Absicherung der Betriebsphase (1/3)



Herausforderung: Aufgrund der Komplexität, Kritikalität und potenzieller Fehleranfälligkeit algorithmischer Entscheidungssysteme ist davon auszugehen, dass die dem Entscheider zugrundeliegenden Modelle und Softwarekomponenten im Betrieb beobachtet und ggfs. aktualisiert werden müssen.

***Beispiel**:: KI/ML-basierten Software wird i.d.R. für Probleme eingesetzt, bei denen klassische Software keine ausreichende Lösung erlaubt. Solche Probleme sind häufig Open Context Probleme. Da der Problemraum eines Open Context Problems, wie beispielswiese die Objekterkennung im autonomen Fahren, beliebig groß ist, können zum Spezifikationszeitpunkt nicht alle möglichen Szenarien umfassend berücksichtigt und geprüft werden. Es ist davon auszugehen, dass im Laufe der Betriebszeit einer KI/ML-basierten Software neue Szenarien identifiziert werden, die mit einem Sicherheitsrisiko verbunden sind. So hat in den letzten 10 Jahren das Verkehrsaufkommen elektrisch betriebener Kleinstfahrzeuge wie Fahrräder, Hoverboards, Roller, Skateboard etc. stark zugenommen. Anwendungen für das autonome Fahren, die vor 10 Jahren spezifiziert worden wären, hätten diese Entwicklung nicht im vollen Umfang vorwegnehmen und in ein entsprechendes System integrieren können.*

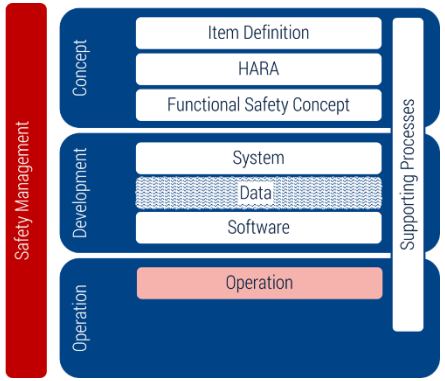
Risiko: Jedes Modell muss kontinuierlich beobachtet und gewartet werden, um mögliche Fehler im Feld zu identifizieren und um zukünftige Fehler zu antizipieren und zu vermeiden. Hierzu ist sicherzustellen, dass relevante und sicherheitskritische Fehlfunktionen im Feld erkannt, aufgezeichnet und in die Entwicklung zurückgespielt werden können. Algorithmische Entscheidungsvorgänge müssen kontinuierlich dokumentiert, gespeichert und überwacht werden. Das gilt vor allem für:

- Ausnahmesituationen und kritische Fehler
- Übergabevorgänge zwischen Menschen und Technik

Neue Risiken, die sich aus bisher nicht berücksichtigten oder neu entstandenen Umgebungssituationen ergeben, müssen identifiziert werden. Es ist sicherzustellen, dass für die Behebung von Fehlfunktionen zusätzliche Trainingsdaten erhoben werden, sodass über den Betriebszeitraum hinweg neue Modelle erstellt und für die Fahrzeuge im Feld bereitgestellt werden können. Das Management dieser Iterationen, die zu einem kontinuierlichen Lernen der Software führen kann, ist eine gemeinsame Aktivität der Qualitätssicherungs- und Entwicklungsabteilungen.

Betrieb, Service und Außerbetriebnahme

Absicherung der Betriebsphase (2/3)

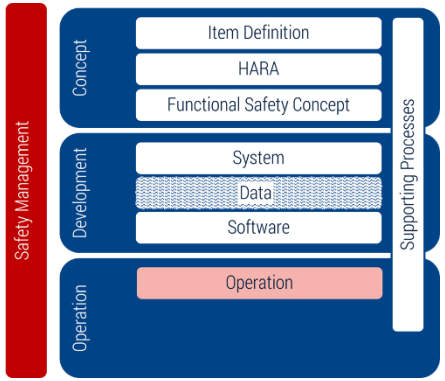


Maßnahmen:

- Technisches Monitoring (Distributional Shift Monitoring, Monitoring auf Sonderfälle und Fehlentscheidungen, ODD Monitoring, Übergabevorgänge)
- Rigides Change- und Updatemanagement über den gesamten Lebenszyklus eines Fahrzeugs.
- Realisierung eines effizienten, vertrauenswürdigen und sicheren Updatemechanismus für Fahrzeuge im Feld.
- Sichere Verfahren zur Prüfung der Sicherheit- und Integrität der aktualisierten Fahrzeuge im Feld.

Betrieb, Service und Außerbetriebnahme

Absicherung der Betriebsphase (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass die Planung für die Aktivitäten im Feld alle notwendigen Besonderheiten von ML-Systemen berücksichtigt wurden und ein konsistentes Überwachungs- und Aktualisierungskonzept vorliegt, sodass technische Fehler ausreichend schnell identifiziert und beseitigt werden können, kritische Übergänge zwischen Mensch und Maschine sicher gewährleistet werden können und alle Systeme dem Stand der Technik entsprechend und für die jeweilige Aufgabe angemessen aktualisiert werden können. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 7.7.5.1 Field observation instructions (ISO 26262)
- 13.5.1Field monitoring process (SOTIF)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 08-ML13 Beobachtungs- und Wartungsplan

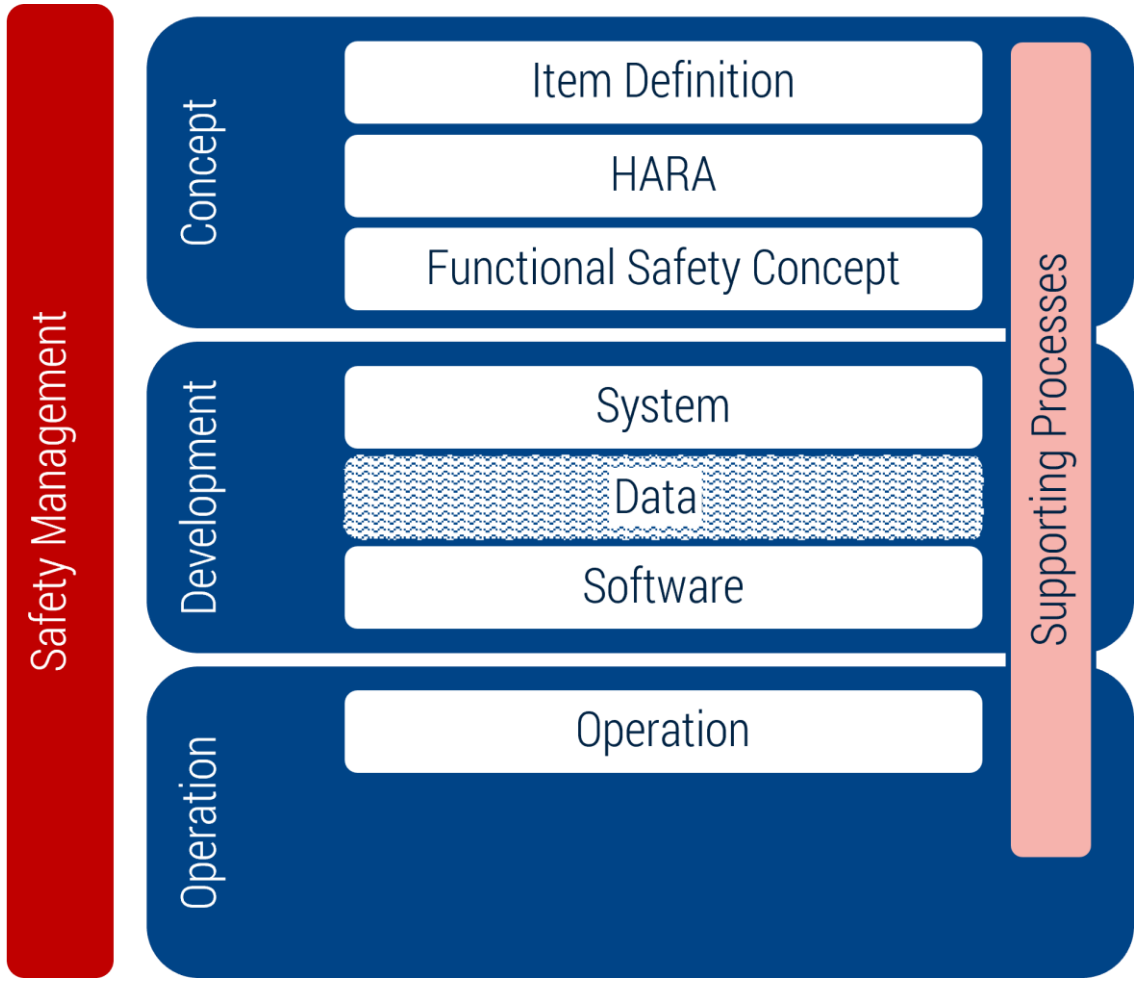
Zusätzlich gehen wir davon aus, dass allein die Planung sowie das Vorhandensein von Anweisungen für den Betrieb nicht ausreichend sind, um den Anforderungen eines sicheren Betriebs gerecht zu werden. Vielmehr bedarf es eines kontinuierlichen Sicherheitsmanagements für die Betriebsphase, mit der sichergestellt wird, dass alle Prozesse für Beobachtung, Wartung und Pflege eines ML-basierten Systems kontinuierlich bereitgestellt werden und in der Lage sind, die Risiken durch Verteilungsverschiebungen, nicht-berücksichtigte Sonderfälle und weiterer Fehler in einem dem Sicherheitszielen entsprechenden Niveau zu halten. Nachweise zur Wirksamkeit und Reife dieser Prozesse sind kontinuierlich über den gesamten Lebenszeitraum der Anwendung zu erbringen.

Berücksichtigt werden sollten dabei die folgenden Arbeitsprodukte, die wir als Teil des ML-Lebenszyklus im Referenzprozess ML definiert haben,

- 13-ML30 Fehlerberichte aus dem Feld (Referenzprozess ML)
- 15-ML09 Risikolagebild (Referenzprozess ML)
- 03-ML50 Rohdaten aus dem Feld (Referenzprozess ML)

Versionsmanagement

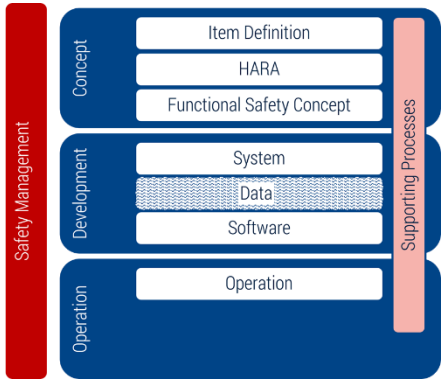
Aufgabe des Versionsmanagements ist es sicherzustellen, dass die Arbeitsprodukte, Funktionen bzw. Systeme (Item), und ihre Bestandteile sowie die Prinzipien und Rahmenbedingungen ihrer Erstellung jederzeit eindeutig identifiziert und kontrolliert reproduziert werden können und dass die Beziehungen und Unterschiede zwischen früheren und aktuellen Versionen nachvollziehbar sind.



zurück zur Navigation

Versionsmanagement

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



Herausforderung: Neben den typischen Entwicklungsartefakten wie Anforderungen, Quellcode und Dokumentation gibt es im ML-Lebenszyklus deutlich mehr Artefakte, die unter das Versionsmanagement fallen. Hierzu zählt die Versionierung der Rohdaten, die Versionierung wichtiger Arte-fakte der Datenaufbereitung (Label, andere Metadaten), sowie alle mit der Modellbildung zusammenhängenden Artefakte und Daten (Parameter, Hyperparameter, Modellarchitektur, Modelle, Testergebnisse).

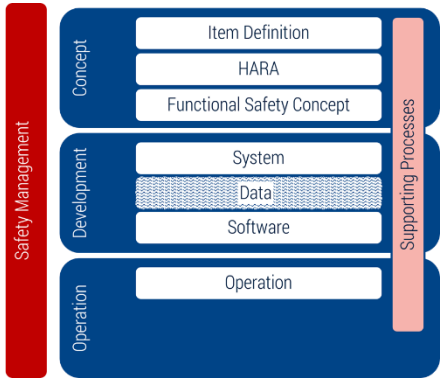
Weitere Aufgabe ist es, die Unterschiede zwischen zwei Modellversionen nachvollziehbar zu gestalten: Unterschiede in den Trainingsdaten, Hyperparameter, Test- und Validierungsergebnisse, KPIs.

***Beispiel**:: Beim maschinellen Lernen visueller Daten kommen schnell Datensätze in der Größenordnung mehrerer Terrabyte zusammen. Für die Beurteilung eines Modells ist es notwendig nachvollziehen zu können, wie das Modell trainiert wurde. Zu diesem Zweck muss der Datensatz, der bei jedem Trainingslauf verwendet wurde, versioniert und nachverfolgt werden können. Dies gibt sowohl Datenwissenschaftlern wie auch der gesamten Entwicklung die Flexibilität, zu einer früheren Version des Datensatzes zurückzukehren und diesen zu analysieren*

Risiko: Die Versionierung von Daten wird derzeit weder in der ISO26262 noch in der SOTIF Norm adressiert. Die Gefahr besteht, dass in der industriellen Praxis kein einheitliches Vorgehen zur Versionierung etabliert und ggfs. wichtige Artefakte aus der Versionierung herausfallen. Letzteres hat Konsequenzen für die Identifikation und Behebung möglicher Fehler sowie für den Nachweis der Sicherheit. Eine große Herausforderung bei der Versionierung der Trainingsdaten besteht in der schieren Menge der Daten, die insbesondere für Einsatzbereiche wie das automatisierte Fahren verwendet werden müssen. Beispiele zeigen, dass bei größeren Flotten Hunderte Petabyte pro Jahr erfasst werden und in die KI-Datenpipeline zur Auf-bereitung und Kennzeichnung aufgenommen werden. Aus diesem Datenbestand werden qualitätsgekennzeichnete Daten ausgewählt und für Training und Tests verwendet.

Versionsmanagement

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



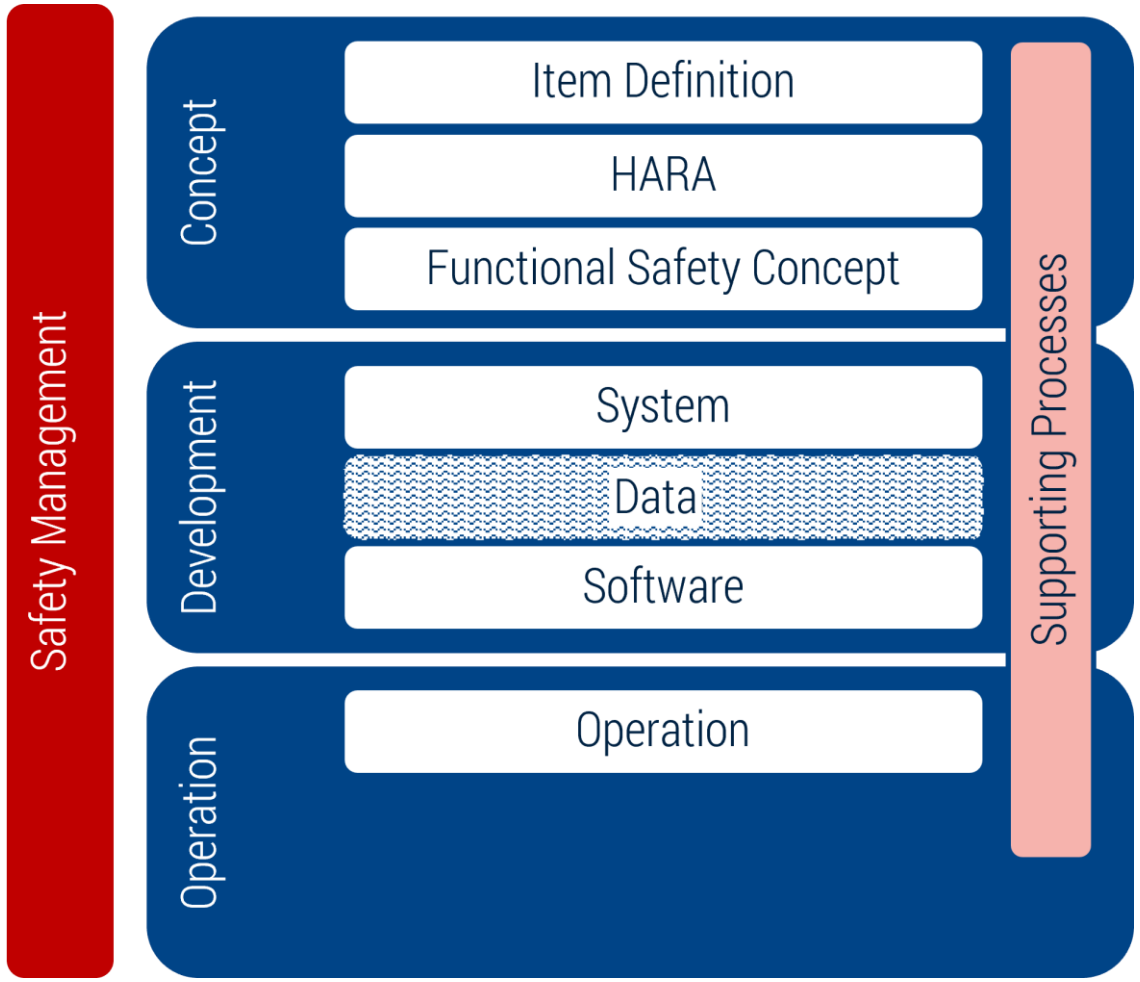
Maßnahmen: Die Grundlage für die Entwicklung von KI-Modellen sind Daten, die in großen Mengen verarbeitet werden, um neuronale Netze zu trainieren. Die Effizienz der KI-Entwicklung ist abhängig von der Effizienz der Datenpipeline. An die Speicherung der Daten ergeben sich insbesondere Anforderungen nach Skalierbarkeit in Bezug auf die Leistungsfähigkeit und Kapazität der Dateninfrastruktur, Effizienz der Datenspeicherung und -verwaltung, Interoperabilität und Flexibilität für den Datenzugriff und den Datenaustausch sowie eine langfristige und kostengünstige Speicherung unter Berücksichtigung von gesetzlichen Auflagen (siehe Ziegler et al. [52]). Versionskontrolle der Daten: Datensammlung und Datenaufbereitung ist keine statische, sondern eine iterative Aufgabe. Änderungen am Datensatz sollten dokumentiert werden, um das Risiko nicht reproduzierbarer oder falscher Ergebnisse zu verringern. Die Versionskontrolle der Daten ist eines der wichtigsten Werkzeuge zur Sicherstellung der Reproduzierbarkeit und Qualität, da es eine systematische Versionskontrolle ermöglicht, Fehler und ungünstige Änderungen während der Entwicklung zu verfolgen und evtl. rückgängig zu machen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass das Konfigurationsmanagement nicht nur die klassischen Entwicklungsartefakte berücksichtigt, sondern in der Lage ist auch die für Training, Test und Validierung im ML notwendigen Daten zu erfassen und zu verwalten. Änderungen an den Datensätzen (gelöschte Daten, hinzugefügte Daten) sowie alle relevanten Verarbeitungsschritte müssen dokumentiert und über das Konfigurationsmanagement nachvollziehbar werden. Das Konfigurationsmanagement muss den gesamten Lebenszyklus einer ML-basierten Anwendung umfassen. Eine entsprechende Dokumentation der Herangehensweise, des Umgangs und der für das Konfigurationsmanagement vorgesehenen Techniken erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 8.7.5.1 Configuration management plan (ISO26262)

Change-Management

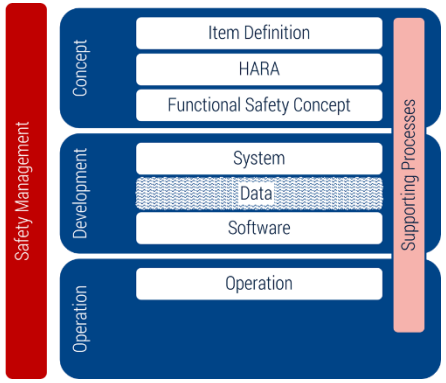
Aufgabe des Change-Managements ist es, Änderungen an sicherheitsrelevanten Arbeitsprodukten, Funktionen bzw. Systeme (Item), und ihre Bestandteile während des gesamten Sicherheitslebenszyklus zu analysieren und zu steuern.



zurück zur Navigation

Versionsmanagement

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



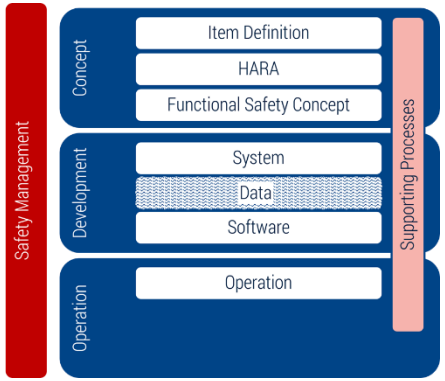
Herausforderung: Sicherheitsrelevante Arbeitsprodukte und Artefakte müssen als solche benannt und gekennzeichnet werden. Änderungen an solchen Arbeitsprodukten müssen im Change-Management gesondert berücksichtigt werden und in Hinblick auf die Kritikalität der Änderungen prüf- und nachvollziehbar dokumentiert werden.

***Beispiel**: Beim maschinellen Lernen kann nicht davon ausgegangen werden, das wenn im Trainingsprozess neue Datensätze hinzugefügt werden, z.B. um neue, bisher nicht berücksichtigte Objekte erkennen zu können, die Erkennungsleistung für die zuvor bereits berücksichtigten Objekte konstant bleibt. Maschinelles Lernen ist ein Optimierungsprozess, bei dem Änderungen an den Hyperparametern und Daten in ihrer Wirkung nicht notwendigerweise lokal eingrenzbar sind. Änderungen an den Trainingsdaten und Hyperparametern müssen entsprechend gut dokumentiert und durch geeignete Prüfungen in ihrer Auswirkung abgesichert werden. Beim maschinellen Lernen kann nicht davon ausgegangen werden, das wenn im Trainingsprozess neue Datensätze hinzugefügt werden, z.B. um neue, bisher nicht berücksichtigte Objekte erkennen zu können, die Erkennungsleistung für die zuvor bereits berücksichtigten Objekte konstant bleibt. Maschinelles Lernen ist ein Optimierungsprozess, bei dem Änderungen an den Hyperparametern und Daten in ihrer Wirkung nicht notwendigerweise lokal eingrenzbar sind. Änderungen an den Trainingsdaten und Hyperparametern müssen entsprechend gut dokumentiert und durch geeignete Prüfungen in ihrer Auswirkung abgesichert werden.*

Risiko: Kleine Änderungen an Artefakten im Trainingsprozess (Datenvorverarbeitung, Hyperpara-meter, Modellarchitektur) können ggfs. größere Auswirkungen auf die Qualitätseigenschaften einen ML-Modells haben. Aufgrund des Fehlens konkreter Change-Management Strategien im ML-Trainingsprozess, können unmoderierte Änderungen aufgrund der komplexen Abhängigkeiten zwischen Datenaufbereitung und Training zu Sicherheitsrisiken führen, die spät oder gar nicht identifiziert werden können.

Versionsmanagement

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



Maßnahmen: Einführen eines Change und Impact Management für ML-Artefakte. Hierzu zählt das Erfassen aller Funktionsspezifischen Design- und Trainingsentscheidungen, Implementierungen sowie Parametrisierungen.

- Change und Impact Management im Rahmen der Aufbereitung der Trainingsdaten
- Change und Impact Management für die Auswahl der Trainingsdaten
- Change und Impact Management für Architektur, Implementierung und Parametrisierung der Modelle

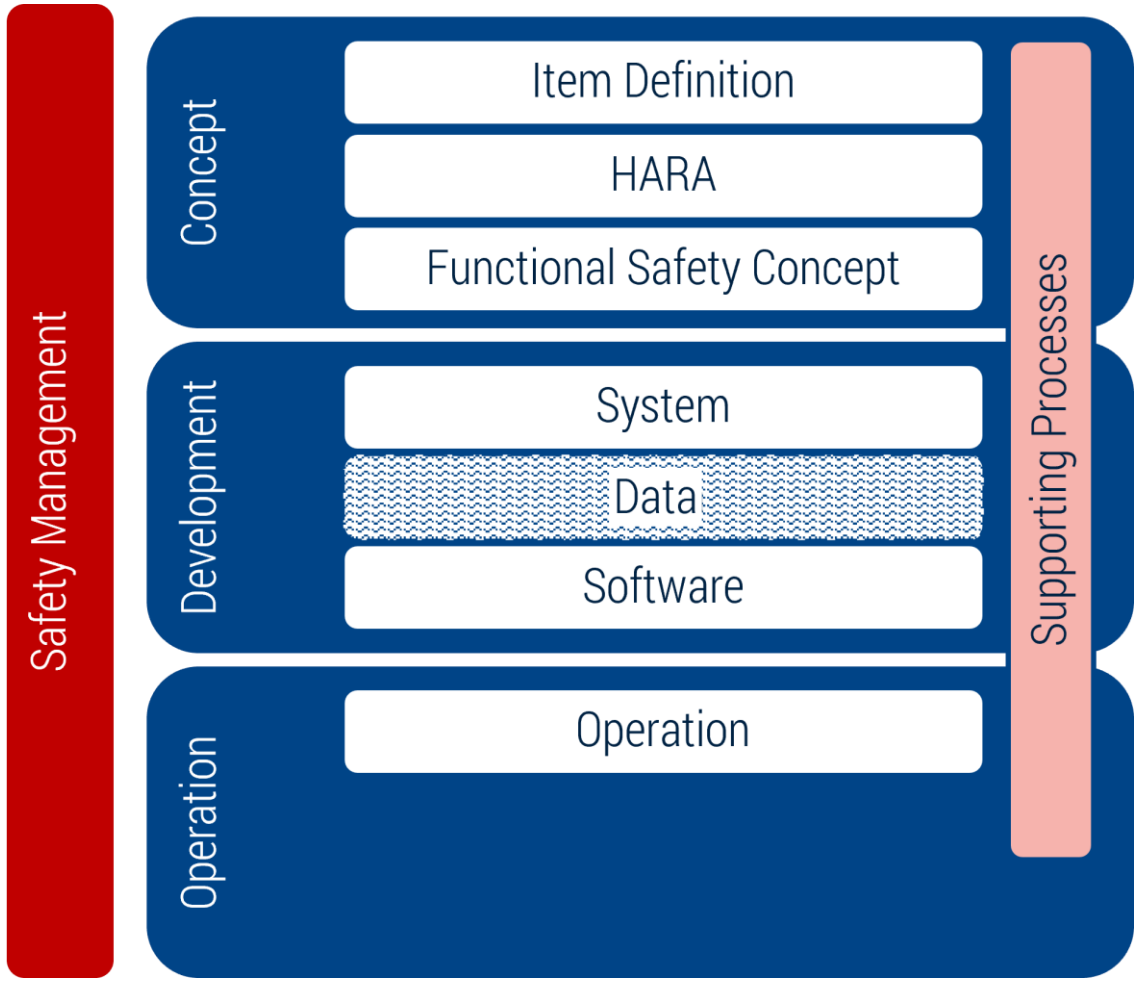
ISO TR 4804 gibt es eine initiale Liste mit sicherheitsrelevanten Arbeitsprodukten und Artefakte. Diese muss in Hinblick auf die Bedeutung für das Change-Management erweitert und mit konkreten Maßnahmen zur Absicherung und Dokumentation der Änderungen hinterlegt werden.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Die Anwendung von ML führt neue Prozesse und Arbeitsprodukte in den Entwicklungs- und Lebenszyklus sicherheitskritischer Softwareanwendung ein. Abhängigkeiten zwischen den Ergebnissen dieser Prozesse und Arbeitsprodukte zu den bereits etablierten Prozessen und Arbeitsprodukten führen dazu, dass Änderungen systematisch auf ihre Auswirkungen in Bezug auf die Funktionssicherheit analysiert und entsprechend vorsichtig propagiert werden müssen. Im Rahmen des Änderungsmanagements ist nachzuweisen, dass alle sicherheitsrelevanten Arbeitsprodukte aus dem ML-Lebenszyklus erfasst und mit geeigneten Maßnahmen zum Änderungsmanagement hinterlegt sind. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 8.8.5.1 Change management plan (ISO 26262)
- 8.8.5.3 Impact analysis and change request plan (ISO 26262)

Verifikation

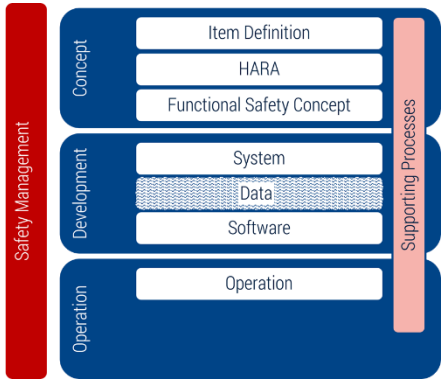
Verifikationsaktivitäten überspannen alle relevanten Phasen des Sicherheitslebenszyklus und prüfen die Korrektheit, Vollständig und Konsistenz der Entwicklungsartefakte. In der Konzepthase werden Korrektheit, Vollständig und Konsistenz des Konzepts sowie der definierten Randbedingungen überprüft. In der Produktentwicklungsphase werden Anforderungsspezifikation, Architekturentwurf, Modelle oder Softwarecode bewertet während in der Testphase das zu integrierende System insbesondere gegen seine Anforderungen geprüft wird.



zurück zur Navigation

Verifikation

Bereitstellung geeigneter V&V Verfahren und Strategien (1/3)



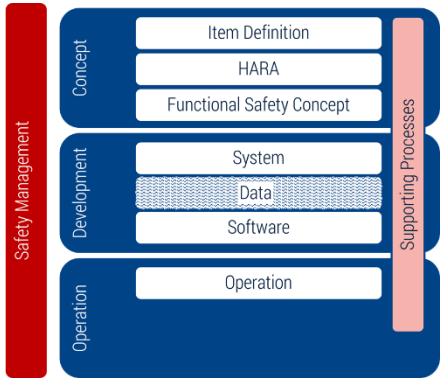
Herausforderung: Für die Zulassung ML-basierter Software in sicherheitskritischen Anwendungen fehlt bisher eine durchgängige Verifikationsstrategie. Testen sowie klassische Verfahren der formalen Verifikation wie beispielsweise Model Checking, Theorem Proving und abstrakte Interpretation, sind für ML-Modelle und ihre Funktionalität technisch nur bedingt anwendbar. Darüber hinaus wird ML i.d.R. für Open Context Anwendungen verwendet, die eine hohe Komplexität besitzen und häufig nicht vollständig spezifizierbar sind. Verifikationsverfahren, die eine vollständige Überdeckung der Spezifikation zum Ziel haben, bleiben notwendigerweise unvollständig. Schlussendlich sind Daten und die Datenqualität ein entscheidender Faktor für die funktionale Sicherheit im Lebenszyklus einer ML-Anwendung. Zurzeit gibt es jedoch wenig Erfahrungen dazu, welche Validierungs- und Verifikationsstrategien für Daten im Kontext sicherheitskritischer Softwareanwendungen sinnvoll und im Zusammenspiel mit anderen Verifikationsverfahren (Modellverifikation, Softwareverifikation) eingesetzt werden können.

Beispiel**:: Grundlage für die Verifikation eines Systems ist die Möglichkeit, die Korrektheit eines Systems spezifizieren zu können. Das bedeutet, dass für jeden Eingangszustand definiert werden kann, was das richtige bzw. wahre Systemverhalten ist. Aufgrund des komplexen Eingaberaums eines autonomen Fahrzeugs ist dieses jedoch nicht möglich. Es wird immer wieder Szenarien und Konstellationen geben (notlandende Flugzeuge auf der Autobahn, mit Mustern beklebte Straßenschilder), die im Rahmen der Spezifikation nicht abgedeckt wurden. **Eine Verifikation auf Basis einer unvollständigen Spezifikation ist per se unvollständig**. Darüber hinaus ist maschinelles Lernen ist ein statistischer Lösungsansatz, bei dem bereits im Rahmen des Trainings akzeptiert wird, dass es falsche Entscheidungen gibt. **Das Konzept der Verifikation ist insofern durch eine systematische Validierung zu ersetzen, die mittels Benchmarking und auf Basis empirischer Daten die Leistungsfähigkeit eines ML-basierten autonomen Fahrzeugs mit konkurrierenden Lösungsansätzen (menschlicher Fahrer, Fernsteuerung) vergleicht.

Risiko: Eine durchgängige Verifikationsstrategie ist die Grundlage dafür, Software in sicherheitskritischen Anwendungsbereichen zulassen und in Verkehr bringen zu können. Eine solche Strategie besteht aus Planung unterschiedlicher Verifikationstechniken, die zu unterschiedlichen Zeitpunkten und Phasen angewendet werden müssen, sodass alle Verifikationsanforderungen nachvollziehbar erfüllt sind. Derzeit fehlt sowohl die Erfahrung, welche Verifikationstechniken sinnvoll für ML-basierte Systeme eingesetzt werden können, als auch die Erfahrung, wie diese Techniken mit den bestehenden Verifikationsansätzen kombiniert werden können, um eine vollständige Verifikation des Gesamtsystem zu ermöglichen.

Verifikation

Bereitstellung geeigneter V&V Verfahren und Strategien (2/3)

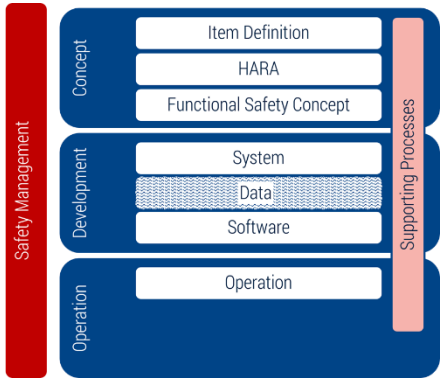


Maßnahmen:

- Bereitstellen geeigneter Datenvalidierungs- und -verifikationsansätze, um Vollständigkeit, Korrektheit und Konsistenz der Trainings-, Test- und Validierungsdaten nachweisen zu können (siehe auch Kapitel Produktentwicklung Daten).
- Bereitstellen von Verifikationsansätzen, die auf ML-Teilprobleme zugeschnitten sind. Hierzu zählen Testverfahren und Ansätze zur formalen Verifikation neuronaler Netze (siehe auch Kapitel Software Unit Verification bzw. Software-Integration und Verifizierung).
- Bereitstellen von Best Practices, wie einzelne Verifikationsansätze zu Teilaspekten des ML (Datenkonsistenz und -korrektheit, Robustheit, Fairness) miteinander und mit den bestehenden Verifikationsverfahren für Software (Softwaretesten, formale Verifikation der Software) sinnvoll zu kombinieren sind.
- Bereitstellen von Vollständigkeitsmaßen, zur Bewertung der Vollständigkeit einzelner Verifikationsansätze sowie der gesamten Verifikationsstrategie.
- Sinnvolle Verzahnung von Verfikations- und Validierungsansätzen

Verifikation

Bereitstellung geeigneter V&V Verfahren und Strategien (3/3)



Anforderungen bzgl. Dokumentations- und Nachweispflicht: Im Rahmen der Verifikationsstrategie muss nachgewiesen werden, dass die verwendeten Verfahren und Technologien geeignet sind, die funktionale Sicherheit der Anwendung nachvollziehbar zu belegen. Da derzeit für ML keine anerkannten Standards und Best Practices existieren, die als Referenz für eine solche Strategie herangezogen werden können, muss für jedes Produkt und für jede Anwendung explizit argumentiert werden, warum der gewählte Verifikationsansatz ausreichend ist. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

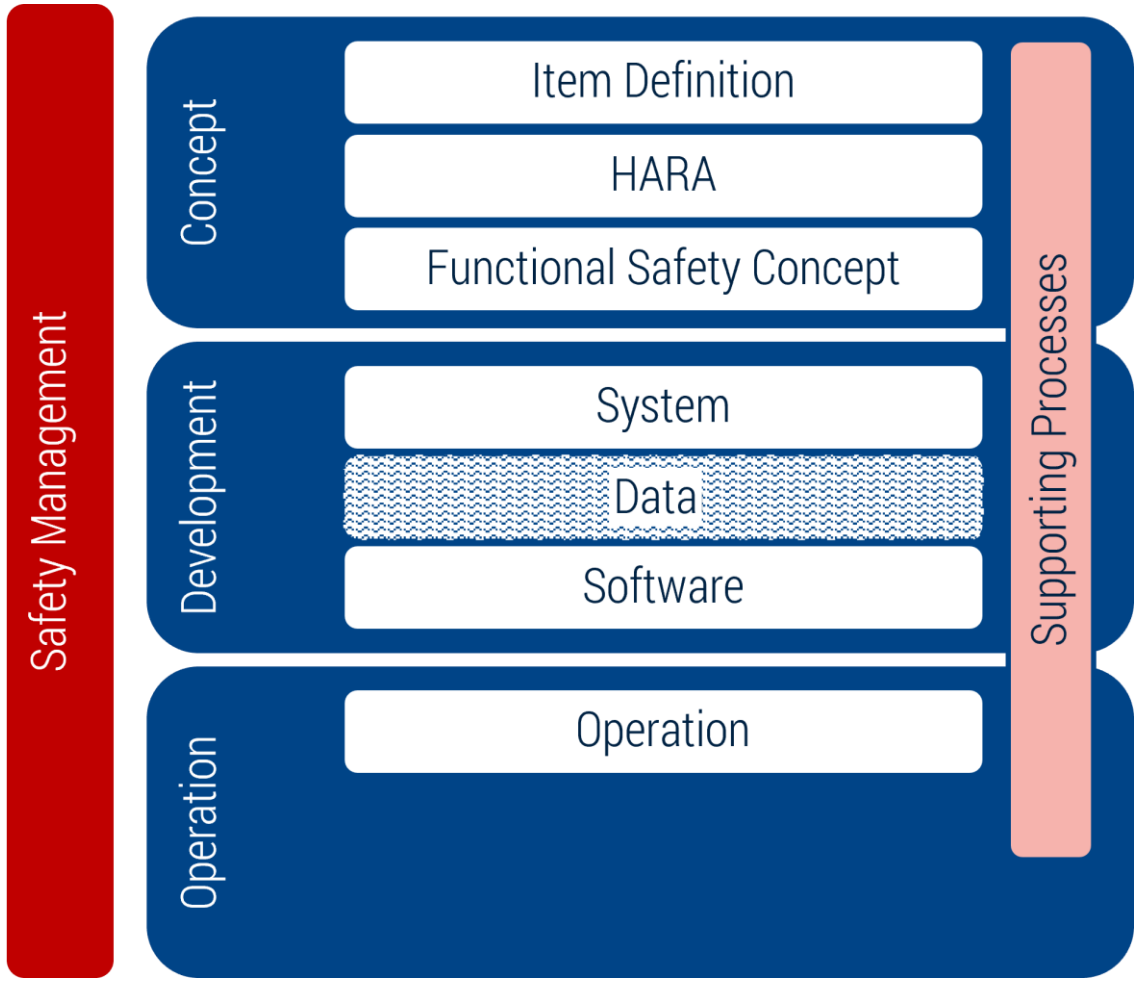
- 8.9.5.1 Verification plan (ISO 26262)
- 8.9.5.2 Verification specification (ISO 26262)
- 8.9.5.3 Verification report (ISO 26262)

Berücksichtigt werden dabei die folgenden Arbeitsprodukte aus dem ML-Lebenszyklus:

- 7-ML02 Verifikations- und Validierungsanforderungen ML-Modell und Daten (Referenzprozess ML)
- 08-ML30 ML-Modell Verifikations- und Validierungsplan (Referenzprozess ML)
- 08-ML50 Spezifikation der ML-Modell Verifikation und Validierung (Referenzprozess ML)
- 13-ML50 Verifikations- und Validierungsergebnisse (Referenzprozess ML)
- 15-ML50 Bericht der Modellvalidierung und -verifikation (Referenzprozess ML)

Dokumentation

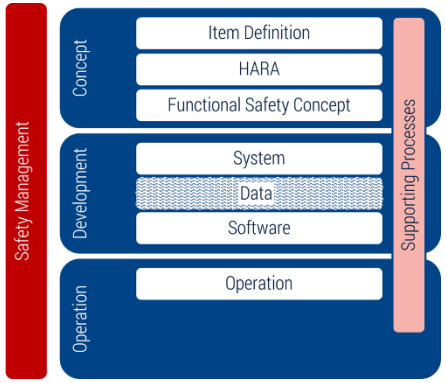
Ziel ist die Entwicklung einer Dokumentationsmanagementstrategie für den gesamten Sicherheitslebenszyklus, um einen effektiven und wiederholbaren Dokumentationsmanagementprozess zu ermöglichen.



zurück zur Navigation

Dokumentation

Beherrschung datenintensiver Entwicklungsprozesse (1/2)



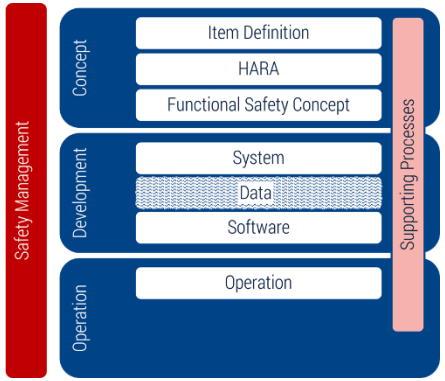
Herausforderung: Dokumentationspflichten für den Trainingsprozess (Daten, Versionen, Kriterien) sind derzeit nicht vollständig geklärt. Es muss definiert werden, welche Artefakte aus dem ML-Lebenszyklus im Rahmen einer Dokumentationsmanagementstrategie erfasst und dauerhaft dokumentiert werden müssen.

***Beispiel**:: Bei der Erstellung eines ML-Modells zur Objekterkennung werden verschiedene Kandidatenmodelle erzeugt, alle mit unterschiedlichen Hyperparmeterkonfigurationen sowie Variationen in der Aufteilung der Trainingsdaten. Wenn im Rahmen der Dokumentation des Trainingsprozesses die Kriterien für die Auswahl des optimalen Modells transparent gemacht werden soll, kann das im Prinzip nur durch Bereitstellung des Benchmarks mit den Kandidatenmodellen erfolgen. In welchem Umfang hierzu die verschiedenen Kandidatenmodelle, ihre Konfiguration sowie die Aufteilung der Trainingsdaten vorgehalten werden muss, ist nicht geklärt.*

Risiko: Der ML-Lebenszyklus ist ein komplexer, werkzeugbasierter Prozess mit großen Abhängigkeiten zwischen den Werkzeugen (Datenaufbereitung, Simulationen) und Arbeitsprodukten (Daten, Modellen). Sind die Dokumentations- und Vorhaltepflichten für Arbeitsprodukte und Werkzeuge nur unzureichend definiert, kann durch das Fehlen wichtiger Arbeitsprodukte oder Werkzeuge die Nachvollziehbarkeit der Entwicklungs- und Absicherungsmaßnahmen nicht mehr gewährleistet sein.

Dokumentation

Beherrschung datenintensiver Entwicklungsprozesse (2/2)



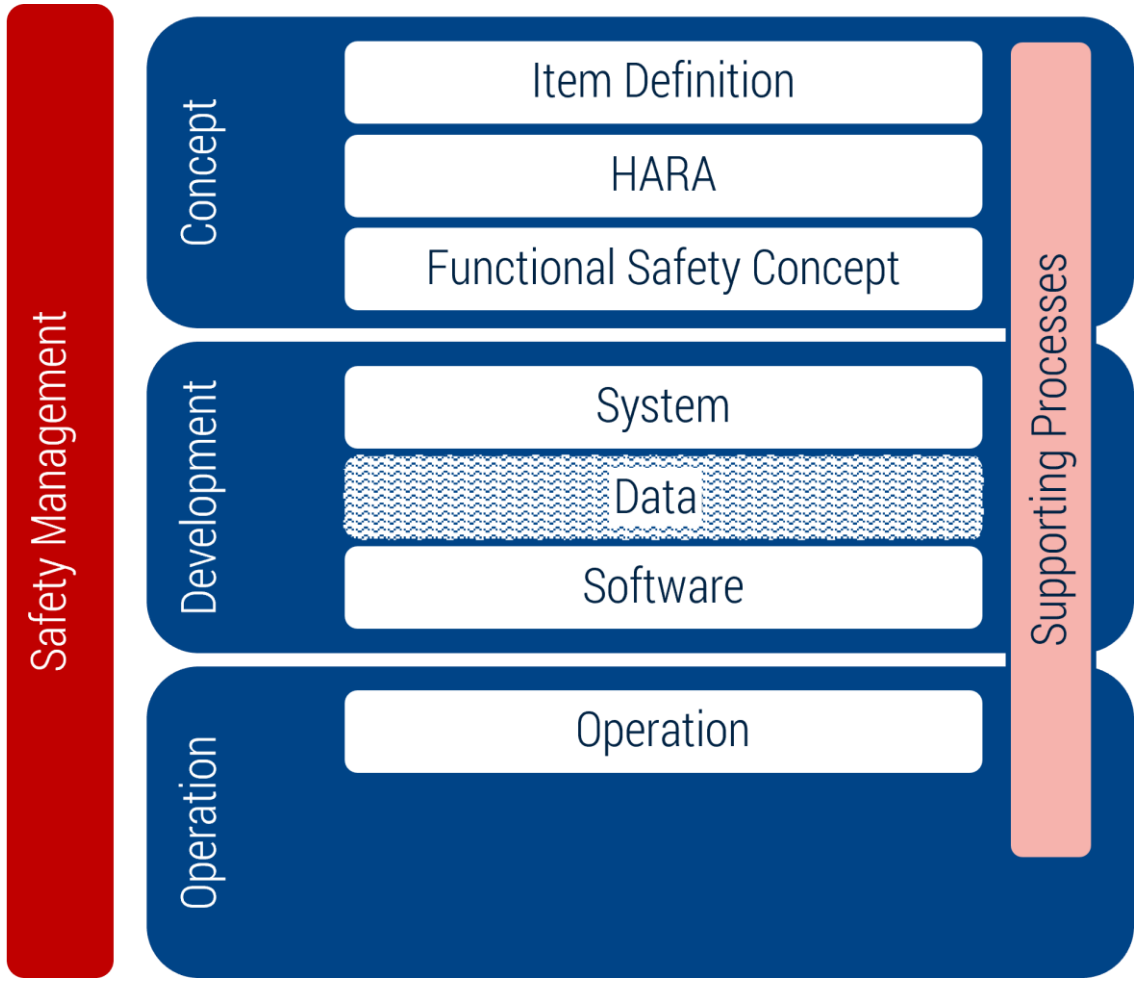
Maßnahmen: Die Nachverfolgung der Leistungsfähigkeit eines Modells und ihrer Genese durch vorangegangene Modelländerungen und -anpassungen ermöglicht das Verständnis des Modells, indem festgestellt wird, welche Änderungen vorteilhaft waren und die Gesamtqualität des Modells verbessern. Die Dokumentation sollte die aufgelisteten Eigenschaften in der Aufgabe der Reproduzierbarkeit der Methode enthalten. Ein werkzeugbasierter Ansatz zur Versionskontrolle und zum Umgang mit Metadaten beim Experimentieren mit ML-Modellen und Hyperparametern existiert (CrispMLq).

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Im Zuge der vorhergehenden Kapitel wurde im Einzelnen erläutert, in welchen Phasen des Absicherungsprozesses neue Anforderungen an die Dokumentation der Arbeits- und Prüfergebnisse entstehen. Es wurde insbesondere darauf hingewiesen, dass speziell für die Phase der Datenbeschaffung und Datenaufbereitung keine allgemein anerkannten Referenzmodelle existieren, die als eine Vorlage zur Dokumentation und Nachweisführung verwendet werden können. Eine sinnvolle Grundlage für die Definition eines solchen Dokumentationsvorgehens bilden die Definition von Arbeitsprodukten im ML-Lebenszyklus, wie sie in ISO TR 4804, dem Referenzprozess ML (Vorgängerprojekt) sowie dieser Arbeit definiert wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 8.10.5.1 Documentation management plan (ISO 26262)
- 8.10.5.2 Documentation guideline (ISO 26262)

Werkzeugqualifizierung

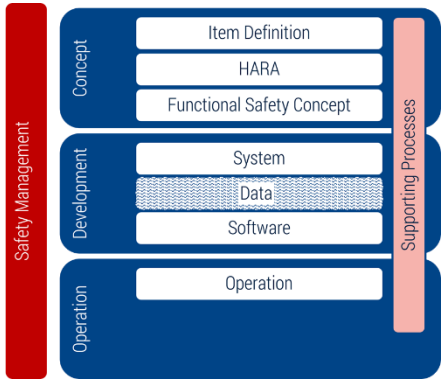
Ziel der Werkzeugqualifizierung ist es, Kriterien zur Bestimmung des erforderlichen Vertrauensniveaus in ein Software-Werkzeug bereitzustellen und gegebenenfalls Mittel für die Qualifizierung des Software-Werkzeugs bereitzustellen, um den Nachweis zu erbringen, dass das Software-Werkzeug geeignet ist, zur Unterstützung der von der Normenreihe ISO 26262 geforderten Tätigkeiten oder Aufgaben eingesetzt zu werden (d.h. der Benutzer kann sich bei den von der Normenreihe ISO 26262 geforderten Tätigkeiten oder Aufgaben auf das korrekte Funktionieren eines Software-Werkzeugs verlassen).



zurück zur Navigation

Werkzeugqualifizierung

Absicherung werkzeugintensiver Prozesse und komplexer Softwareframeworks (1/2)



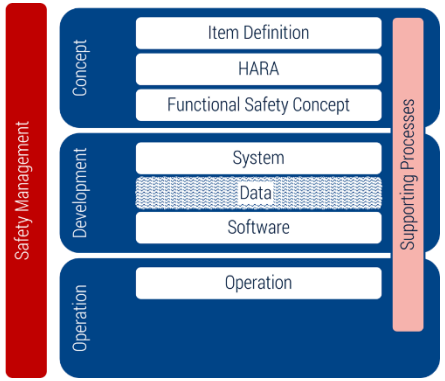
Herausforderung: Datenbeschaffung, Datenaufbereitung, Training und die Verifikation von ML-Modellen ist ein komplexer, werkzeugintensiver Prozess. Die verwendeten Werkzeuge unterscheiden sich deutlich von den Werkzeugen der klassischen Softwareentwicklung und sind entscheidend für die Daten- und Modellqualität.

***Beispiel**:: Eine Simulationsumgebung zur Erzeugung fotorealistischer Verkehrssituationen weist einen systematischen Fehler auf, der Szenarien mit Fußgängern mit einem subtilen Muster kennzeichnet, das durch die KI erkannt wird, nicht aber durch den Menschen. Die Artefakte treten nur bei 30 Prozent der Daten auf, weil unterschiedliche Simulationsumgebungen verwendet wurden. Die Erkennungsleistung der KI wird dadurch positiv verfälscht und kann in der physikalischen Umgebung erst nach der Inbetriebnahme erkannt werden, weil die dafür im Testbetrieb realisierten Testfahrten für das Erkennen dieses Fehlers nicht umfangreich genug waren.*

Risiko: Aktuell fehlen auf die Werkzeugketten des ML zugeschnittene Kriterien zur Bestimmung des erforderlichen Vertrauensniveaus in ein Software-Werkzeug. Dieses ist insofern kritisch, weil sowohl die Datenaufbereitung als auch die synthetische Datengenerierung werkzeugintensive Prozesse sind, deren Reife und Qualität einen entscheidenden Beitrag in der Erstellung sowie der Verifikation ML-basierter Software liefern.

Werkzeugqualifizierung

Absicherung werkzeugintensiver Prozesse und komplexer Softwareframeworks (2/2)



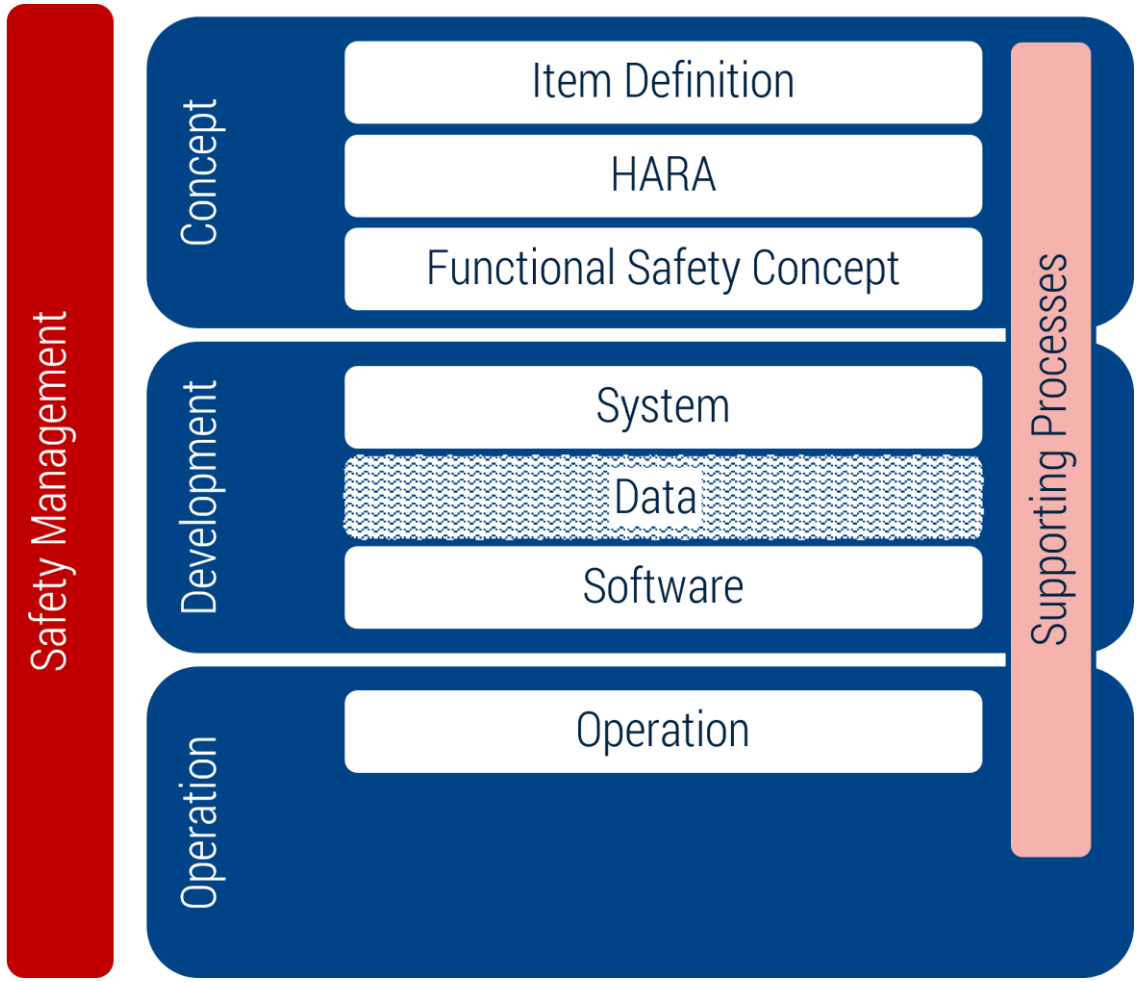
Maßnahmen:

- Definition von Kriterien zur Qualifizierung der Datenpipeline und des Trainingsframeworks
- Definition von Kriterien zur Qualifizierung von Werkzeugen zur synthetischen Datengenerierung (Simulationsumgebungen)

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Im Zuge der vorhergehenden Kapitel wurde im Einzelnen erläutert, in welchen Phasen des Absicherungsprozesses neue Anforderungen an die Dokumentation der Arbeits- und Prüfergebnisse entstehen. Es wurde insbesondere darauf hingewiesen, dass speziell für die Phase der Datenbeschaffung und Datenaufbereitung keine allgemein anerkannten Referenzmodelle existieren, die als eine Vorlage zur Dokumentation und Nachweisführung verwendet werden können. Eine sinnvolle Grundlage für die Definition eines solchen Dokumentationsvorgehens bilden die Definition von Arbeitsprodukten im ML-Lebenszyklus, wie sie in ISO TR 4804, dem Referenzprozess ML (Vorgängerprojekt) sowie dieser Arbeit definiert wurden. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 8.10.5.1 Documentation management plan (ISO 26262)
- 8.10.5.2 Documentation guideline (ISO 26262)

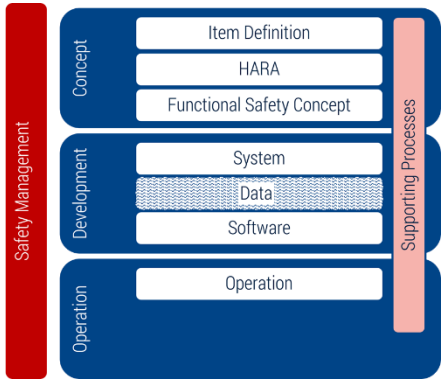
Qualifizierung von Softwarekomponenten



zurück zur Navigation

Werkzeugqualifizierung

Absicherung werkzeugintensiver Prozesse und komplexer Softwareframeworks (1/2)



Herausforderung: Je komplexer eine Softwareanwendung wird, umso stärker steigt der Bedarf die Software, die in anderen Anwendungsgebieten erfolgreich eingesetzt wurde, wiederzuverwenden.

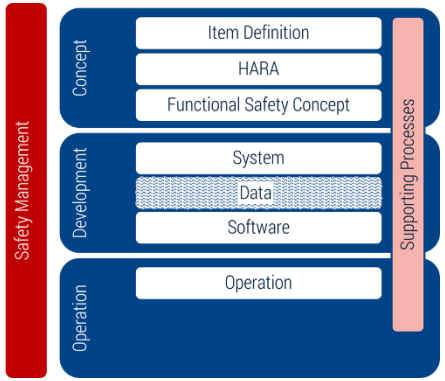
Beispiel:: *Transferlernen ist eine Technik, mit der Modelle auf großen Quelldatensätzen vortrainiert werden und mit Zieldatensätzen für spezielle Aufgaben abgestimmt werden. Die Motivation dahinter ist, Modelle mit Allzweckfähigkeiten auszustatten und das Wissen auf nachgelagerte Aufgaben zu übertragen. So ist es Beispielsweise vorstellbar ein Modell zur Objekterkennung auf die zu erkennenden Objekte vorzutrainieren und dann für verschiedene Kamerasysteme anzupassen bzw. für den Einsatz auf ressourcenbeschränkten Geräten zu quantisieren. Ein solcher Prozess kann Ressourcen sparen, muss aber in seiner Komplexität und Fehleranfälligkeit begriffen und entsprechend reguliert werden.*

Risiko: Bei der Verwendung von ML-basierter Software sind hier insbesondere die beiden folgenden Wiederverwendungsszenarios relevant.

- Wiederverwendung von vortrainierten Modellen, die bereits gewünschte Eigenschaften besitzen und durch Nachtrainieren auf den konkreten Anwendungsfall angepasst werden.
- Wiederverwendung des gesamten Software-Stacks bestehend aus Standardsoftware zum Training und zur Ausführung eines Modells (OS, ML-Framework, Hochsprachen)

Werkzeugqualifizierung

Absicherung werkzeugintensiver Prozesse und komplexer Softwareframeworks (2/2)



Maßnahmen:

- Definition von Kriterien zur Qualifizierung von ML-Modellen.
- Definition von Kriterien zur Qualifizierung von ML-Frameworks (Ausführungsumgebung).
- Definition von Kriterien zur Qualifizierung von Datensätzen.

Anforderungen bzgl. Dokumentations- und Nachweispflicht: Es ist nachzuweisen, dass wiederverwendete Arbeitsprodukte und Frameworks frei von systematischen Fehlern sind. Zu diesem Zweck sind für alle wiederverwendeten Arbeitsprodukte und Frameworks die Dokumentationen und Qualifizierungsreport bereitzustellen, die eine ausreichende Reife der wiederverwendeten Arbeitsprodukte und Frameworks argumentieren. Eine entsprechende Dokumentation erfolgt im Rahmen der folgenden Arbeitsprodukte:

- 8.12.5.1 Software component documentation (ISO 26262)
- 8.12.5.2 Software component qualification (ISO 26262)
- 8.12.5.3 Software component qualification verification (ISO 26262)

Da in der ISO 26262 allein die Wiederverwendung von Softwarekomponenten vorgesehen ist, schlagen wir vor, zusätzliche Qualifizierungsmaßnahmen und Dokumente für Daten und Modelle zu ergänzen.

Zusammenfassung und Handlungsbedarf

zurück zur Navigation

Zusammenfassung und Handlungsbedarf

Beherrschung offener Einsatzumgebungen und neuer Technologierisiken

AQI Automotive Quality Institute

- Nachweis, dass alle relevanten Szenarien und Rahmenbedingungen aus der geplanten Betriebsumgebung nachvollziehbar dokumentiert sind und Nachgewiesen wird, dass das alle Betriebsrisiken im geforderten Umfang gemindert wurden
- Nachweis, dass alle technologiespezifischen Risiken (nicht-determinismus, BIAS, fehlende Robustheit) adressiert und durch Gegenmaßnahmen und Prüfungen im geforderten Umfang gemindert werden konnten.

Dokumentation erfolgt über: Erweiterte Arbeitsprodukte der Konzept und Entwicklungsphase (SOTIF, ISO 26262)

Maschinelles Lernen **wird insbesondere für Aufgaben eingesetzt, die nur unvollständig beschrieben werden können und mit stochastischen Lösungsverfahren bewältigt werden können.**

Zusammenfassung und Handlungsbedarf

Beherrschung datenintensiver Entwicklungsprozesse

In der Nachweisführung ist die hervorgehobene Bedeutung der Trainingsdaten, ihrer Auswahl und Aufbereitung Rechnung zu tragen. Hierzu zählt:

- Schaffung von Referenzprozessen für Datenbereitstellung und ML zur Definition des State of the Art bzw. als Grundlage für Qualitätsbetrachtungen auf Prozessebene.
- Definition von wohldefinierten Arbeitsprodukten im Referenzprozess Datenbereitstellung und ML als Grundlage für eine Dokumentation der datenbezogenen Aktivitäten und ihrer Ergebnisse
- Definition von Datenmanagement-und Datenqualitätssicherungsprozessen inklusive Versions- und Change-Management für Daten

Die Dokumentation der Nachweise erfolgt über neu zu definierende Arbeitsprodukte aus Referenzprozesses für das ML..

ML basiert auf der Verarbeitung großer Datenmengen. Die Daten müssen mit geeigneten Verfahren und Werkzeugen erzeugt, aufbereitet, verwaltet und qualitätsgesichert werden. Entsprechende Prozesse, Verfahren und Arbeitsprodukte müssen Gegenstand der Absicherung werden.

Zusammenfassung und Handlungsbedarf

Bereitstellung geeigneter V&V Verfahren und Strategien

AQI Automotive Quality Institute

- Bereitstellung von Verfahren zu Datenvalidierung, die geeignet sind die Korrektheit, Konsistenz, Verzerrungsfreiheit und Vollständigkeit zu belegen.
- Bereitstellung von Verfahren zur Verifikation und Validierung von Modellen, die dazu geeignet sind Robustheit und Zuverlässigkeit von ML Modellen zu belegen.
- Bereitstellung von Verfahren zur Verifikation und Validierung in der Integration, die dazu geeignet sind (a) die Korrektheit, Robustheit und Zuverlässigkeit der Integration der von ML-Modellen in der Verarbeitungskette von der Sensorik zur Aktuatorik sicherzustellen und (b) das Zusammenspiel zwischen algorithmischen Entscheider und softwaretechnischen Absicherungsverfahren (Redundante Entscheider, Sicherheitskäfige, Fail Operational/Fail Safe Modi) zu belegen.
- **Bereitstellung geeigneter Validierungsverfahren, mit denen Open Context Systeme auf Basis empirischer Daten sinnvoll validiert und verglichen werden können.**

Die Dokumentation der Nachweise erfolgt über erweiterte Arbeitsprodukte der Konzept und Entwicklungsphase (ISO 26262, SOTIF) sowie den neu definierten Arbeitsprodukten eines Referenzprozesses ML .

Aktuell gibt es für ML-basierte Software keine etablierten Prüfverfahren, die geeignet wären, die sicherheitsrelevanten Eigenschaften eines ML-Modells und seiner Integration ausreichend nachzuweisen.

Zusammenfassung und Handlungsbedarf

Absicherung werkzeugintensive Prozesse und komplexe Softwareframeworks

Automotive Quality Institute

Es ist nachzuweisen, dass die hochgradig automatisierten Prozesse in der Datenbereitstellung und Erzeugung (z.B. Simulatoren), der Datenaufbereitung, sowie des Trainings und der Validierung durch Werkzeuge erfolgen, sodass systematische Fehler in die ML-Anwendung vermieden werden.

- Definition von Kriterien zur Qualifizierung der Datenpipeline und des Trainingsframworks.
- Definition von Kriterien zur Qualifizierung von Werkzeugen zur synthetischen Datengenerierung (Simulationsumgebungen)

Dokumentation erfolgt über: Erweiterte Arbeitsprodukte existierender Standards (SOTIF, ISO 26262) bzw. neu zu definierende Arbeitsprodukte im Rahmen von Referenzprozessen und Standards zur Verwendung von ML in sicherheitskritischen Anwendungen.

ML ist ein werkzeugintensiver Prozess und basiert auf komplexen Frameworks und hochautomatisierten Plattformen zur Datenbereitstellung, -aufbereitung und Prüfung.

Zusammenfassung und Handlungsbedarf

Absicherung der Betriebsphase

Automotive Quality Institute

Der Einsatz von ML-basierter Software bedarf eines kontinuierlichen Sicherheitsmanagements für die Betriebsphase, mit der sichergestellt wird, dass alle Prozesse für Beobachtung, Wartung und Pflege eines ML-basierten Systems kontinuierlich bereitgestellt werden.

- Technisches Monitoring (Distributional Shift Monitoring, Monitoring auf Sonderfälle und Fehlentscheidungen, ODD Monitoring, Übergabevorgänge)
- Rigides Change- und Updatemanagement über den gesamten Lebenszyklus eines Fahrzeugs.
- Realisierung eines effizienten, vertrauenswürdigen und sicheren Updatemechanismus für Fahrzeuge im Feld inklusive der Prüfung von Sicherheit- und Integrität der aktualisierten Fahrzeuge im Feld.

Dokumentation erfolgt über: Erweiterte Arbeitsprodukte existierender Standards (SOTIF, ISO 26262) bzw. neu zu definierende Arbeitsprodukte im Rahmen von Referenzprozessen und Standards zur Verwendung von ML in sicherheitskritischen Anwendungen.

ML-basierte Software muss kontinuierlich überwacht werden, um sicherheitskritische Abweichungen und Fehler auch in der Betriebsphase systematisch identifizieren und beseitigen zu können.