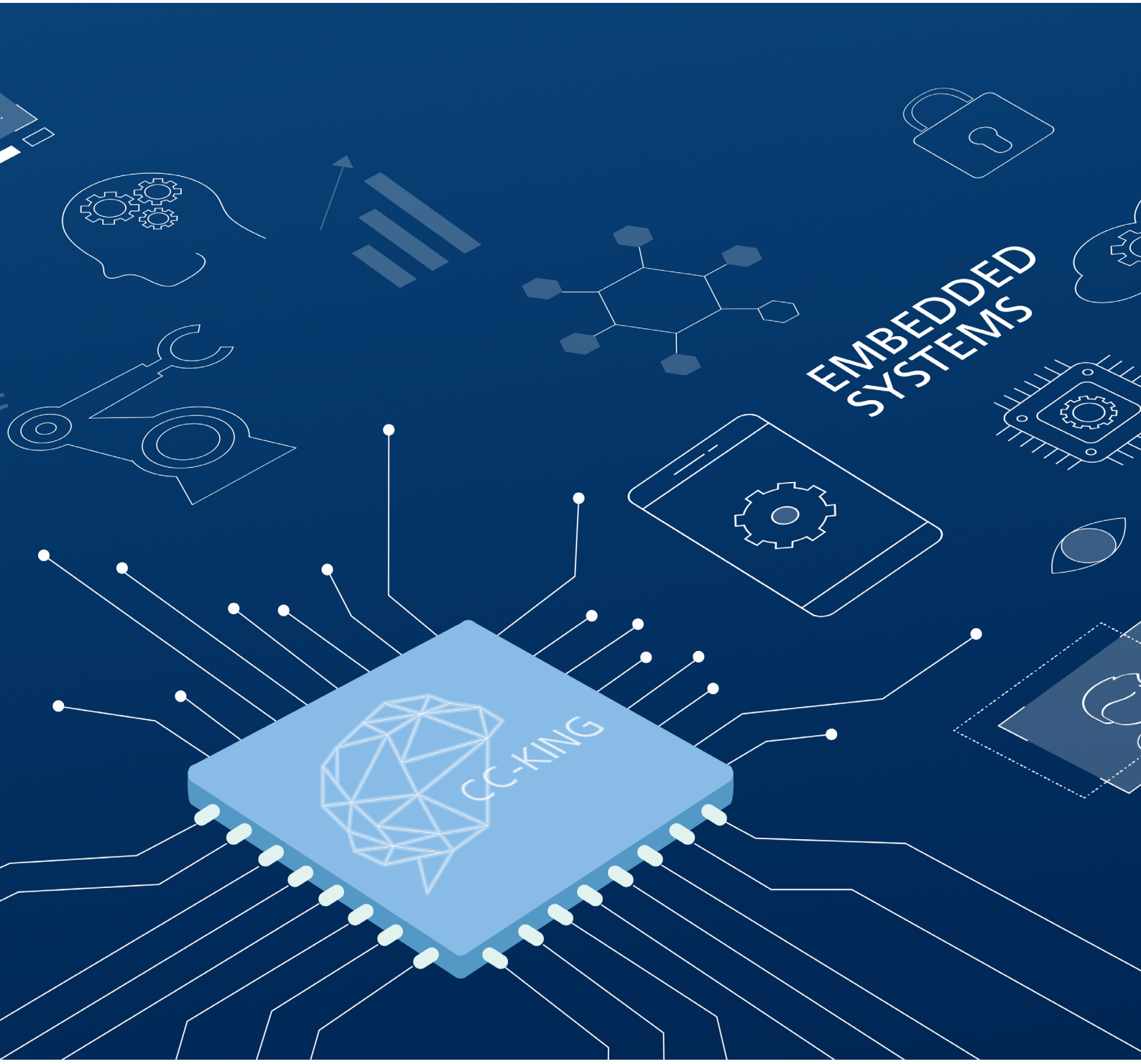
CC-KING
Kompetenzzentrum
KI-Engineering

PAISE®

Das Vorgehensmodell für KI-Engineering



in Kooperation mit:

FZI

KIT Karlsruher Institut für Technologie

Fraunhofer IOSB

www.ki-engineering.eu

Inhalt

CC-KING
Kompetenzzentrum
KI-Engineering

Einleitung

Das Process Model for AI Systems Engineering, kurz: PAISE®, ist ein Vorgehensmodell für KI-Engineering. Es wurde innerhalb des Kompetenzzentrums für KI-Engineering (CC-KING) entwickelt. PAISE® umfasst die systematische und standardisierte Entwicklung und den Betrieb von KI-basierten Systemlösungen. Vorgehensweisen aus der Informatik und datengetriebenen Modellbildung werden mit denen der klassischen Ingenieurdisziplinen, wie z. B. Systems Engineering, kombiniert.

Die Disziplin KI-Engineering zielt darauf ab, Methoden der ***Künstlichen Intelligenz (KI)***, im Englischen als „artificial intelligence" (AI) bezeichnet, aus ingenieurtechnischer Sicht systematisch, planbar und verlässlich in die Konzeption, Entwicklung und den Betrieb von technischen Systemen, bestehend aus Hardware und Software, integrieren zu können. Getrieben wird dieses Ziel von dem hohen Potenzial, das der Einsatz insbesondere von ***maschinellen Lernverfahren (ML)*** birgt. Im Bereich der Interpretation von Daten erreichen diese Verfahren heute mitunter Ergebnisgüten (Performanzen), die mit klassischen Verfahren trotz jahrzehntelanger Entwicklungsarbeit nicht erzielt werden konnten[1].

In der Vergangenheit basierte KI vornehmlich auf klassischen Algorithmen, wie z. B. wissens- oder regelbasierten Systemen. Diese Softwaresysteme wurden von Menschen programmiert und sind speziell auf einzelne Anwendungsfälle zugeschnitten. Durch den in den letzten Jahrzehnten zu verzeichnenden Fortschritt im Bereich der Computerrechenleistung und mit einer zunehmenden Verfügbarkeit von Daten wird der Einsatz von datengetriebenen Verfahren begünstigt und zunehmend ausgebaut. Daher rücken maschinelle Lernverfahren als Untergruppe von KI-Algorithmen vermehrt in den praktischen Vordergrund. Der ***ML-Algorithmus*** programmiert einfach gesagt eine Software zur Erfüllung einer gegebenen Aufgabe, indem er sogenannte Trainingsdaten analysiert und dort Muster und Zusammenhänge identifiziert. Die Funktionen der erstellten Software werden daher zum wesentlichen Teil durch die Zusammensetzung und die Qualität der Trainingsdaten bestimmt.

Dieser Ansatz bringt zwar einen immensen Effizienzgewinn in der Entwicklung mit sich, verlangt jedoch auch nach einem geeigneten Vorgehen während der Entwicklung. Klassische Vorgehensmodelle setzen ein manuell spezifiziertes und systematisch prüfbares System voraus und sind nur bedingt mit dem Einsatz maschineller Lernverfahren kompatibel. Für maschinelle Lernverfahren existieren derzeit nur empirische Testmethoden – ungeachtet dessen, ob ML im Endprodukt verbaut ist[2] oder ob ML-basierte Methoden genutzt werden, um ein klassisches Endprodukt zu entwickeln[3]. Sobald die Ergebnisse komplexer ML-basierter Verfahren die Tauglichkeit des Endprodukts in kritischer Weise beeinflussen, ist es erforderlich, diesen Einfluss auf Prinzipien der verwendeten ML-basierten Methoden und der zugrunde liegenden Daten zurückführen zu können und ihn im Rahmen der Systementwicklung fundiert mitzuverfolgen. Auch muss nach Auslieferung einer ML-basierten Komponente sicher gestellt werden, dass das Produkt etwa unter geänderten Umgebungsbedingungen weiterhin zuverlässig funktioniert.

Im Folgenden werden die Herausforderungen für die Projektleitung dargestellt, für die das Vorgehensmodell PAISE® Lösungen anbietet.

Hinweise zur Textformatierung

Zu ***fett und kursiv gedruckten Schlagwörtern*** finden Sie Begriffe aus dem Glossar (S. 32), **blaue Begriffe** entsprechen den Phasen von PAISE®

1 Verfahren des Deep Learning sind z.B. im Bereich der Bildinterpretation in der Lage, menschliche Probanden zu übertreffen [D. Ciresan, U. Meier, J. Schmidhuber. (2012). Multi-column Deep Neural Networks for Image Classification. IEEE Conference on Computer Vision and Pattern Recognition, S. 3642–3649.].

2 Z. B. wenn im Betrieb erlernte Entscheidungen getroffen werden oder wenn das Programm sogar weiter angepasst wird, also anhand der Daten im Betrieb weiter lernt.

3 Beispielsweise um mithilfe von ML-basierte Methoden geeignete Werkstoffe oder Auslegungsparameter zu wählen.

Herausforderungen bei der Entwicklung KI-basierter Systeme

Die Disziplin KI-Engineering[4] (engl. AI Systems Engineering) geht aus der Disziplin Systems Engineering hervor. Während heute erfolgreich Methoden und Techniken des Systems Engineering angewandt werden, um komplexe technische Systeme zu entwickeln, stellt die Verwendung von KI innerhalb solcher Systeme neue Herausforderungen an den Entwicklungsprozess.

Systems Engineering bietet formalisierte Methoden, um organisatorischen Herausforderungen, wie etwa eine interdisziplinäre Zusammenarbeit bei der Entwicklung von komplexen technischen Systemen, zu begegnen. Softwarelösungen, speziell Embedded Software, zum Beispiel auf einem Microcontroller oder in einer speicherprogrammierbaren Steuerung sind elementarer Bestandteil in vielen Anwendungsdomänen, wie zum Beispiel der Medizintechnik, dem Maschinen- und Anlagenbau oder der Mobilität.

Zur Entwicklung komplexer technischer Systeme werden heute unter anderem Vorgehensmodelle eingesetzt, die ursprünglich aus der Softwareentwicklung stammen und auf das Systems Engineering übertragen wurden. Beispiele sind das Wasserfallmodell[5], das V-Modell[6] und SCRUM[7].

Zwei spezielle Herausforderungen für die Entwicklung ***KI-basierter*** Systeme wurden identifiziert, auf die im Entwicklungsprozess eingegangen werden muss:

1. Die Performanz eines KI-basierten Ansatzes lässt sich häufig nicht im Voraus einschätzen, sondern muss empirisch ermittelt werden.
2. Datengetriebene Verfahren wie ***maschinelles Lernen*** benötigen bereits während der Entwicklung Daten aus dem Betrieb.

Im Folgenden werden diese beiden Aspekte detaillierter ausgeführt und die jeweilige Auswirkung auf den technischen Entwicklungsprozess (siehe ISO/IEC/IEEE 15288:2015) dargestellt. Das Ziel von PAISE® liegt in einer – aus technischer Sicht – qualitativ hochwertigen Lösung. Betriebswirtschaftliche Aspekte sowie unternehmensspezifische Prozesse werden in diesem Dokument nicht adressiert.

1. Die Performanz eines KI-basierten Ansatzes lässt sich häufig nicht a-priori einschätzen, sie muss empirisch ermittelt werden.

In einem Entwicklungsprozess nach dem Wasserfallmodell wird aus den Anforderungen eine High-Level-Architektur[8] abgeleitet und diese schrittweise verfeinert. In Rahmen einer Zertifizierung kritischer Systeme müssen diese Schritte der Ableitung und Verfeinerung zudem nachvollziehbar dokumentiert werden.

In vielen klassischen Ingenieurdisziplinen ist es möglich, eine High-Level-Architektur eines Systems anforderungsgemäß zu entwerfen, ohne funktionale Aspekte durch prototypische Umsetzungen testen zu müssen. Das wird durch Modellierungen auf Basis von physikalischen Modellen, Erfahrungswerten und Simulationen ermöglicht.[9]

---

4 https://www.ki-engineering.eu/de/was-ist-ki-engineering.html

5 [W. Royce. (1970). Managing the Development of Large Software Systems. Proceedings of IEEE WESCON 26 (August), (S. 1–9).]

6 [B.W. Boehm. (1981). Software Engineering Economics, Prentice Hall.]
[J. Friedrich, M. Kuhrmann, M. Sihling, U. Hammerschall. (2009). Das V-Modell XT. Informatik im Fokus. Springer, Berlin, Heidelberg.]

7 [K. Schwaber, M. Beedle. (2002). Agile Software Development with Scrum. Prentice Hall, Upper Saddle River, United States]

8 Die High-Level Architektur entspricht in PAISE® dem Systemmodell, welches in der Phase des Funktionale Dekomposition erstellt wird und während der Phase des Entwicklungszyklus verfeinert und angepasst wird.

Bei KI-Verfahren ist es jedoch schwieriger, die Performanz theoretisch oder anhand von Erfahrungswerten abzuschätzen da die High-Level-Architektur auf einer solchen Grundlage nur bedingt abschließend festgelegt werden kann. Gerade wenn nicht die mittlere Performanz, sondern selten auftretende Sonderfälle ausschlaggebend sind, können Details in der Umsetzung große Unterschiede im empirischen Verhalten bedeuten.

Es kommt somit häufig vor, dass bei einer engen Integration KI-basierter Verfahren in ein ***Gesamtsystem*** diese KI-basierten Verfahren vorab zumindest prototypisch umgesetzt und ihre Performanz hinsichtlich der Anforderungen empirisch geprüft sein müssen, bevor die High-Level-Systemarchitektur finalisiert werden kann. Diesem Aspekt kann durch ein iteratives Vorgehen Rechnung getragen werden, das in PAISE® in der Phase des **Entwicklungszyklus** verankert ist. Hier wird die High-Level-Architektur iterativ verfeinert und angepasst.

In manchen Fällen ist wegen bestimmter Rahmenbedingungen ein iteratives Verfeinern der High-Level-Architektur nicht möglich, etwa weil die Entwicklung einzelner ***Komponenten*** an Fremdfirmen vergeben werden soll. Hier können Vorentwicklungen von ***KI-basierten Komponenten*** zur Abschätzung der Performanz helfen, sodass die High-Level-Architektur verlässlich festgelegt werden kann und ein iteratives Anpassen wegfällt. Andererseits geht bei einem solchen Vorgehen Flexibilität verloren, was dazu führen kann, dass die aus technischer Sicht optimale Lösung von vornherein ausgeschlossen wird.

2. Datengetriebene Verfahren, wie maschinelles Lernen, benötigen oft bereits während der Entwicklung Daten aus dem Betrieb.

Datengetriebene Verfahren wie maschinelle Lernverfahren benötigen qualitativ hochwertige Daten, um aus diesen ihr Verhalten zu erlernen. Die Qualität wird unter anderem dadurch bestimmt, wie repräsentativ die zum Lernen verwendeten Daten (Trainingsdaten) für den vorgesehenen Anwendungsfall sind. Hierbei stellt sich die Herausforderung, dass die bei der Entwicklung benötigten Trainingsdaten möglichst bereits aus der Anwendung des Systems kommen sollten, das jedoch noch nicht fertig entwickelt ist.

Im ML4P-Vorgehensmodell (Machine Learning for Production[10]) der Fraunhofer-Gesellschaft wird von einer bestehenden Produktionsanlage ausgegangen, in die maschinelle Lernverfahren integriert werden sollen (Brownfield-Entwicklung). In diesem Fall ist es möglich, von der bestehenden Anlage qualitativ hochwertige Trainingsdaten zu bekommen. Wird ein KI-basiertes System hingegen komplett neu entwickelt (Greenfield-Entwicklung), so sind erst nach der Inbetriebnahme Realdaten aus dem System selbst vorhanden. Da aber schon die Entwicklung mithilfe dieser Daten stattfinden sollte, müssen andere Wege gegangen werden. Mögliche Varianten sind:

- *Gestaffelte Umsetzung:* Die Daten werden von einem bereits bestehenden technischen System erzeugt, in das KI integriert werden soll.
- *Messkampagnen:* Daten werden unter kontrollierten Bedingungen in dedizierten Messkampagnen und Experimentreihen, eventuell unter leicht anderen Bedingungen als im Anwendungsfall (Laborbedingungen), erzeugt.
- *Simulationen:* Daten werden durch Simulationen des technischen Systems erzeugt.
- *Externe Datenquellen:* Daten von externen Anbietern werden hinzugezogen.[11]

Datenquellen, sowohl für die Entwicklung als auch für den Betrieb, müssen somit während der Systementwicklung von Anfang an berücksichtigt werden. In PAISE® werden Datensätze im Systemmodell mit verortet, für deren Entwicklung Zeit und Ressourcen eingeplant werden müssen. Dieser Aspekt wird in PAISE® in einem eigenen Prozess, der „Datenbereitstellung", berücksichtigt.

---

9 Ein Beispiel für ein unterstützendes Werkzeug ist die Modellierungssprache Modelica, in der Multi-Physik Simulationen mit vordefinierten Bibliotheken wiederverwendbarer Bausteine einfach kombiniert werden können.

10 https://www.iosb.fraunhofer.de/de/projekte-produkte/ml4p-maschinelles-lernen-fuer-produktionsprozesse.html

11 Externe Datenquellen werden oft im Bereich der Bilderkennung verwendet. Hier sind zusätzliche Bilddaten entweder kostenlos verfügbar oder können von kommerziellen Anbietern erworben werden.

PAISE® – das Vorgehensmodell

PAISE® beleuchtet die Entwicklung eines Produkts als ein Gesamtsystem, das in Subsysteme zerlegbar ist. Charakteristisch in PAISE® ist das zyklische Durchlaufen von Verfeinerungsschritten, der Komponentenentwicklung und den Checkpoints. Dadurch wird ein Wechsel zwischen explorativem Vorgehen auf der einen Seite und zielgerichtetem Vorgehen auf der anderen Seite möglich.

Die primären Anwendungsdomänen für PAISE® sind Mobilität und Produktion, wobei unterschiedliche Nutzungsszenarien adressiert werden. Diese Szenarien umfassen sowohl die einmalige kundenspezifische Entwicklung und Implementierung von ***KI-basierten*** Systemen als auch die komplette Neuentwicklung von Produkten, die in mehrfacher Ausführung gefertigt und verkauft werden sollen.
Das Vorgehensmodell PAISE ist schematisch in Abbildung 1 dargestellt.

Die ***Subsysteme*** eines ***Gesamtsystems*** stellen unabhängig voneinander individuelle Funktionalitäten bereit, weisen klar definierte Schnittstellen auf und können selbst zerlegbar sein. ***Maschinelle Lernverfahren (ML)*** können zum einen direkt in Subsysteme integriert werden, zum anderen können sie bei der Entwicklung von ***Hilfssystemen*** (engl. Enabling Systems) eingesetzt werden. Subsysteme wie auch Hilfssysteme können klassifiziert sein als KI-basiert und/oder ***Datenquelle***. Zudem werden ***Datensätze*** individuell entwickelt. Subsysteme, Hilfssysteme und Datensätze werden in PAISE® als ***Komponenten*** bezeichnet.

Der Einsatz maschineller Lernverfahren birgt inhärente Risiken beim Entwicklungsprozess.[12] Solche Risiken ergeben sich z. B. aus der Abhängigkeit von der Datenqualität, die zu einer Einschränkung der Funktionalitäten führen kann oder der Nichtvorhersagbarkeit der Performanz von KI-basierten Systemen. Um diesen Risiken während des Entwicklungsprozesses Rechnung zu tragen, durchläuft der **Entwicklungszyklus** sogenannte ***Checkpoints***, an denen eine (Teil-)Integration und Bewertung hinsichtlich der Anforderungen stattfindet. Für ML-Komponenten bedeutet dies die Evaluierung anhand von Validierungsmetriken, um die Funktion innerhalb des Gesamtsystems zu bewerten. Die Ergebnisse eines Checkpoints können zu Verfeinerungen sowie Anpassungen in den verwendeten Lösungsansätzen führen, mit denen die Komponentenfunktionalität angestrebt wird. Durch zyklisches Durchlaufen von **Komponentenentwicklung**, **Checkpoint/Bewertung** und **Verfeinerung** wird der Reifegrad aller Komponenten und damit des Gesamtsystems kontinuierlich erhöht.

Bei einer parallelisierten **Komponentenentwicklung** ist eine hierarchische Kombination unterschiedlicher Vorgehensmodelle möglich. Während bei der Entwicklung klassischer Systemkomponenten präferierte Vorgehensweisen verfolgt werden können, wird für ML-Komponenten die „ML-Komponentenentwicklung" definiert, die die Entwicklungsphasen in einer ingenieurmäßigen und standardisierten Form verknüpft. Da die Bereitstellung der Trainingsdaten, die für die Entwicklung der ML-Komponente nötig sind, mitunter sehr komplex sein kann, betrachtet PAISE® den Vorgang der **Datenbereitstellung** (S. 19-21) gesondert.

Für das Vorgehen bei der Gesamtsystementwicklung wurde in dieser Darstellung ein Wasserfallmodell gewählt. Es sei jedoch hervorgehoben, dass sich die beschriebenen sieben Phasen auch in anderen Modellen anordnen lassen. Das grundlegende „checkpoint-basierte" Konzept von PAISE® ist dabei weiterhin anwendbar.

PAISE® ist eine Vorgehensvorlage, die je nach den organisatorischen Rahmenbedingungen im Unternehmen angepasst werden kann und sollte. Wie bei jedem Vorgehensmodell müssen die Elemente von PAISE® auf den jeweiligen Anwendungsfall übertragen werden, um konkrete Handlungen daraus abzuleiten. Der Fokus von PAISE® liegt auf dem technischen Prozess (siehe ISO/IEC/EEE 15288:2015) und lässt betriebswirtschaftliche Aspekte und unternehmensspezifische Prozesse außen vor.

12 Hier sind Risiken gemeint, die das Projekt bedrohen und die gemäß des Spiralmodells von Boehm durch iterative Ansätze minimiert werden können. [B. Boehm. (19 86). A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 11 (4): 14–24.]

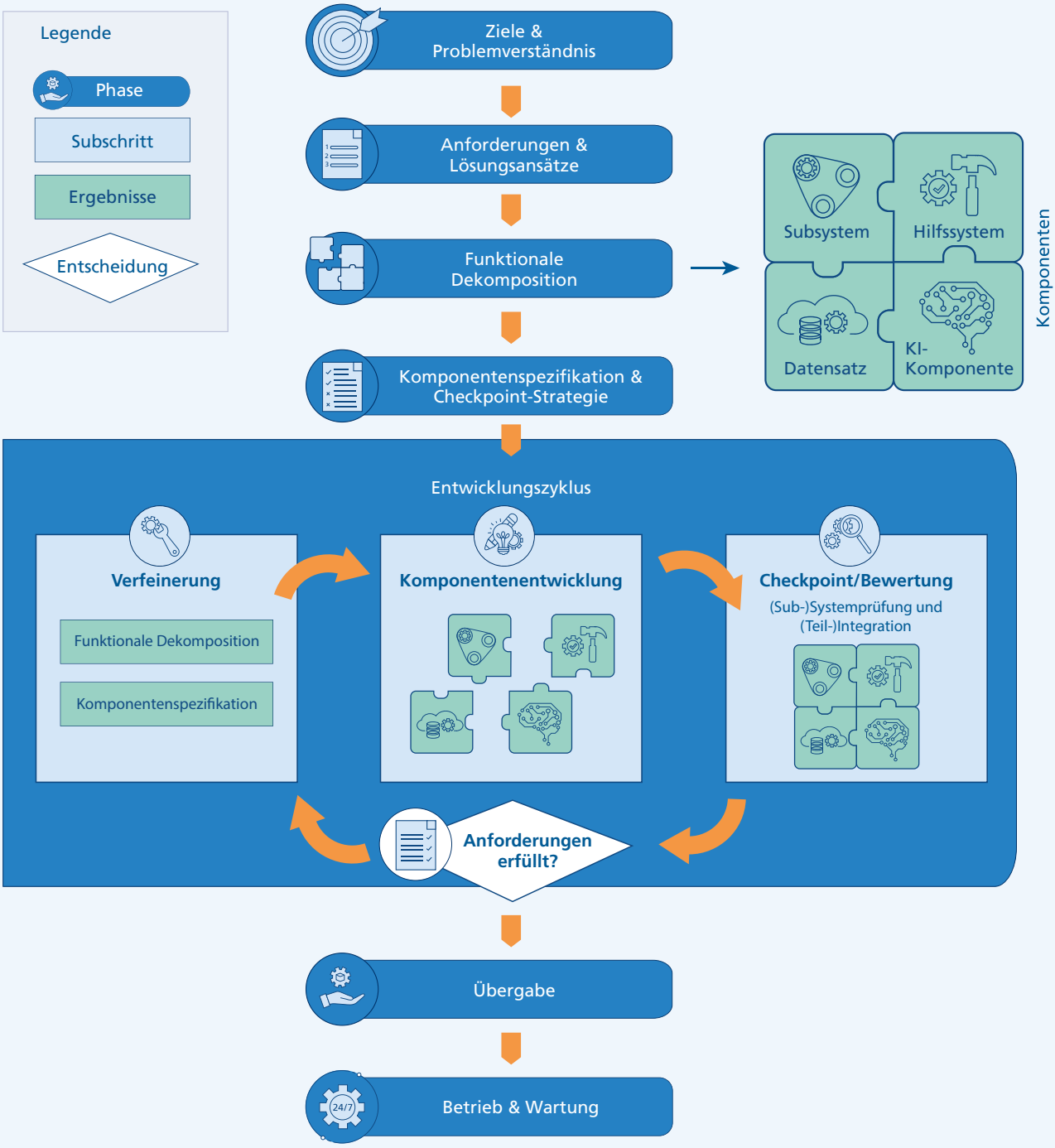


Abbildung 1: Schematische Darstellung des Vorgehensmodells PAISE®

Durchgehende Artefakte

PAISE® weist vier durchgehende Artefakte auf. Die Artefakte werden in bestimmten Phasen initialisiert und im Verlauf der Entwicklung kontinuierlich erweitert und angepasst.

Systemmodell

Das Systemmodell beschreibt die Abhängigkeiten der ***Komponenten*** (also ***Subsysteme***, ***Hilfssysteme*** und ***Datensätze***) untereinander sowie deren Schnittstellen.[13] Es orientiert sich an dem 1979 von Ropohl im Rahmen der Systemtheorie der Technik veröffentlichten Modell für technische Systeme (Ropohl, 2009). Das Systemmodell wird in der Phase **Funktionale Dekomposition** initial erstellt und definiert die Komponenten, die mithilfe der einzelnen Fachdisziplinen im **Entwicklungszyklus** entwickelt werden. Ein Beispiel für ein Systemmodell findet sich auf Seite 13.

Rollenverteilung

Die Rollenverteilung definiert, welche Verantwortlichkeiten in welcher Phase benötigt werden. Dieses Artefakt wird in **Anforderungen & Problemverständnis** initialisiert und läuft in allen weiteren Phasen des Vorgehensmodells mit. Eine nähere Darstellung der Aspekte der Rollenverteilung findet sich auf Seite 28-29.

Dokumentation für externe Prüfungen

Diese Art der Dokumentation erfasst die Eigenschaften, die das ***Gesamtsystem*** oder einzelne Komponenten zwingend für die Prüfung und Abnahme durch externe Parteien (z.B. Behörden) erfüllen müssen, sowie Indikatoren für deren Nachweis beziehungsweise Argumentation, die im Rahmen der Entwicklung und Erprobung akkumuliert werden. Typische Beispiele entsprechender Garantien sind die funktionale Sicherheit („Safety“) sowie die IT-Sicherheit („Security“), die auch Aspekte des Datenschutzes durch das System mit einschließen kann. Ferner kann die Dokumentation Aspekte wie beispielsweise Erklärbarkeit, Steuerbarkeit oder eine gerichtliche Verwertbarkeit umfassen.

Datendokumentation

Die Datendokumentation ist eine Beschreibung der Daten, die für die Entwicklung und Erprobung der ***KI-basierten*** Komponenten verwendet wurden und die deren Funktion maßgeblich bestimmen. Dokumentation wird während des **Entwicklungszyklus** erstellt sowie während **Betrieb & Wartung** stetig erweitert und angepasst. Verwendete Daten sind einerseits in Bezug auf ihre Quelle zu kategorisieren (bspw. verwendete öffentliche Datensätze, Erhebungs- und Annotationsverfahren, Messverfahren, Umgebungsbedingungen), und andererseits im Bezug auf ihre Qualität (technische Fehler, Unsicherheiten etc.), Umfang und ihrer Vorverarbeitung (Schätzen von fehlenden Werten, Anreicherung bei deutlich unterrepräsentierten Populationen) darzustellen. Entsprechende Qualitäten werden unter anderem im Vorschlag der Europäischen Kommission für gesetzliche Regelungen bezüglich KI gefordert.[14]

Durch das Archivieren sämtlicher genutzter Daten können Anforderungen wie die der Europäischen Kommission teilweise bedient werden. Dies ist jedoch nicht in jedem Fall sinnvoll, z. B. bei besonders umfangreichen Datenmengen oder im Fall von online lernenden Systemen. Hierbei können dann Methoden zur Reduktion von Datensätzen, bspw. auf Stichproben oder Metadaten, oder zur Versionierung von Änderungen, bspw. über Hashwerte, angewandt werden.

13 Diese Betrachtung findet sich in zahlreichen Ansätzen zur aktuellen Behandlung von KI- bzw. ML-basierten Systemen wieder. Beispielsweise geht man im Mobilitätsbereich im Zuge der ISO 21448 „Road vehicles — Safety of the intended functionality“ von klassischen Fehlern einzelner Komponenten über zu komplexen Risiken des Gesamtsystems.

14 [European Commission. (2021). Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative ActBrüssel: COM/2021/206 final.]

Ziele & Problemverständnis

In der ersten Phase von PAISE® werden die Ziele definiert, die mit dem zu entwickelnden Produkt erreicht werden sollen. Zudem wird ein Problemverständnis etabliert, das in den folgenden Phasen weiter ausgebaut und geschärft wird.

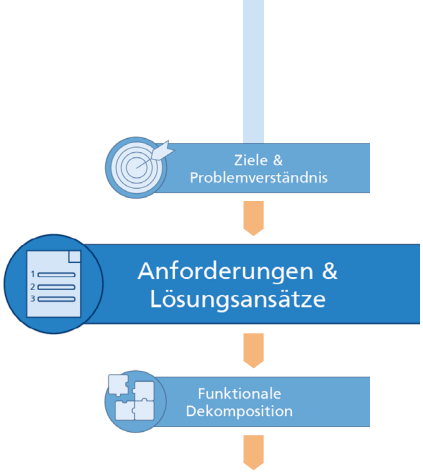
In dieser Phase ist es besonders bei hoher organisatorischer Komplexität wichtig, darauf zu achten, dass das Problemverständnis zwischen allen beteiligten Teams und Organisationen übereinstimmt. Bei den Zielen können auch Geschäftsmodelle einfließen, die verfolgt werden sollen, wie etwa ob ein Produkt entwickelt und anschließend serienmäßig vertrieben werden soll oder ob eine bestimmte Serviceleistung umgesetzt werden soll.

Beispiel[15]***:***

Die Entwicklung eines kamerabasierten Notbremssystems für PKW wird in Auftrag gegeben. Das System soll bei Autobahnfahrten aus einer einzelnen Frontkamera vorausfahrende Fahrzeuge erkennen, ihren Abstand und ihre Relativgeschwindigkeit schätzen, und bei drohenden Auffahrunfällen eine Notbremsung auslösen können. In der ersten Phase wird herausgearbeitet, dass das System Auffahrunfälle mit hoher Zuverlässigkeit vermeiden soll, dass die gesamte Verarbeitungskette von Kameraauswahl bis zur Realisierung der elektronischen Bremsenansteuerung zu spezifizieren ist, und dass bislang im auftraggebenden Unternehmen weder KI-Verfahren noch Datensätze bestehen, auf denen die Entwicklung aufsetzen soll.

Leitfragen	**Ergebnisse**
• Welches Problem soll gelöst werden? • Was soll an Kundinnen und Kunden verkauft werden? • Was ist der Ausgangszustand? • Was kennzeichnet den gewünschten Endzustand? • Welche Daten sollen verwertet werden? • Befinden sich schon KI-Methoden im Einsatz? • Können schon vorhandene ***ML-Modelle*** im Zuge von Transfer Learning genutzt werden?	• Dokumentation der Antworten bzw. Betrachtungsergebnisse der Leitfragen im Sinne einer Darstellung von Problemstellung und Zielen, z.B. in einem Projektsteckbrief

15 Das Beispiel eines Automotive-Notbremssystems wurde hier insbesondere ausgewählt, um die Nachvollziehbarkeit zu verbessern, da in diesem Bereich weitverbreitete Alltagserfahrungen bestehen.



Anforderungen & Lösungsansatz

In dieser Phase werden die Anforderungen an das Gesamtsystem analysiert und mögliche Lösungsansätze zur Umsetzung definiert.

Aus den Anforderungen an das Produkt werden hier erstmals Ideen zu möglichen Lösungsansätzen abgeleitet. Dies findet noch auf der High-Level-Ebene statt. In dieser Phase können sich auch mehrere Lösungsansätze ergeben, die hinsichtlich ihrer Realisierbarkeit bewertet werden. Im weiteren Entwicklungsprozess wird dann zunächst an dem Ansatz gearbeitet, der am realistischsten erscheint, und dieser weiter verfeinert.

Wenn die Entscheidung getroffen wird, ***KI-basierte*** Lösungsansätze zu verwenden, müssen in dieser Phase in Zukunft auch Anforderungen aus gesetzlichen Regulierungen berücksichtigt werden.[16] Beispiele für solche zusätzlichen Anforderungen sind Dokumentationspflichten, wie z.B. konzeptionelle Entscheidungen bezüglich Verfahren zur Sicherstellung der Datenhoheit, Datenverwaltung, Datenerfassung- und Datenaufbereitung oder die Einführung eines Risikomanagementsystems.

Beispiel:

Im Rahmen dieser Phase wird erarbeitet, dass das System so zuverlässig arbeiten soll, dass es kritische Situationen ohne das Eingreifen von Fahrerin oder Fahrer zu 99 Prozent sicher auflösen kann. Schäden können sowohl durch falsch-negative Auslösung eintreten (keine Vermeidung des Auffahrunfalls) sowie durch falsch-positive Auslösung (Unfälle durch unnötiges Notbremsen) entstehen.

Das System soll nicht während der Fahrt weiterlernen, sondern nur im Bedarfsfall durch Hersteller-Updates bei jährlichen Wartungsterminen aktualisiert werden. Das System verfügt über eine hohe Autonomie über die Längsführung des Fahrzeugs, um Fahrerin oder Fahrer bei mangelnder Aufmerksamkeit zu unterstützen und es verlässt sich nicht darauf, dass Fahrende selbst rechtzeitig bremsen. Umgekehrt können Fehlauslösungen in der Regel durch die Fahrenden nicht mehr rechtzeitig abgebrochen werden. Die reguläre Fahraufgabe (Distanzregelung, Querführung) hingegen übernimmt das System nicht. Es besteht die Einschätzung dass KI für die Objekterkennung und die Distanzschätzung genutzt werden kann, aber auch, dass für die Zulassung eine ausführliche Nachvollziehbarkeit der Ergebnisgüte gegeben sein muss.

16 Siehe Vorschlag der Europäischen Kommission für gesetzliche Regelungen bezüglich KI: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206

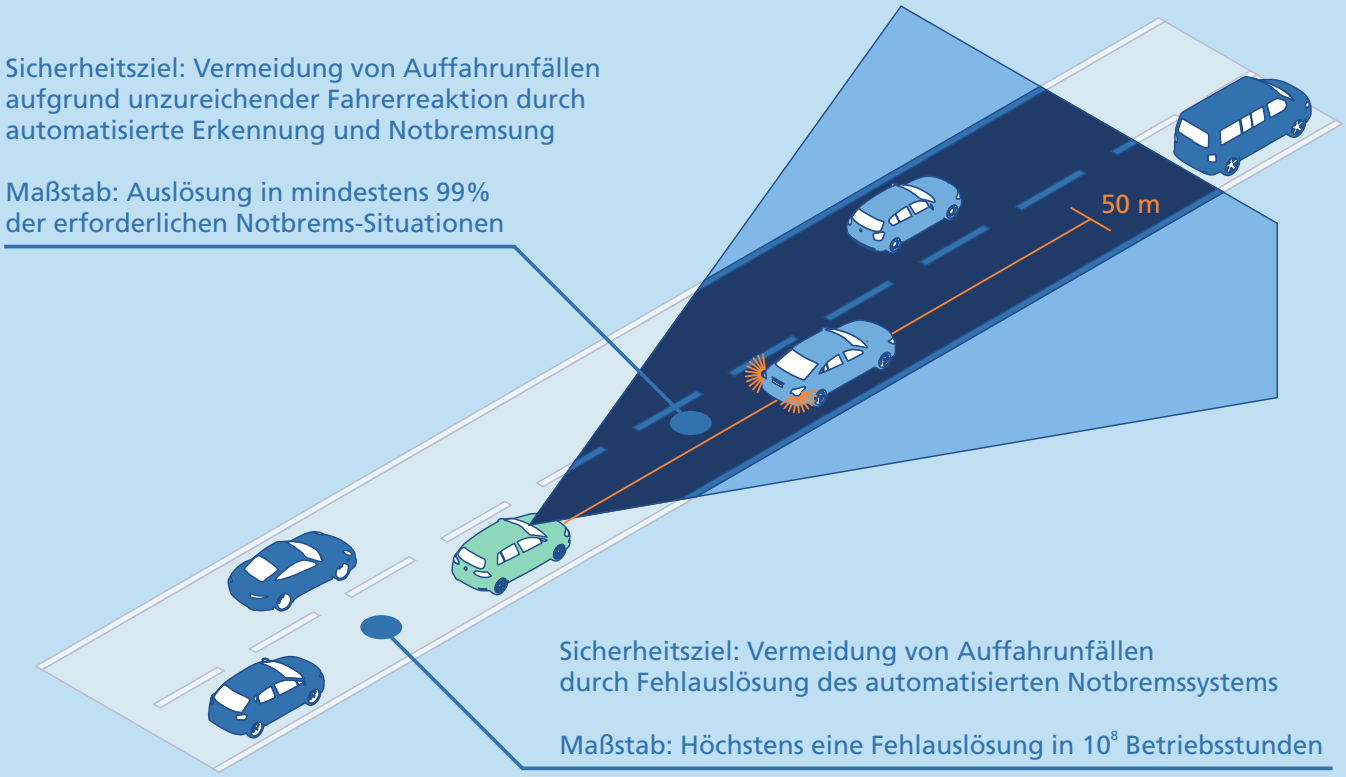
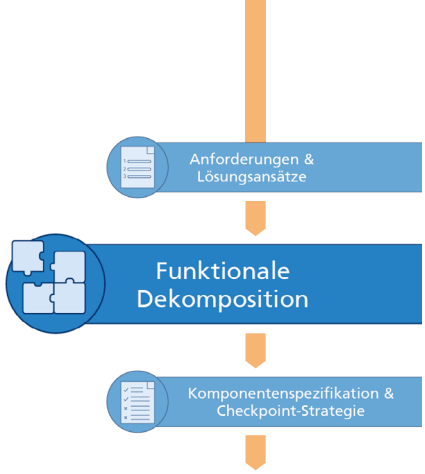


Abbildung 2: Visualisierung der Hauptanforderungen an das im Beispiel eingeführte Notbremssystem.

Leitfragen	Ergebnisse
• Welche Risiken für Schäden an Mensch und Umgebung müssen berücksichtigt werden? • Auf Basis welcher Daten soll das System Erkenntnisse liefern oder Entscheidungen treffen? • Wer soll die Rechte an den Daten haben? • Zu welchem Zeitpunkt soll aus Daten gelernt werden? • Soll die KI aktiv Prozesse steuern können? • Für welche Anforderungen kann KI ein Lösungsansatz sein? • Ist ein gesteigertes Maß an Nachvollziehbarkeit (Erklärbarkeit) der KI-Anwendung gewünscht/nötig?	• Priorisierte Systemanforderungen • Anforderungen an den Entwicklungsprozess • Auswahl des am realistischsten erscheinenden Lösungsansatz für das Gesamtsystemätze • Feststellung des Nutzens und Schwerpunkts der KI im Lösungsansatz und dem Projekt • Erste Risikobewertung des Systems • Einordnung des Systems nach Autonomiegrad[17] • Vertragliche Regelungen für Datennutzung

17 Anwendungsfälle des (teil-)autonomen Fahrens lassen sich in fünf Stufen des Standards SAE J3016 der Organisation SAE-International einteilen. Für Anwendungsfälle aus der Produktion kann ebenfalls eine Einordnung in fünf Stufen vorgenommen werden, wie von der Plattform Industrie 4.0 vorgeschlagen [Plattform Industrie 4.0. (2019). Technologieszenario „Künstliche Intelligenz in der Industrie 4.0", Working Paper].



Funktionale Dekomposition

In der dritten Phase werden die Funktionen des Gesamtsystems auf Subsysteme initial heruntergebrochen, mit dem Ergebnis einer meist hierarchischen Subsystemspezifikation mit wohldefinierten Schnittstellen. Ergänzt wird diese durch die Spezifikation zusätzlich benötigter Hilfssysteme.

Die in den Anforderungen an das ***Gesamtsystem*** definierten Funktionen werden auf ***Subsysteme*** verteilt. Die Granularität der Unterteilung hängt hierbei stark von der Komplexität des Systems ab. Einhergehend mit der Zerlegung erfolgt die Definition wohldefinierter Schnittstellen.

Neben der Betrachtung des Gesamtsystems werden in dieser Phase ***Hilfssysteme*** (engl. Enabling Systems) spezifiziert, die nicht Teil des ausgelieferten Systems sind, jedoch aber für dessen Entwicklung notwendig sind.

Wichtig ist, hervorzuheben, dass Relationen zwischen ***Komponenten***, also zwischen Hilfs- und Subsystemen, sehr unterschiedlicher Natur sein können. Hier können Energieflüsse (z. B. Strom, Hydraulik, Druckluft), Informationsflüsse (z. B. Daten), Kraftflüsse und Materialflüsse wirken.

Abbildung 3 zeigt ein Beispiel für eine solche Zerlegung. Hierbei ist hervorzuheben dass Komponenten auch als Datenquellen agieren können. Als Datenquellen werden Sub- oder Hilfssysteme verstanden, die Daten für die Entwicklung und/oder für den Betrieb liefern und somit maßgeblich die Funktionalität der KI-Komponenten beeinflussen.

Die Entscheidung, welches Subsystem ***KI-basiert*** ist kann entweder mithilfe von Erfahrungen während der initialen funktionalen Dekomposition getroffen werden oder während des **Entwicklungszyklus**, in dem die Verfeinerung der initialen Dekomposition vorgesehen ist. Wichtig ist, zu beachten, dass die Entscheidung, KI zu verwenden, Teil des Lösungsansatzes und nicht Teil der Anforderungen ist. Zusätzlich kann sich während der Entwicklung die Notwendigkeit ergeben, dass weitere Hilfssysteme oder Subsysteme hinzugezogen werden müssen. Ebenso können Komponenten wegfallen, falls diese nachweislich nicht mehr benötigt werden. Es ist daher wichtig zu erwähnen, dass die erste funktionale Dekomposition nicht final ist. Sie wird für die ersten Iterationen des Vorgehens genutzt und sollte anschließend angepasst werden. Die ***funktionale Dekomposition*** unterstützt die genaue Zweckbestimmung von sowohl klassischen Komponenten, wie mechanischen und elektrischen Systemen, als auch von KI-basierten Komponenten.

Beispiel:

Die Dekomposition ergibt das in Abbildung 3 gezeigte Ergebnis. Im Detektor werden in den von der Kamera aufgenommenen Bildern Objekte identifiziert. Im Entscheider wird eine Einschätzung der Distanz- und Relativgeschwindigkeit der Objekte vorgenommen und darauf basierend eine Entscheidung zur Notbremsung getroffen. Fällt diese Entscheidung entsprechend aus, wird ein Auslösesignal an die Bremssteuerung gesendet, die die hydraulisch-mechanische Bremsauslösung überwacht. Die Bremse ist nicht mehr Teil des ausgelieferten Gesamtsystems, sondern wird über eine vordefinierte Schnittstelle mit der Bremssteuerung verbunden. Der Detektor und der Entscheider sind KI-basierte Subsysteme. Die Montageposition der Kamera am Auto soll über ein KI-basiertes Hilfssystem optimiert werden. Weiterhin sollen Daten von der Kamera, sobald diese verfügbar ist, in einer internen Datenbank gespeichert werden, um Trainingsdaten für die Entwicklung des Detektors verfügbar zu machen. Da jedoch die Kamera nicht von Anfang an zur Verfügung steht, wird der Detektor zunächst auf Basis von synthetisierten Kamerabildern und mit einem extern verfügbaren Datensatz von Verkehrssituationen namens „Cityscapes“ entwickelt. Der Cityscapes-Datensatz wird ebenso bei der Entwicklung des Entscheiders eingesetzt. Im dargestellten Beispiel sind „Kamera“, „Interne Datenbank“, „Synthese von Kamerabildern“ und „Datenbank mit Cityscapes-Datensatz“ Datenquellen.

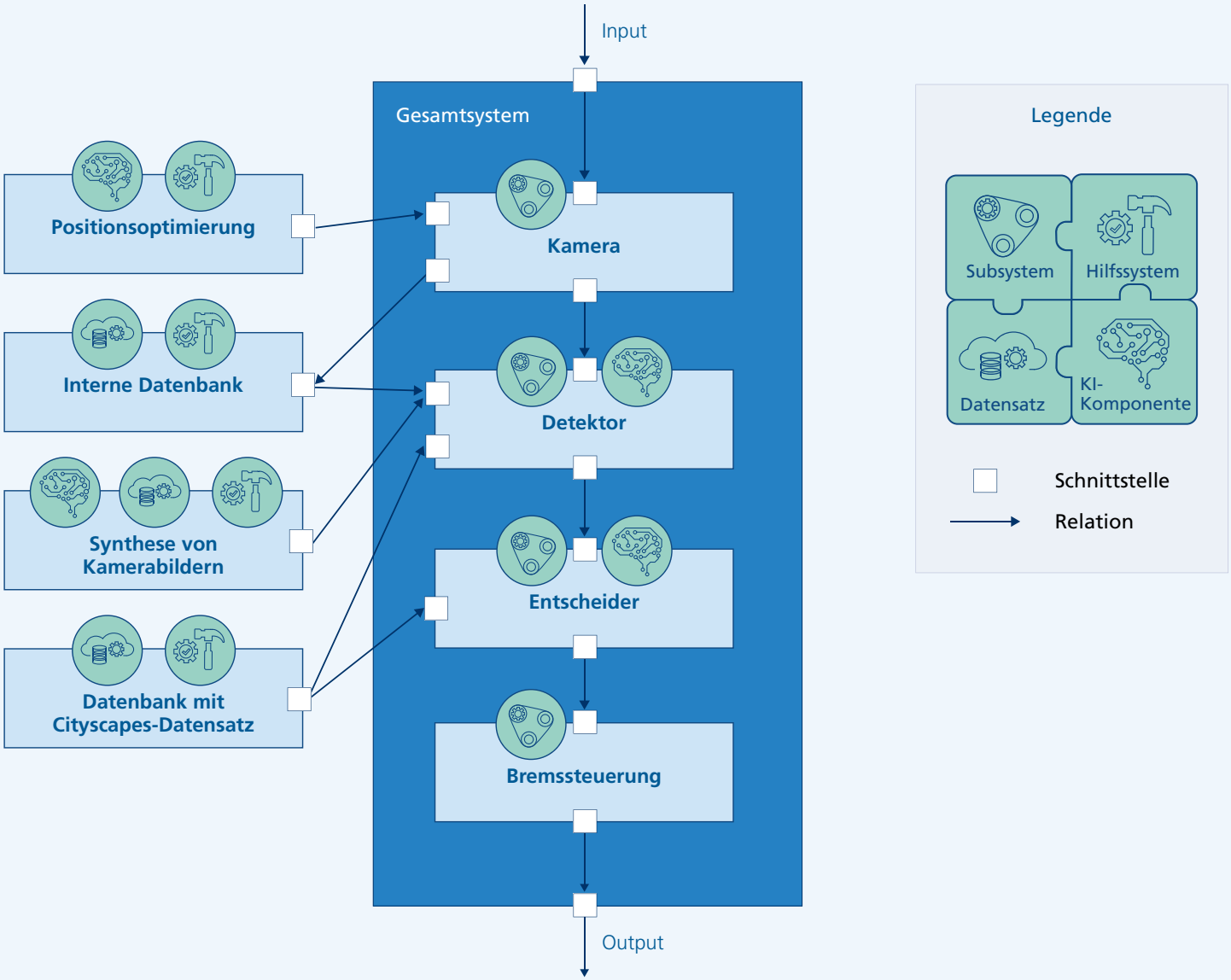
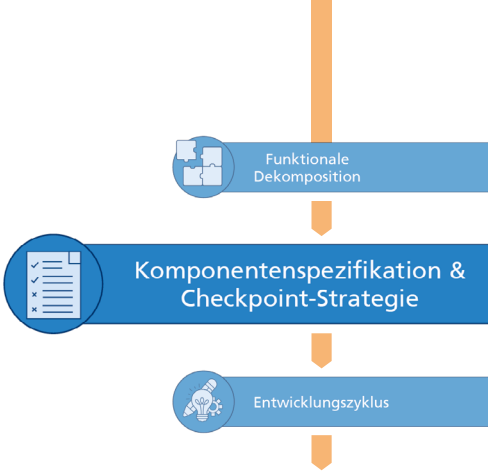


Abbildung 3: Schematische Darstellung des Systemmodells für das Beispiel eines Notbremssystems.

Leitfragen	Ergebnisse
• Welche Subsysteme übernehmen welche Funktionen? • Welche Hilfssysteme werden für die Entwicklung der Subsysteme benötigt? • Welche Funktion kann voraussichtlich von einer KI-Komponente übernommen werden? • Welche Datenquellen gibt es für das Training, das Testen und den Betrieb der KI-Komponenten?	• (Hierarchische) System- und Hilfssystemspezifikation • Verortung der Datenquellen und Datensätze für Training, Test und Laufzeit der KI-Sub- bzw. KI-Hilfssysteme • Definition der Schnittstellen zwischen Subsystemen und Hilfssystemen • Schematische Darstellung der Entwicklungsumgebung



Komponentenspezifikation & Checkpoint-Strategie

In dieser Phase wird eine erste Version der Komponentenspezifikation erstellt sowie eine Strategie für die Checkpoints der parallelen Subsystementwicklung festgelegt.

Die Komponentenspezifikation leitet sich in erster Instanz aus den Anforderungen an das ***Gesamtsystem*** und aus dem Systemmodell ab. Es werden konkrete Anforderungen an die ***Subsysteme*** erstellt und mögliche komponentenspezifische Lösungsansätze erdacht. Diese Lösungsansätze werden während der Phase des **Entwicklungszyklus** überprüft und weiterentwickelt werden.

Checkpoints dienen zur Synchronisation des Entwicklungsstands aller ***Komponenten*** sowie zum Testen des Zusammenspiels der Subsysteme innerhalb des Gesamtsystems. Zu diesem Zweck findet hier eine (Teil-)Integration der Subsysteme statt, mit entsprechenden Verifizierungs- und Validierungstests bezüglich der Anforderungen an das Gesamtsystem. Da die Entwicklung jedes Subsystems ein eigenes Vorgehen erlaubt, ist ein unterschiedlich schneller Fortschritt je Komponente zu erwarten. Nicht alle Komponenten müssen aktiv an jedem Checkpoint teilnehmen.

Bei der Festlegung, wann und nach welchen Kriterien ein Checkpoint stattfindet, können unterschiedliche Strategien verfolgt werden. Obwohl auch eine klassische Meilensteinplanung möglich ist, die viele Vorteile hinsichtlich der Planbarkeit birgt, wird ein agileres Konzept empfohlen. Der Unterschied von Checkpoints zu formalen Meilensteinen ist, dass nicht zwingend ein definierter Entwicklungsstand vorgegeben werden muss. Hinsichtlich der Checkpoint-Planung sind folgende Strategien möglich:

- *Die featurebasierte Strategie:* Ein bestimmtes Feature oder eine bestimmte Anforderung sollen bis zum nächsten Checkpoint umgesetzt werden. Diese Strategie orientiert sich an der Umsetzung der sogenannten „User Story" im agilen Vorgehensmodell SCRUM. Wichtig ist, dass während der Entwicklung nur Designentscheidungen getroffen werden, die zur Umsetzung des Features und zur Erreichung eines Minimalproduktes (Minimum Viable Product) notwendig sind. So werden schlussendlich Kosten bei Umentscheidungen gespart.

- *Die reifegradbasierte Strategie:* Ein Checkpoint ist immer dann erreicht, wenn mindestens zwei Subsysteme einen bestimmten Reifegrad erworben haben. Reifegrade können hier z. B. sein:
  - Vorläufige Analyse (Proof of Concept)
  - Sicherstellen von Grundfunktionalitäten
  - Erfüllung der Leistungsmetriken (KPI)
  - Leistungssteigerung/-optimierung
  - Optimierung der Nutzerfreundlichkeit

  Je nach Anwendungsfall können die Reifegrade weiter spezifiziert und unterteilt werden. Sie stellen somit Zwischenziele auf dem Weg zum fertigen Produkt dar.

- *Die zeitbasierte Strategie:* Ein Checkpoint ist zu einem regelmäßigen Zeitpunkt erreicht, z. B. immer nach einer Woche. Hier besteht die Herausforderung, dass die Arbeitspakete bis zum nächsten Checkpoint so gewählt werden, dass dann eine (Teil-)Integration mit allen Neuerungen möglich ist. Da jedoch nicht alle Komponenten zusammen an einem Checkpoint teilnehmen müssen, sind auch hier verschiedene Zeitintervalle je Komponente möglich.

Sowohl bei der Verfolgung einer Strategie als auch bei der Kombination verschiedener Strategien ist es wichtig, dass im Verfeinerungsschritt der jeweils nächste Checkpoint klar definiert wird. Dazu gehören die Fragen, welche Subsysteme an der (Teil-)Integration teilnehmen und wann ein Checkpoint erreicht ist.

Beispiel:

Folgende Tabelle stellt die Aspekte der reifegrad- und feature-basierten Strategie für die Checkpoints gegenüber.

Feature-basierte Strategie	Reifegrad-basierte Strategie
Im Folgenden sind beispielhaft 2-3 Features für jede Komponente dargestellt, die innerhalb der ersten Entwicklungszyklen umgesetzt werden sollen. Ein Checkpoint findet statt, sobald eine Integration einer der weiterentwickelten Komponenten möglich ist.	Im Folgenden sind exemplarisch ein paar Reifegrade des Gesamtsystems in Stichpunkten charakterisiert.
Kamera: • Erarbeitung geeigneter Spezifikationen (Auflösung, Bildfrequenz, etc.) • Beschaffung Prototyp • Bestimmung optimaler Einbau-Position	Proof of Concept: • Versuchsfahrzeug mit Kamera-Prototyp • Detektor und Entscheider wurden auf Cityscapes-Datensatz entwickelt • Bremseingriff über Serien-AEB-Schnittstelle • Testfahrten auf dem Versuchsplatz
Datensätze: • Auswahl Datensatz (Cityscapes) • Schnittstellen-Anbindung zum Detektor • Erhöhung Übereinstimmung mit Zielanwendung durch Synthetische Erzeugung neuer Kamerabilder	Umsetzung 70% der Anforderungen: • Umzug auf Kamera-Zielsystem • Noch kein embedded Rechner im Fahrzeug sondern dedizierte Recheneinheit auf dem Beifahrersitz • Einsatz des realen Zielfahrzeugs im Entwicklungsstadium • Testfahrten auf dem Versuchsplatz
Detektor: • Abgleich mit Kameraspezifikation • Schnittstellenanbindung an Entscheider • Verifizierung mit realen Kameradaten	Umsetzung 90% der Anforderungen: • Hardware- und Software in Zielarchitektur vorliegend • Performanz des Notbremssystems noch unbekannt • Tests im Realverkehr mit Sicherheitsfahrer
Entscheider: • Schnittstellenanbindung an Detektor • Verifizierung auf reale Daten des Detektors	Umsetzung 100% der Anforderungen • Fertig entwickeltes Gesamtsystem mit bekannter und den Anforderungen entsprechender Performanz
Bremssteuerung: • Erarbeitung geeigneter Spezifikation • Verifizierung der Bremsumsetzung auf dem Prüfstand	

Bei der Behandlung des Beispiels in nachfolgenden Phasen wird die feature-basierte Strategie gewählt. Zu jedem Verfeinerungsschritt werden Ziele und Features für jede Komponente erarbeitet, die bis zum nächsten Checkpoint umgesetzt werden sollen. Beispiele für die initialen Komponentenspezifikationen für die Komponenten „Datenbank mit Cityscapes-Datensatz" und „Detektor" sind jeweils auf den Seiten 20 und 24 im Zusammenhang mit den entsprechenden Vorgehensweisen während der Entwicklung ausgeführt.

Leitfragen	Ergebnisse
• Welche komponentenspezifischen Lösungsansätze sollen verfolgt werden? • Welche Informationen stehen dem KI-Subsystem als Eingabe zur Verfügung (Feature-Vektor)? • Welche Güte muss das KI-Subsystem erreichen und wie erfolgt der Nachweis? • Welche Datengüte muss zu welchem Checkpoint vorliegen, damit die KI-Subsystem-Entwicklung voranschreiten kann? Wie erfolgt der Nachweis der Güte?	• Dokumentation aller initialen Spezifikationen • Dokumentation der Strategie für die Checkpoints

Entwicklungszyklus

Die Komponentenentwicklung findet in iterativen Zyklen statt, die den Reifegrad des Gesamtsystems kontinuierlich steigern. Die Zyklen bestehen aus einem **Verfeinerungs**schritt, einer Phase der parallelisierten **Komponentenentwicklung** und einem **Checkpoint** inklusive der Bewertung der Fortschritte. Die Ergebnisse der Bewertung führen abschließend zu der Entscheidung, ob das Gesamtsystem gemäß den Anforderungen fertig gestellt ist.

Die **Verfeinerung** findet auf Basis der Ergebnisse von **Checkpoint/Bewertung** statt. Der Lösungsansatz, der zur Realisierung der jeweiligen Komponentenspezifikation gewählt wurde, wird detaillierter ausgearbeitet oder falls nötig, variiert. Eine Variation der komponentenspezifischen Lösungsansätze ist in den ersten Zyklen des **Entwicklungszyklus** sinnvoll, um keine Lösungen von vornherein auszuschließen. In späteren Zyklen sollte nur noch an der detaillierten Ausgestaltung der Lösungsansätze gearbeitet werden, um den Reifegrad des Produkts kontinuierlich zu erhöhen. Der Schritt der **Verfeinerung** findet interdisziplinär statt, um die Abhängigkeiten zwischen den ***Komponenten*** zu berücksichtigen. Anschließend werden entsprechende Anpassungen bezüglich des Systemmodells sowie der Komponentenspezifikation vorgenommen. Hierbei kann es insbesondere zu einer weiteren Dekomposition von Komponenten kommen, etwa um Engpässe bei der Entwicklung zu vermeiden, oder es können zusätzliche ***Komponenten*** hinzugefügt werden, wenn z. B. neue ***Datenquellen*** und ***Datensätze*** einbezogen werden.

Grundlegend für die parallelisierte **Komponentenentwicklung** sind die Komponentenspezifikationen, die zu erfüllen und zu validieren sind. Die Entwicklung findet für jede Komponente nach einem individuell geeigneten und domänenspezifischen Vorgehen statt. Bei klassischen Komponenten wie mechanischen oder elektrischen ***Subsystemen*** kann beispielsweise auf ein Vorgehen aus dem ***Systems Engineering*** zurückgegriffen werden. Voraussetzung ist, dass sich dieses Vorgehen in das hier beschriebene zyklische Prinzip integrieren lassen. Für die ML-Komponentenentwicklung (S. 22) sowie die Datenbereitstellung (S. 19) gibt PAISE® das Vorgehen vor.

Wurde ein Lösungsansatz für eine Komponente umgesetzt (egal ob prototypisch oder verfeinert), so kann sie in das umgebende ***Gesamtsystem*** oder das umgebende Subsystem integriert und innerhalb von Integrationstests validiert und verifiziert werden. ***Checkpoints*** synchronisieren diese Integration von Subsystemen. Komponenten können auch passiv an Checkpoints teilnehmen, wenn z.B. der Entwicklungsstand keine Integration zulässt. Bei der Integration wird dann bezüglich dieser Komponente auf den Entwicklungsstand des vorherigen Zyklus aufgesetzt, auf Simulationen zurückgegriffen oder es werden nur die Schnittstellen getestet. Abschließend findet eine Bewertung statt, mit entsprechender Dokumentation des Entwicklungsstands aller Komponenten und der Ergebnisse bezüglich des Gesamtsystems. Die Dokumentation sollte insbesondere bei ML-Komponenten durch einen Versionierungsprozess unterstützt werden, der die verwendeten Daten mit einschließt.

Insgesamt dient der Checkpoint dazu interdisziplinäre Querschnittsaspekte in den Blick zu nehmen. Neben der Betrachtung von funktionaler Sicherheit oder auch der Kosten beinhaltet das ggf. auch die offene Diskussion potentieller ethischer Konflikte, die etwa durch die Nutzung von unvollständigen Daten oder Daten mit enthaltenen Verzerrungen (Bias), erzeugt werden können. Solche Aspekte werden gezielt von der Rolle des oder der Datenbeauftragten adressiert und sind ebenso im Vorgehen der Datenbereitstellung verankert.

Das beschriebene „checkpoint-basierte"-zyklische Vorgehen hat eine stetige Verbesserung des Gesamtsystems zum Ziel. Dabei birgt es drei Eigenschaften, die aus unserer Sicht für KI-Engineering essentziell sind:

- Es berücksichtigt, dass die (Weiter-)Entwicklung mancher Komponenten von den Ergebnissen anderer abhängig ist. Beispielsweise kann die Entwicklung einer ML-Komponente erst dann effektiv stattfinden, wenn erste Daten vorhanden sind. Genauso kann die Konzeption eines Subsystems, das durch ML-basierte Methoden optimiert werden soll,

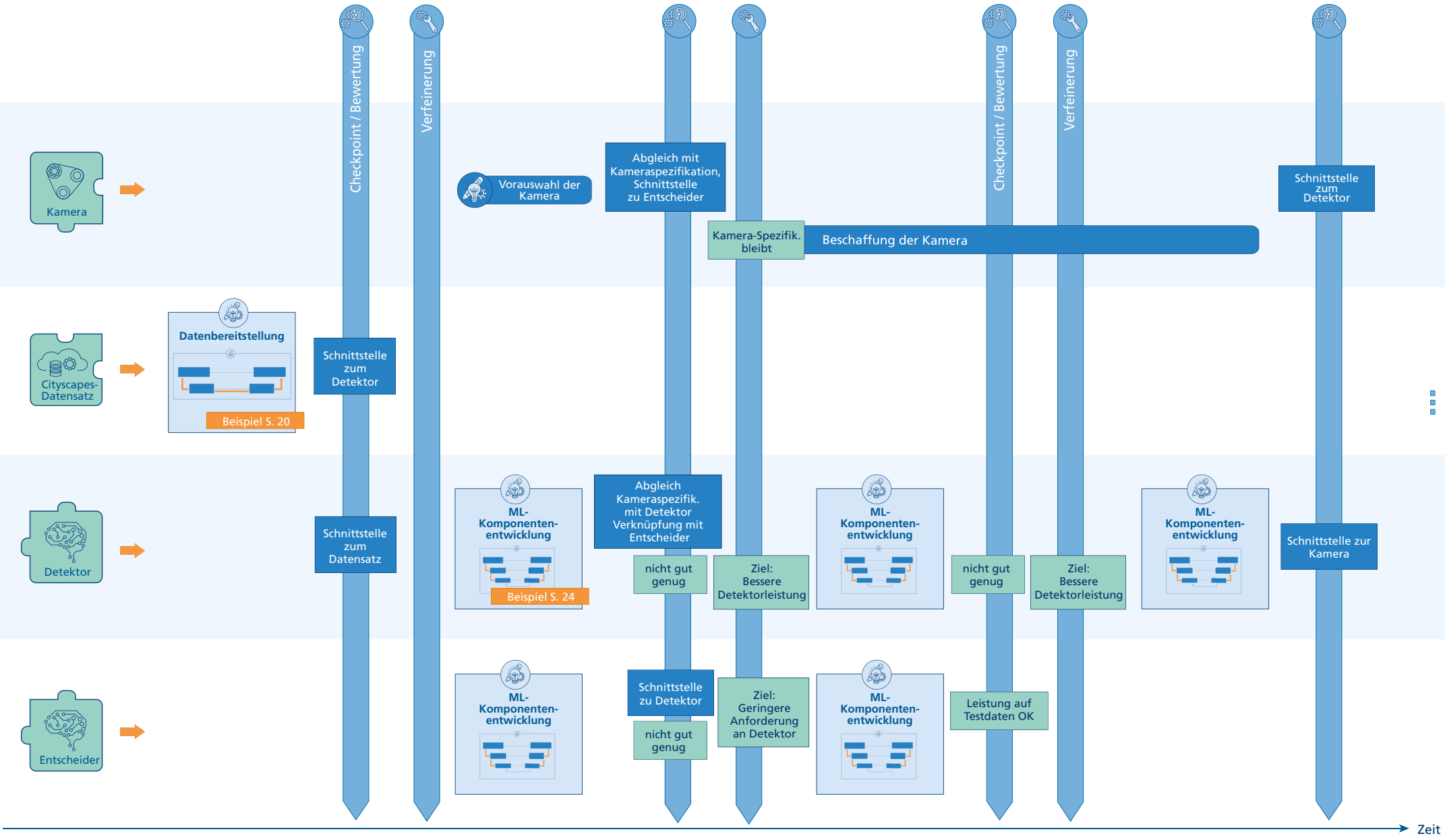


Abbildung 4: Entwicklungszyklus Beispiel

erst nach der Entwicklung des zugehörigen ML-basierten ***Hilfssystems*** begonnen werden. Die Abhängigkeiten der Komponenten untereinander gehen aus dem Systemmodell hervor. Nützlich kann hier zusätzlich eine zeitliche Abhängigkeitsdarstellung ähnlich der eines Gantt-Charts sein, wobei darauf zu achten ist, dass bei einem agilen Vorgehen nicht mit absoluten Zeitintervallen geplant werden kann.

- Es ermöglicht ein exploratives Vorgehen, das speziell für die Entwicklung von ML-basierten Komponenten notwendig ist, da hier im Voraus oft keine Garantien gegeben werden können, ob alle Anforderungen erfüllt wurden.

- Es bietet den Rahmen für eine risikobasierte Entwicklung, die alternative Lösungsansätze zulässt, diese auf Basis von Prototypen gegeneinander abwägt und bezüglich ihrer Risiken bewertet. So muss nicht zwingend davon ausgegangen werden, dass eine Komponente als ML-Komponente entwickelt wird. Beispielsweise kann sich herausstellen, dass ML-basierte Methoden nicht geeignet sind und auf klassische statistische Verfahren zurückgegriffen werden sollte.

- Gerade während der ersten Entwicklungszyklen kann es sich auszahlen, verschiedene Alternativen auch unter einem größeren Risiko grob zu betrachten während das Risiko in späteren Phasen immer weiter reduziert wird.[18] Die Risikoanalyse ist damit essenzieller Teil dieser Phase.

Nicht alle Zyklen müssen dieselbe Länge besitzen, vielmehr können sie sich an organisatorische Gegebenheiten anpassen.

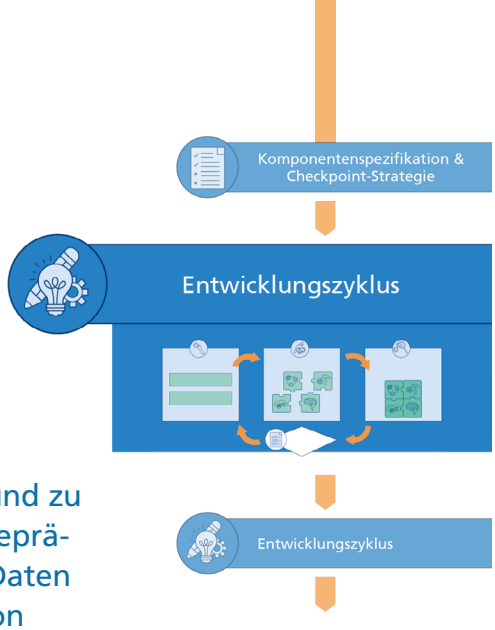
Beispiel:

In dieser Phase findet die Entwicklung aller Sub- und Hilfssysteme statt. Für eine übersichtliche Darstellung in nebenstehender Abbildung beschränken wir uns hier nur auf die Subsysteme Kamera, Detektor und Entscheider sowie auf den Cityscapes-Datensatz, der für die Entwicklung des Detektors bereitgestellt wird. Die einzelnen Aktionen sind nur abstrakt dargestellt und werden hinsichtlich ML-Komponentenentwicklung und Datenbereitstellung in den folgenden Abschnitten genauer ausgeführt.

Zu Beginn wird der Cityscapes-Datensatz aufbereitet (siehe ausführliches Beispiel im folgenden Kapitel) und als Komponente mit einer definierten Schnittstelle gekapselt. Am ersten **Checkpoint** findet der Test der Schnittstellen zwischen Cityscapes-Datensatz und Detektor sowie zwischen Cityscapes-Datensatz und des Entscheiders statt. Dieser ist erfolgreich, sodass im **Verfeinerungs**schritt keine Anpassungen bei den Spezifikationen, der Dekomposition oder den Schnittstellen vorgenommen werden müssen. Weiterhin wird als Ziel für die folgende **Komponentenentwicklung** die Entwicklung einer ersten prototypischen Version des Detektors (siehe ausführliches Beispiel S. 22) und des Entscheiders festgelegt. Zeitgleich wird der Auswahlprozess des Subsystems Kamera gestartet.

Am zweiten **Checkpoint** nehmen Kamera, Detektor und Entscheider teil. Detektor und Entscheider sind prototypisch basierend auf initial abgeschätzten Spezifikationen entwickelt worden. Die spezifizierten Kameraparameter werden mit den Annahmen des Detektors abgeglichen – es zeigt sich, dass die ausgewählten Parameter im Wesentlichen mit den Trainings- und Testdaten aus dem Cityscapes-Datensatz übereinstimmen. Jedoch erreicht der Detektor nicht die erforderliche Zuverlässigkeit der Detektion, die der Entscheider vorausgesetzt hat. In der Verfeinerung wird beschlossen, die Kameraspezifikation beizubehalten, jedoch sowohl für Detektor als auch für Entscheider neue Zielwerte vorzugeben: Der Detektor soll seine Leistung verbessern, der Entscheider soll seine Robustheit gegen Detektionsschwächen steigern und somit seine Anforderungen an die Detektorleistung senken. Zeitgleich wird der Beschaffungsprozess der Kamera gestartet, der sich über die nächsten zwei Zyklen erstreckt. Im Verlauf der folgenden **Komponentenentwicklung** erreicht der Entscheider die spezifizierten Zieleigenschaften auf den bisherigen Testdaten. Während der Integrationstests im **Checkpoint** ergibt sich jedoch, dass im Zusammenspiel von Detektor und Entscheider noch weniger als 99 Prozent der im Datensatz enthaltenen kritischen Situationen auch als solche bewertet werden. In der **Verfeinerung** wird entschieden, dass das größere Optimierungspotenzial beim Detektor liegt und der Entscheider vorerst auf seinem aktuellen Stand belassen wird. Die **Komponentenentwicklung** konzentriert sich nun auf die Weiterentwicklung des Detektors. Am nachfolgenden **Checkpoint** wird das Zusammenspiel zwischen Detektor und Entscheider wieder getestet. Zusätzlich kann die inzwischen gelieferte Kamera an den Detektor angebunden werden, damit erste funktionale Tests durchgeführt werden können. Nachfolgende Schritte umfassen das Erheben und Annotieren von Realdaten mit dem Ziel-Kamerasystem, um den Entwicklungsstand des Detektors innerhalb des Gesamtsystems zu bewerten.

Datenbereitstellung



Das Vorgehen der **Datenbereitstellung** hat zum Ziel, Trainings-, Test- und Validierungsdatensätze zu erzeugen, aufzubereiten und zu bewerten. Dabei sollen Forderungen bezüglich der Relevanz, Repräsentativität und Fehlerfreiheit der Daten erfüllt werden.[19] Die Daten bilden die Grundlage für die Entwicklung und Funktionalität von KI-Komponenten.

Im Schritt der **Verfeinerung** wird die Spezifikation der Daten angepasst. Dazu zählen sowohl ***Datenquellen***, die für Training, Test und Laufzeit unterschiedlich sein können, als auch Anforderungen an die Daten selbst.

Die Anforderungen können technische Aspekte umfassen, die für die Erfüllung der gesetzten Aufgaben der KI-Komponente relevant sind, wie etwa die Menge an Daten, die Qualität, also ob fehlende oder falsche Angaben vorhanden sind, und die Repräsentativität, also ob die Trainingsdaten auch die Daten, die zur Laufzeit anfallen, repräsentieren können. Zusätzlich gibt es übergeordnete nicht-technische Aspekte wie Verzerrungen in der Verteilung der Daten (bias), die zu unfairen Entscheidungen führen können (z. B. geschlechterspezifische Entscheidungen bei der Personalauswahl), Kosten bei der Datenbeschaffung oder rechtliche Aspekte von personengebundenen Daten, die wiederum zusätzliche Schritte, wie Anonymisierung oder Pseudonymisierung nötig machen können.

Das anschließende Vorgehen zur Datenbereitstellung orientiert sich am V-Modell[20]. Auf einer übergeordneten Ebene werden in einem ersten Schritt die Zielmetriken definiert, anhand derer die Daten später bewertet werden sollen. Die Zielmetriken werden hierbei von den oben genannten Anforderungen abgeleitet.

Danach folgt das Experiment bzw. die Datensammlung. Unter dem Begriff „Experiment" ist hier eine Datensammlung unter kontrollierten Bedingungen zu verstehen. Beispielsweise kann für eine Anomalieerkennung das Sammeln von Daten – mit Repräsentanten bekannter Anomalien und des Normalzustands – durch geführte Experimente erfolgen. Im Fall von überwachtem Lernen erleichtert dies das Labeling, also das Zuordnen von Zielwerten, die von einem ML-basierten Algorithmus vorhergesagt werden sollen. In den Schritt der Datensammlung fällt zusätzlich das Aufnehmen von Daten unter realen Einsatzbedingungen und die Sammlung von künstlich erzeugten Daten, z. B. aus Simulationen, oder die Vermehrung eines schon vorhandenen ***Datensatzes*** durch Techniken der Data Augmentation. Ebenso kann in dieser Phase eine Auswahl von öffentliche Datensätzen in Betracht gezogen werden.

Die akquirierten Rohdaten werden in der **Datenaufbereitung** bezüglich der Problemstellung gesichtet und vorbereitet. Basierend auf der Spezifikation werden z.B. Features, also Eingabemerkmale, abgeleitet. Diese Features bilden die Grundlage für die korrekte Funktionalität der ML-Komponente. Die Auswahl der Features stützt sich vorrangig auf Domänen- bzw. Expertenwissen, aber auch Signifikanzbetrachtungen können hier maßgeblich sein.

Zusätzlich können Techniken der Aggregation mehrerer Datenpunkte, der Rauschentfernung oder aber das Filtern von unvollständigen Datenpunkten in diesem Schritt zum Einsatz kommen. Eine weitere hilfreiche Methode ist die der Data Imputation, also das Schätzen von fehlenden Werten. So können z. B. bei einem kurzzeitigen Ausfall eines Sensors das entsprechende Feature und die unvollständigen Datenpunkte trotzdem genutzt werden. Des Weiteren lassen sich mehrere Features zu einem neuen Feature kombinieren, um die Dimension und damit die Komplexität der Datenpunkte zu reduzieren.
Im Falle des überwachten Lernens im Zusammenhang mit einer Klassifikation werden in diesem Schritt, falls nicht schon automatisch bei der Aufnahme der Daten geschehen, die Daten annotiert, d.h. die Datenpunkte werden ihren jeweiligen

18 Vgl. Spiralmodell [B. Boehm. (1986). A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 11 (4): 14–24.]

19 Diese drei Aspekte werden im Vorschlag der Europäischen Kommission für gesetzliche Regelungen bezüglich KI gefordert [European Commission. (2021). Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative ActBrüssel: COM/2021/206 final.]

20 [B.W. Boehm. (1981). Software Engineering Economics, Prentice Hall.]
[J. Friedrich, M. Kuhrmann, M. Sihling, U. Hammerschall. (2009). Das V-Modell XT. Informatik im Fokus. Springer, Berlin, Heidelberg.]

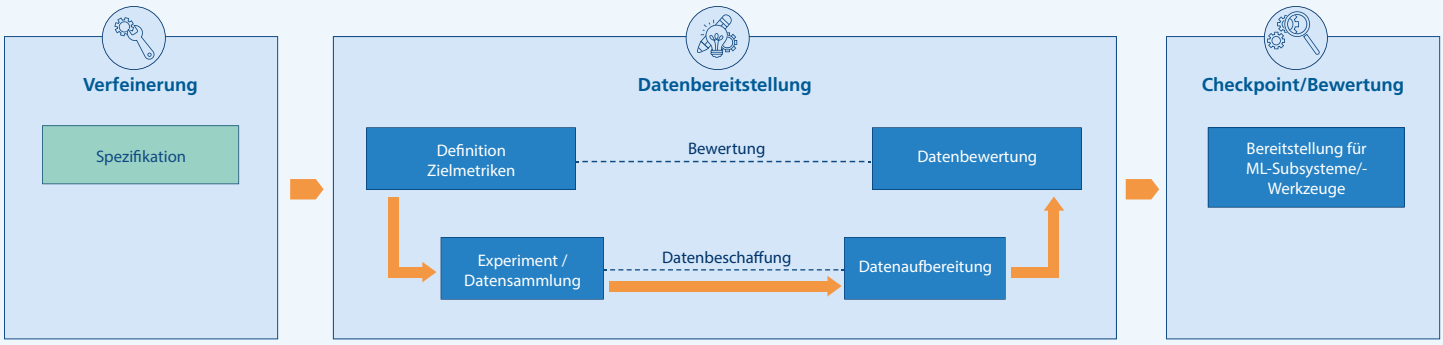


Abbildung 5: Schematische Darstellung des Vorgehens zur Datenbereitstellung

Klassen zugeordnet. Im Rahmen der Entwicklung können der KI-Komponente auch Informationen bezüglich des Zusammenhangs zwischen Datenpunkt und Klasse mitgegeben werden. Je nach Anforderung werden in diesem Schritt auch Techniken der Anonymisierung oder Pseudonymisierung von Daten angewandt.
Während manche der Aufbereitungsschritte nur für Trainings-, Test-, und Validierungsdaten nötig sind (z. B. die Annotation), müssen einige Methoden aus Gründen der Konsistenz auch auf die Daten zur Laufzeit angewendet werden (z. B. Rauschentfernung, Data Imputation, Kombination von mehreren Features etc.).

Das Vorgehen der Datenbereitstellung wird mit dem Schritt der Datenbewertung abgeschlossen. Hier kommen die vorher definierten Zielmetriken bezüglich der Anforderungen an die Daten zum Einsatz. Der Datenbewertung kommt, neben dem technischen Aspekt, auch unter rechtlichen Aspekten eine wichtige Funktion zu wenn Nachweise gegenüber externen Organisationen erbracht werden müssen. Dazu sollen laut EU-Vorschlag geeignete Governance- und Datenverwaltungsverfahren genutzt werden.

Die aufbereiteten und bewerteten Daten werden nach erfolgreichem Abschluss des Schrittes für die Entwicklung von KI-Komponenten bereitgestellt.

Es sei hervorgehoben, dass der Prozess der Datenaufbereitung nicht zwangsläufig händisch ablaufen muss. Gerade hier gibt es ein großes Potenzial für die Automatisierung von einzelnen Arbeitsschritten oder Schrittfolgen.

Beispiel:

Für die Subsystementwicklungen von Detektor und Entscheider sind Bilddaten der Fahrzeugkamera erforderlich, in denen vorausfahrende Fahrzeuge annotiert sind. Da diese Daten zu Projektbeginn nicht vorliegen, wird der öffentlich verfügbare Cityscapes-Datensatz genutzt, der reale Fahrzeugkamerabilder mit manuell annotierten Objekten enthält. Im Rahmen der Datenbereitstellung ist zu prüfen, inwieweit der Datensatz zur Zielanwendung kompatibel ist, also wie repräsentativ er ist (etwa in Bezug auf die Kameraauflösung, die Fahrszenarien, etc.). Im Folgenden beschränken wir uns beispielhaft nur auf den Aspekt der Fahrszenarien. Dafür werden im ersten Schritt Zielmetriken für die Verkehrssituation im Cityscapes-Datensatz festgelegt. Im weiteren Schritt wird der Datensatz für die Auswertung vom Anbieter heruntergeladen. In der Datenbereitstellung werden die Bilder herausgefiltert, auf denen nur Personen zu sehen sind, die zu Fuß gehen bzw. mit dem Rad fahren, da diese für den Zielanwendungsfall der Autobahnfahrt keine Bedeutung haben. Im letzten Schritt der Datenbewertung wird der gesamte Datensatz noch einmal hinsichtlich der Repräsentativität der Szenarien geprüft. Es ergibt sich, dass die Szenarien für eine vorläufige Entwicklung akzeptabel sind, jedoch zu wenige Daten aus dem Zielanwendungsfall enthalten. Cityscapes legt einen starken Fokus auf urbane Szenarien, wohingegen die Zielanwendung primär Autobahnen adressiert. Für eine erste prototypische Entwicklung des Detektors wird Cityscapes als Datenquelle für ausreichend befunden; später im Projekt sollen jedoch mittels einesHilfssystems, einer Fahrzeugsimulation, Bilder künstlich erzeugt und anschließend mit KI-Verfahren nachprozessiert werden, um sie realistischer zu machen. Schließlich sollen, sobald verfügbar, mit dem realen Ziel-Kamerasystem Messdaten im öffentlichen Verkehr erhoben und manuell annotiert werden. Dabei kommen zu den Fragen der Repräsentativität und der Datenqualität Fragen des Datenschutzes hinzu.

a) Reale Daten aus dem Cityscapes-Datensatz

b) Daten aus dem realen Ziel-Kamerasystem

c) Trainings- und Testdaten müssen auch die Breite der Ziel-Herausforderungen abdecken.

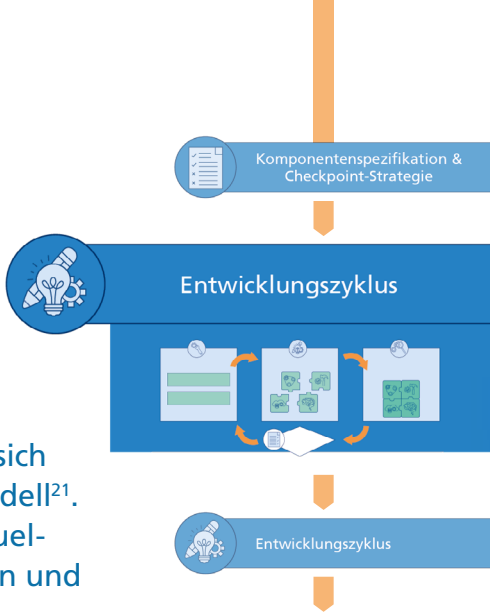
d) Simulierte Bilddaten aus einer 3D-Rendering-Engine.

e) Simulierte Bilddaten, mit maschinellen Lernverfahren, bspw. sogenannten Generative Adversarial Networks verbessert.

f) Simulierte Daten können Annotationen automatisch mitliefern.

Abbildung 6: Mögliche Datenquellen und Herausforderungen in der Beispielanwendung.

ML-Komponentenentwicklung



Das Vorgehen bei der ML-Komponentenentwicklung orientiert sich an dem im Software und Systems Engineering etablierten V-Modell[21]. Ziel ist eine enge Integration mit ML-Hilfssystemen und Datenquellen, um innerhalb der Checkpoints Ergebnisse iterativ integrieren und validieren zu können.

Durch dieses Vorgehen soll ein ***ML-Modell*** (also der datengetrieben gelernte Teil) in eine möglichst genau spezifizierte ***Komponente*** gekapselt werden. Dabei wird eine organisatorische
Schnittstelle zwischen der klassischen Data-Science-Disziplin und dem ***Systems Engineering*** geschaffen.

Ausgangspunkt ist die Spezifikation der Komponente im Kontext des umgebenden Systems, die innerhalb der **Verfeinerung** iterativ detailliert und angepasst wird. Zur Spezifikation zählt unter anderem das ML-Verfahren (z. B. Neuronales Netz, Entscheidungsbaum etc.) als möglicher Lösungsansatz auf Basis der Anforderungen an die Komponente. Bei dessen Wahl fließen sowohl übergeordnete Anforderungen an das ***Gesamtsystem*** ein (z. B. die Nachvollziehbarkeit von Entscheidungen) als auch direkte Abhängigkeiten von anderen Komponenten (z.B. beschränkte Rechenressourcen, Verfügbarkeit von Daten, Verfügbarkeit von Zielgrößen).

Der erste Schritt befasst sich mit der Integration der ***Datenquellen*** für Training, Tests und Validierung. Dabei kann es vorkommen, dass die Datenquellen und damit die Schnittstellen nicht in jedem Zyklus dieselben sind, z. B. falls zunächst Daten in Tabellenform vorliegen und später in einer Datenbank.

Bei der Entwicklung der Test- und Validierungsmetriken werden globale Kostenfunktionen aus den Anforderungen an die Komponente abgeleitet, die sich für die datengetriebene Bewertung eignen. Hierbei sollte Domänenwissen einfließen, um ML-Komponenten einzeln, jedoch mit Bezug auf ihre Funktion innerhalb des Gesamtsystems, testen zu können.

Das ML-Verfahren wird im nächsten Schritt zu einer konkreten ML-Architektur implementiert, mit festgelegten Hyperparametern. Beispiele hierfür sind die Festlegung der Anzahl der Neuronen und Schichten in künstlichen Neuronalen Netzen sowie die Definition der lokalen Kostenfunktion und Lernrate.

Die ML-Architektur wird im Modelltraining in ein für die zu erfüllende Funktionalität geeignetes ML-Modell überführt. Da die Ergebnisse des erlernten Modells stark von der ML-Architektur, also den gewählten Hyperparametern, abhängen, werden diese mehrfach variiert. Ziel ist es die Hyperparameterkonfigurationen zu finden, die auf Basis der lokalen Kostenfunktion und eines Validierungsdatensatzes zum besten Modell führen. Diese Hyperparameteroptimierung kann in vielen Fällen durch geeignete Werkzeuge teilweise oder komplett automatisiert werden (z.B. durch Auto-ML Systeme).

Bei der anschließenden Modellbewertung wird anhand eines Testdatensatzes und der zuvor definierten Test- und Validierungsmetriken die Güte des erlernten und optimierten ML-Modells bewertet. Somit lässt sich die Genauigkeit und Performanz des Modells bezüglich bisher ungesehener Daten und Metriken bewerten, die auf die zu erfüllende Funktionalität der Komponente zugeschnitten sind.

Der letzte Schritt innerhalb der ML-Komponentenentwicklung bildet die Modularisierung als Komponente. Hier wird das trainierte und validierte ML-Modell so aufbereitet, dass es auf die Zielplattform gebracht werden kann. Während das Modell zuvor nur auf Basis von Daten validiert wurde, wird nun sichergestellt, dass das Modell auf der Zielplattform, wie z.B. auf einem ressourcenbeschränkten eingebetteten System, lauffähig ist und dort vergleichbare Ergebnisse liefert.

Zum **Checkpoint** findet die eigentliche Integration der Komponenten in das übergeordnete System statt. Hier erst zeigt sich innerhalb von Tests, ob die Spezifikation und die Ableitung der Architektur gelungen ist und ob sich die komponentenspezifisch erreichten Metriken auch positiv auf das Gesamtsystem auswirken. Sollte dies nicht der Fall sein, sind weitere Zyklen

21 [B.W. Boehm. (1981). Software Engineering Economics, Prentice Hall.]
[J. Friedrich, M. Kuhrmann, M. Sihling, U. Hammerschall. (2009). Das V-Modell XT. Informatik im Fokus. Springer, Berlin, Heidelberg.]

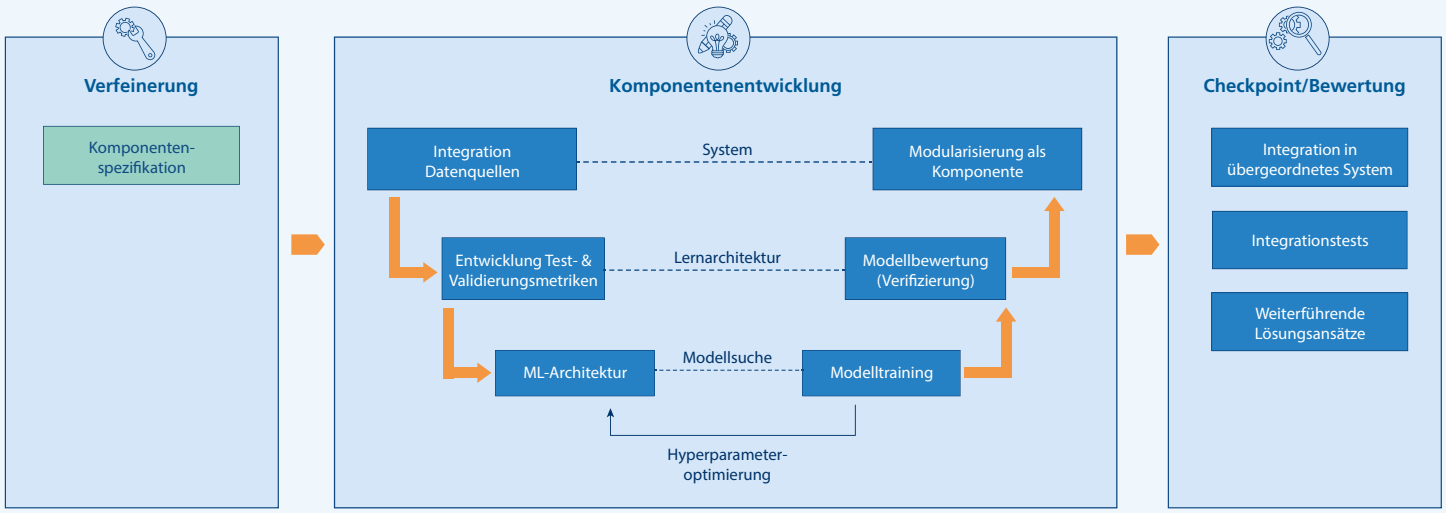


Abbildung 7: Schematische Darstellung des Vorgehens zur ML-Komponentenentwicklung.

notwendig, in denen in Absprache mit den anderen Komponenten weitere Lösungsansätze, wie z .B. andere ML-Verfahren, erprobt werden. Die Ziele weiterer Zyklen können ebenso über die Anforderungen hinausgehende Optimierungen sein, wie etwa die Steigerung der Vorhersagegenauigkeit des ML-Systems. Zur Steuerung des Entwicklungsprozesses ist es wichtig, dass innerhalb des Checkpoints anhand der erstellten Berichte Metriken in konkrete Schätzungen zu Kosten und Risiken übersetzt werden, um Entscheidungen bzgl. der Weiterentwicklung ableiten zu können.

Leitfragen	Ergebnisse
• Wie können Anforderungen in Test- und Validierungsmetriken überführt werden? • Welche Kostenfunktionen eignen sich für das Modelltraining? • Welches ML-Verfahren eignet sich, um eine optimale Komponente im Sinne des Gesamtsystems zu erhalten? • Welche Verbesserungen können aufgrund von Veränderungen in anderen Subsystemen (z.B. Daten) erreicht werden? • Gibt es verfügbare KI-Komponenten von externen Anbietern, die genutzt und weiterentwickelt werden können?	• ML-Komponente mit einer klaren Dokumentation der Anforderungen, der genutzten Daten und Werkzeuge • Ergebnisse aus der Evaluierung von Test- und Validierungsmetriken • Einschätzung zu Verbesserungspotenzialen in Abhängigkeit von anderen Subsystemen • Kontinuierliches Test- und Überwachungskonzept der ML-Komponente unter Nutzung von Test- und Validierungsmetriken

Beispiel:

Für diesen Einblick liegt der Fokus auf der Subsystementwicklung des Detektors. Dieser soll vorausfahrende Fahrzeuge im Videostream einer Einzelkamera erkennen. Beispielhaft wird hier die Entwicklung anhand eines schon vorhandenen externen Datensatzes namens „Cityscapes" dargestellt

Im Rahmen der Komponentenspezifikation wird folgender Lösungsansatz als vielversprechend bewertet: Die Aufgabe soll mithilfe eines künstlichen Neuronalen Netzwerks zur Erkennung von Objekten in Einzelbildern gelöst werden, das auf einer kleinen GPU-basierten Recheneinheit ausgeführt wird. Das Neuronale Netz erhält Kamerabilder über eine Ethernet-Schnittstelle, und gibt Ergebnisse über eine RAM-Schnittstelle an den Entscheider weiter, der ebenfalls auf dieser Recheneinheit ausgeführt wird. Zielvorgabe sind mindestens zwei Detektionen pro Fahrzeug innerhalb einer Sekunde Sichtbarkeit bei einem Abstand von bis zu 50 Metern Entfernung. Bei selteneren Detektionen darf der Tracker das Objekt als Fehldetektion verwerfen. Es sind weniger als 0,2Prozent Fehler in dieser Detektion erlaubt.

Im Rahmen der Integration der Datenschnittstellen werden die technischen Schnittstellen im Zielsystem (Ethernet-Streams von Kamerabildern) in operative Schnittstellen (Videobilder im RAM) überführt, und die KI-bezogenen Softwareteile der Komponente aus der Embedded-Hardware gelöst, um sie beispielsweise auf einem PC oder in einem Rechencluster entwickeln, trainieren und testen zu können. Es wird festgelegt, dass das System zunächst in der Programmiersprache Python entwickelt wird, und aufgrund der fehlenden Datensätze aus dem Realsystem ein bestehender annotierter Fahrzeugdatensatz, Cityscapes[22], verwendet wird, der dem Zielsystem grundsätzlich ähnelt.

Im Rahmen der Entwicklung von Test- und Validierungsmetriken werden die Anforderungen an die Komponente (zwei Detektionen pro Sekunde) auf Anforderungen an das spezifizierte ML-Verfahren adaptiert. Da dieses Verfahren nur eine Einzelbildauswertung nutzt, können Zeitdauern nicht direkt übernommen werden. Es wird eine Bildrate von 25 Bildern pro Sekunde angenommen und gefordert, dass unter 25 aufeinanderfolgenden Einzelbildern jedes Objekt mindestens zweimal erkannt worden sein muss. Außerdem darf diese Anforderung nur bei weniger als 0,2 Prozent der Objekte im Datensatz verletzt werden.

Im Rahmen der Wahl der ML-Architektur wird konkretisiert, dass ein „Mask R-CNN"-Ansatz[23] gewählt wird, der Einzelobjekte anhand ihrer Bild-Pixelbereiche erkennt. Um dieses Neuronale Netz zu trainieren, muss die die Kostenfunktion auf Einzelbilder und ihre Pixel heruntergebrochen werden. Obwohl also eigentlich „erkannt" vs. „nicht erkannt" für ein ganzes Bild bewertet werden soll, ist eine dementsprechende Kostenfunktion für das Training ungeeignet. Praktischer ist es, im Training jeden falsch klassifizierten Pixel zu „bestrafen" – und die Leistungsfähigkeit des Neuronalen Netzes über die gängige mIoU-Metrik zu bewerten. Hier findet ein wesentlicher Bruch in den Anforderungen statt: Es wird ein Netz trainiert, das Objektumrisse so genau wie möglich erkennt, um eine ML-Komponente zu erhalten, die Objekte möglichst häufig erkennt.

Das Modelltraining wird also auf Basis der mIoU-Metrik anhand des Cityscapes-Datensatzes durchgeführt. Sollten die Ergebnisse nicht zufriedenstellend sein, können Hyperparameter, wie z. B. die Anzahl der Schichten des Neuronalen Netztes, angepasst werden.

Im Rahmen der Modellbewertung wird das ML-Modell auf Grundlage der Einzelbildfolgen bewertet, das heißt, Objektumrisse aus dem Mask-R-CNN-Resultat werden auf hinreichende Größe geprüft und diese Metrik auf Bildsequenzen aus dem Cityscapes-Datensatz angewendet, um zu bewerten, ob das Ziel von zwei Detektionen je 25 Einzelbilder zumindest auf dem Cityscapes-Datensatz erfüllt wird.

Im Rahmen der Modularisierung als Komponente wird das System auf die GPU-Recheneinheit übertragen, und die Cityscapes-Datensatz-Bilder werden simulativ über die reale Ethernet-Schnittstelle eingespielt, um einen echten Videostream nachzubilden. Hier wird der Maßstab von zwei Detektionen je Sekunde auf der realen Plattform der Komponente erprobt, jedoch (in dieser Iteration) noch ohne reale Daten.

Die Komponente kann am Checkpoint jetzt in ihren Spezifikationen und mit der erreichten Ergebnisqualität auf Kompatibilität mit den parallel entwickelten Komponenten geprüft werden

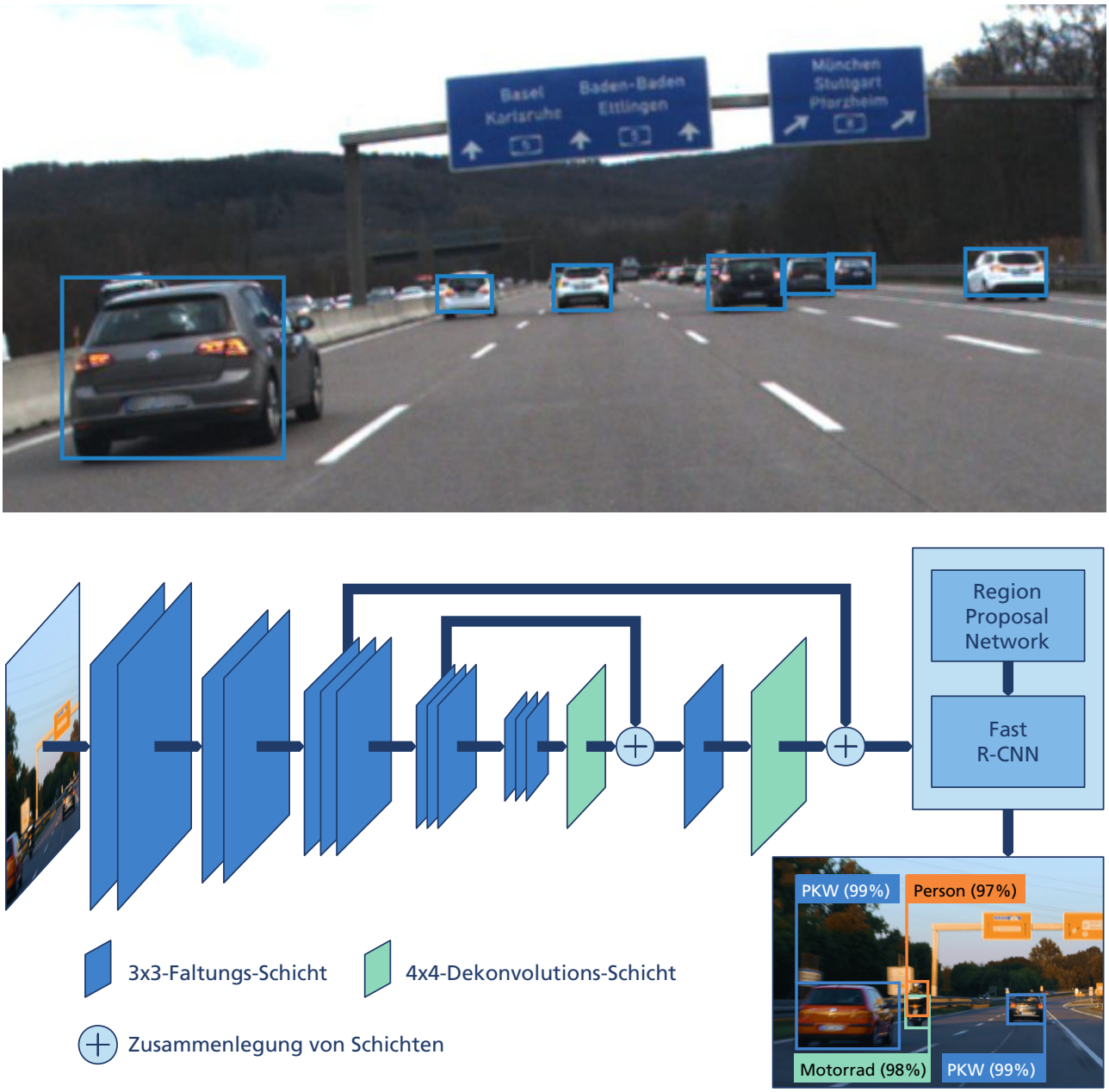


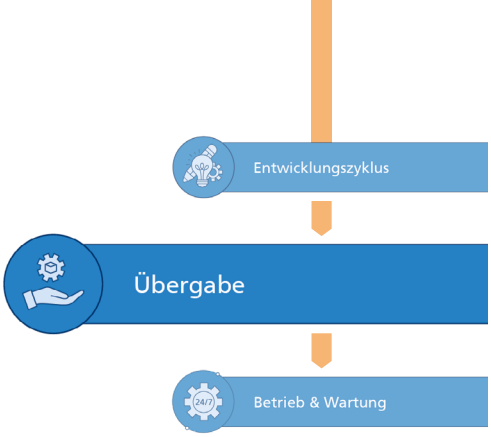
Abbildung 8: Schematische Darstellung eines R-CNN. Adaptiert aus[25].

22 cityscapes-dataset.com
[M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth und B. Schiele. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)]

23 Ein System basierend auf einem neuronalen Netz, das zur Erkennung von Objekten in Bildern genutzt wird. [K. He, G. Gkioxari, P. Dollár, R. Girshick. (2017). Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision, S. 2961–2969.]

24 Die „mean intersection over union"-Metrik bewertet den durchschnittlichen Überlappung zwischen maschinell erkanntem Objekt und real bekanntem Objekt.

25 [L. Sommer et al. "Multi Feature Deconvolutional Faster R-CNN for Precise Vehicle Detection in Aerial Imagery," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 635-642]



Übergabe

In dieser Phase wird das fertig entwickelte Produkt vom Entwicklungsteam an die organisatorischen Einheiten übertragen, die sich um den Betrieb und die Wartung kümmern. Dazu werden Dokumentationen für die Anwenderinnen und Anwender und eventuell ein Serviceteam erstellt. Speziell bei KI-Systemen werden hier Fragestellungen bezüglich der Fehleranfälligkeit und Wartung (das Nachtrainieren) von ML-Modellen betrachtet.

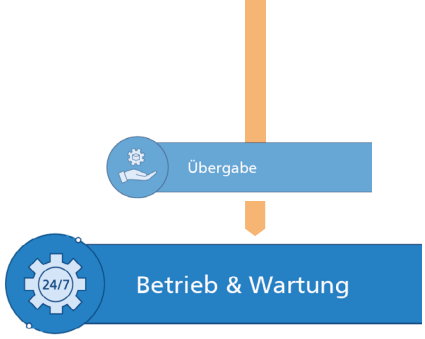
Die für die Phase des **Betriebs** notwendigen Informationen werden für die Anwenderinnen und Anwender und das Serviceteam aufbereitet, z. B. in Form einer Bedienungsanleitung. Diese schließt Eskalationsstufen beim Auftreten von Fehlerzuständen im System mit ein. Speziell beim Einsatz von ML-basierten Systemen müssen hier Fragestellungen bezüglich der Beständigkeit des Modells und der Konzepte zur Erkennung von fehlerhaften Modellen abschließend geklärt werden.

Je nach Art des Systems und abhängig von seinem Autonomiegrad und der Risikobewertung für Mensch und Umwelt müssen zusätzliche Randbedingungen, wie etwa Meldepflichten und Konformitätserklärungen, in dieser Phase adressiert werden. Dies betrifft speziell ***Hochrisiko-KI-Systeme.***

Beispiel:

Das System wird dem beauftragten Unternehmen übergeben. Dieses analysiert sowohl die Designprinzipien als auch die Ergebnisqualität anhand der Artefakte und prüft die Abnahme. Es wird spezifiziert, dass die ML-basierte Objekterkennung wesentlich auf Bestandsdatensätzen zu Fahrzeugbildern fußt, und zukünftige Erscheinungsbilder womöglich nicht abbilden kann (bspw. das künftige Aufkommen von Shuttle-Kleinbussen, die im Datensatz nicht enthalten sind). Es wird spezifiziert, dass sich das System durch Abgleich mit einfachen Nicht-ML-Verfahren begrenzt selbst überwacht. Zum einen sind regelmäßige Erhebungsfahrten notwendig, um aktualisierte Datensätze für ein Nacherproben und Nachtrainieren bereitzustellen, und zum anderen müssen potenziell jährlich aktualisierte Parameter im Fahrzeug gesetzt werden. Es wird an die zuständigen Behörden kommuniziert, dass der Einsatz des Systems aufgrund seiner Eigenschaft als Hochrisiko-KI-System registriert und regelmäßig im Rahmen einer Wartung überprüft werden muss.

Leitfragen	Ergebnisse
• Wie können Veränderungen in der Datenverteilung, die zu einem veränderten Verhalten der ML-Komponenten führen, erkannt werden? • Nach welchen Kriterien sollen Wartungsarbeiten, z. B. das Nachtrainieren eines ML-Modells, eingeleitet werden? • Wie kann das Modell hinsichtlich fehlerhaften Verhaltens überwacht werden? • Wer ist für fehlerhaftes Verhalten verantwortlich, wenn das System während des Betriebs weiter lernt?	• Bedienungsanleitung • Dokumentation für die Serviceabteilung des Unternehmens • Wartungskonzept (inklusive Aktualisierungen der KI-Komponente(n))



Betrieb & Wartung

In der abschließenden Phase des Betriebs und der Wartung wird das im vorhergehenden Schritt definierte Service- und Wartungskonzept umgesetzt. Ziel ist es, alle Funktionalitäten während des Betriebes sicherzustellen.

In Bezug auf ***KI-Subsysteme*** umfasst diese Phase insbesondere die Überwachung und regelmäßige Überprüfung der ML-Modelle (Monitoring). Manche ***KI-basierten*** Systeme werden vor ihrer Auslieferung getestet (und ggf. auch zertifiziert) und in der Betriebsphase nicht mehr verändert. Andere KI-basierte Systeme werden dagegen anhand von Daten aus dem Betrieb kontinuierlich aktualisiert. Zwischen diesen beiden Extremen gibt es viele Abstufungen.

Durch Veränderungen in den Daten, die während des Betriebs verarbeitet werden, kann die Performanz der KI-Subsysteme über die Zeit degradieren. Solche Veränderungen können sowohl durch latente Einflussgrößen (z. B. Temperatur, Luftfeuchtigkeit etc.) als auch durch Veränderungen am Einsatzzweck (z. B. ausländische Verkehrsschilder; neues Material, das in der Maschine verarbeitet wird etc.) herbeigeführt werden.

Der Auslöser für eine Aktualisierung des KI-Subsystems kann sowohl statisch als auch basierend auf den im Betrieb gesammelten Daten festgelegt werden. Im letzteren Fall kommen Metriken über die Modellgüte oder Veränderungen in der den Daten zugrunde liegenden Verteilung zum Einsatz. Die gesammelten Daten werden im Rahmen der Datenbereitstellung (siehe S. 19) verarbeitet und für die Weiterentwicklung des KI-Subsystems vorbereitet, die in einem nächsten Schritt mithilfe des Vorgehens nach der ML-Komponentenentwicklung (siehe S. 22) stattfindet. Abschließend wird die aktualisierte ***Komponente*** wieder in das ***Gesamtsystem*** integriert, getestet und in Betrieb genommen.

Beispiel:

Während des Betriebs werden die Parameter des Systems im Rahmen der jährlichen Wartungsintervalle geprüft, und Auffälligkeiten, die in der Selbstüberwachung des Systems aufgetreten sind, analysiert. Es werden regelmäßige Datenerhebungen durchgeführt, sowie Neuzulassungen von anderen Fahrzeugen mit ungewöhnlichen Formen (z.B. neuartige Shuttle-Kleinbusse) gezielt dahingehend untersucht, ob das Bremssystem eine hinreichend hohe Erkennungsrate mit diesen Fahrzeugen erzielt.

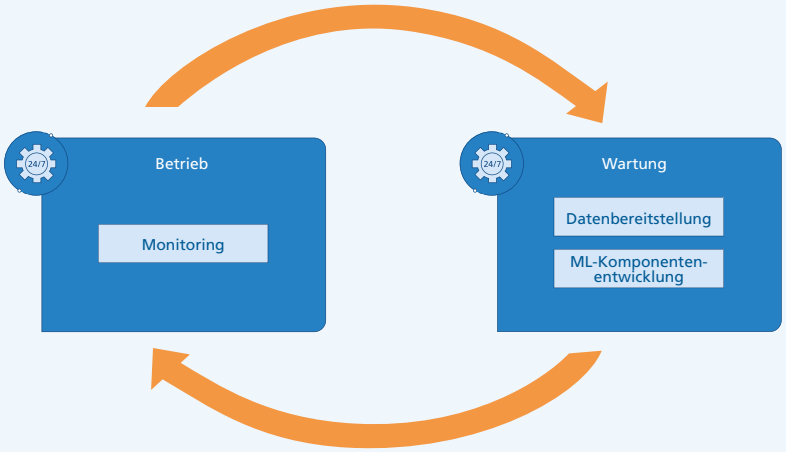


Abbildung 9: Schematische Darstellung der Substruktur der Betriebs- und Wartungsphase.

Rollenverteilung

Vorhaben und Projekte im Bereich des KI-Engineering sind meist interdisziplinär und können sehr komplex sein. Daher verlangt es nach einer Rollenverteilung, die Zuständigkeiten und Verantwortlichkeiten klar definiert.

Die aus unserer Sicht wichtigsten Rollen und Expertisen sind im Folgenden aufgelistet. In der Phase **Anforderungen & Lösungsansatz** sollte die Notwendigkeit jeder Rolle geprüft und einer Person zugeordnet werden. Es ist auch möglich, dass mehrere Rollen in einer Person vereint sind.

Projektsponsoring / beauftragendes Unternehmen:
Das Projekt-Sponsoring oder das beauftragende Unternehmen gibt das Budget für das Projekt frei und formuliert den zu bearbeitenden Auftrag. Von hier aus werden die Rahmenbedingungen, Ziele und Anforderungen definiert und regelmäßige Informationen zum Projektstand eingefordert.

Projektleitung:
Die Projektleitung organisiert und strukturiert das Projekt. Sie legt bzw. fordert benötigte Ressourcen für die Erreichung der Ziele an und hält den Kontakt zum Projektsponsoring aufrecht. Die Projektleitung ist auch dafür zuständig, die eventuell hohe organisatorische Komplexität durch Einteilung der Teams, Zuordnung von Verantwortlichkeiten und Sicherstellen der Kommunikation abzufangen und so das Projekt zum Erfolg zu führen.

Domänenexpertin/-experte:
Eine Domänenexpertin oder ein Domänenexperte weist ein hohes Verständnis für die Prozesse der Anwendungsdomäne auf, aus denen die Daten für die Entwicklung der KI-Komponente stammen. Er bzw. sie wird im besonderen Maße bei Anwendungen mit hohem Bezug zur physikalischen Realität hinzugezogen.

Sicherheitsbeauftragte:
Sie befassen sich mit der funktionalen Sicherheit (safety) bei hoch kritischen Systemen, erstellen Risikoabschätzungen und -bewertungen und sind dafür verantwortlich, dass die geforderten kritischen Grenzen eingehalten werden.

Weitere Aufgaben von Sicherheitsbeauftragten sind die Einschätzung von Risiken, die sich bei einem Ausfall der Anwendung ergeben, und die Einleitung entsprechender vorbeugender Maßnahmen ein.

Anwendende/Bedienende:
Anwenderinnen und Anwender haben eine beratende Funktion, um die Praxistauglichkeit der Anwendung einzuschätzen und auf Mängel hinzuweisen. Sie verfügen insbesondere über Erfahrungen im Bereich der Schnittstelle zwischen Mensch und Anwendung.

Automatisierungsingenieurin bzw. -ingenieur:
Ein Automatisierungsingenieurin bzw. ein Automatisierungsingenieur sorgt für die Umsetzung der in der Software generierten Befehle durch die Implementierung der Prozesssteuerung. Besonders bei einem hohen Grad an Autonomie ist diese Expertise gefordert, da hierfür Entscheidungen, die durch KI-basierte ***Komponenten*** getroffen werden, automatisiert an die Aktorik weitergegeben werden müssen.

KI-Expertin/-Experte:
KI-Expertinnen und -Experten erstellen datengetriebene Modelle und validieren und verifizieren diese. Sie verfügen über ein grundlegendes Verständnis der Zusammenhänge, die den Daten zugrunde liegen.

Verantwortliche/Verantwortlicher für IT-Sicherheit:
Diese Rolle hat das Ziel, die Aspekte „Vertraulichkeit der Daten" und „Integrität der Anwendung" sicherzustellen.
Der Einsatz von ***KI-basierten*** Verfahren erfordert die Verarbeitung von Daten, sowohl in der Entwicklung als auch im Betrieb des Systems. Speziell personenbezogene Daten müssen laut DSGVO gegen Missbrauch gesichert werden, aber auch firmeninterne Daten können zum Ziel von Hackern werden.
Die Integrität einer Anwendung umfasst das Sicherstellen der Unveränderlichkeit ihrer Funktion. Es geht darum, dem Fall vorzubeugen, dass die Anwendung für andere Zwecke als vorgesehen missbraucht werden kann, z. B. indem einem kontinuierlich lernenden System falsche Sachen beigebracht werden.

Softwareentwicklung:
Softwareentwicklerinnen und -entwickler erstellen die Softwarearchitektur, koordinieren die Entwicklung und führen diese aus. Dabei arbeiten sie eng mit KI-Expertinnen und -Experten zusammen, um die erstellten KI-Modelle zu integrieren.

IT-Infrastrukturexpertin/-experte:
IT-Infrastrukturexpertinnen und -experten erstellen die Architektur des IT-Systems inklusive der notwendigen Schnittstellen, geforderten Rechenressourcen, Kommunikationskanäle etc. Sie stimmen sich eng mit den KI-Expertinnen und -Experten ab.

Datenbeauftragte:
Sie befassen sich mit rechtlichen Aspekten, die die Daten betreffen. Darunter fällt z. B. die Einschätzung, ob die verwendeten Daten personengebunden sind und damit einem gesonderten Schutz unterliegen

Verantwortlichkeiten in PAISE®

Im Folgenden werden die beschriebenen Rollen anhand der RACI Matrix in jede Phase des Vorgehens zur Gesamtsystementwicklung eingeordnet[26]. So erfolgt eine Kategorisierung der jeweiligen Rollenbeteiligung in Responsible (R), Accountable (A), Consulted (C) und Informed (I). In den Phasen Komponentenspezifikaton & Checkpoint-Strategie und Entwicklungszyklus sind alle Rollen bis auf Projektleitung, beauftragendes Unternehmen und Anwendende dazu verpflichtet, ihrer Verantwortung bezüglich der ihre Expertise betreffenden Komponenten nachzukommen.

Verantwortlichkeiten nach RACI
Responsible (R), Accountable (A), Consulted (C), Informed (I)

Rolle / Phase	1	2	3	4	5	6	7
Projektsponsoring/ beauftragendes Unternehmen	R	R	I	-	-	A	R
Projektleitung	R	R	R	A	A	R	I
Domänenexperte/-expertin	I	C	C	R	R	C	-
Sicherheitsbeauftragte	I	C	C	R	R	C	-
Anwendende/Bedienende	I	C	-	C	C	C	-
Automatisierungsingenieur/-ingenieurin	I	C	C	R	R	C	-
Verantwortliche für IT-Sicherheit	I	C	C	R	R	C	-
KI-Experte/-Expertin	I	C	C	R	R	C	-
Softwareentwicklung	I	C	C	R	R	C	-
IT-Infrastrukturexperte/-expertin	I	C	C	R	R	C	-
Datenbeauftragte	I	C	C	R	R	C	-

Legende:
1. Phase: Ziele & Problemverständnis, 2. Phase: Anforderungen & Lösungsansätze, 3. Phase: Funktionale Dekomposition,
4. Phase: Komponentenspezifikation & Checkpoint-Strategie, 5. Phase: Entwicklungszyklus, 6. Phase: Übergabe, 7. Phase: Betrieb & Wartung

Optionale Querverbindungen

Die bisherige Darstellung von PAISE® folgte, abgesehen vom Entwicklungszyklus, einem azyklischen Vorgehen. Im Folgenden werden optionale Querverbindungen betrachtet, die einen Rücksprung, bspw. aus der Betriebsphase, in vorhergehende Phasen erlauben.

Im Folgenden werden mögliche Anwendungsszenarien beleuchtet, welche optionale Querverbindungen zwischen Phasen hervorrufen. Insbesondere geht es hier um die Optimierung und agile Erweiterung, basierend auf dem **Betrieb**.

Optimierung des KI-Subsystems, basierend auf Betriebsdaten:

Durch sorgfältige Tests wird das System bezüglich der spezifizierten Anforderungen und Operationsdomäne validiert. Im Betrieb können dennoch unvorhergesehene Umgebungsbedingungen, Nutzerverhalten oder Systeminteraktionen auftreten, die in der ursprünglichen Spezifikation nicht berücksichtigt waren. Wichtig hierbei sind Sonderfälle, die sich meist aus einer Kombination mehrerer extremer Randbedingungen ergeben. Solche neuen Erkenntnisse sind essenziell für eine kontinuierliche Systemoptimierung. Stehen diese Daten dem Entwicklungsteam zur Verfügung, so lassen sich Optimierungen direkt an einzelnen ***Subsystemen*** vornehmen, die dann in den Betrieb zurückgeführt werden, aber auch neue Releases zur Folge haben können. Diese Querverbindung unterscheidet sich von einer regulären Wartung insofern, dass die reguläre Wartung eingeplant ist und entweder vom Betreiber des ***Gesamtsystems*** oder von einem Serviceteam durchgeführt wird. Bei einer Optimierung wird das Entwicklungsteam wieder tätig, um das generelle Verhalten einzelner ***Komponenten*** zu verbessern.

Agile Erweiterung des Gesamtsystems, basierend auf Betriebsdaten:

Die agile Erweiterung greift den DevOps-Ansatz[26] auf. Wie auch bei der Optimierung können die im Betrieb gewonnenen Erfahrungen Impulse für Weiterentwicklungen geben. Während bei der Optimierung bestehende Funktionalitäten des Produkts verbessert werden, geht die agile Erweiterung einen Schritt weiter. Hier wird die Frage gestellt, welche weiteren Problemstellungen vom Produkt mitgelöst werden können, bzw. um welche Funktionalitäten das Gesamtsystem erweitert werden kann. Daher verlangt die agile Erweiterung einen erneuten Durchlauf des gesamten Vorgehensmodells, wenn vielleicht auch in verkürzter Weise, da auf bestehende Systeme und Dokumentationen aufgebaut werden kann.

26 Der Begriff leitet sich von „Development“ und „Operations“ ab und beschreibt einen Ansatz, der die Zusammenarbeit zwischen Softwareentwicklung und IT-Betrieb verbessern soll. Es wird ein kontinuierlicher Wechsel zwischen Entwicklung und Betrieb vorgesehen, wobei kleine inkrementelle Updates vorgenommen werden und die Erkenntnisse aus deren Betrieb wieder in die Entwicklung zurück gespielt werden.

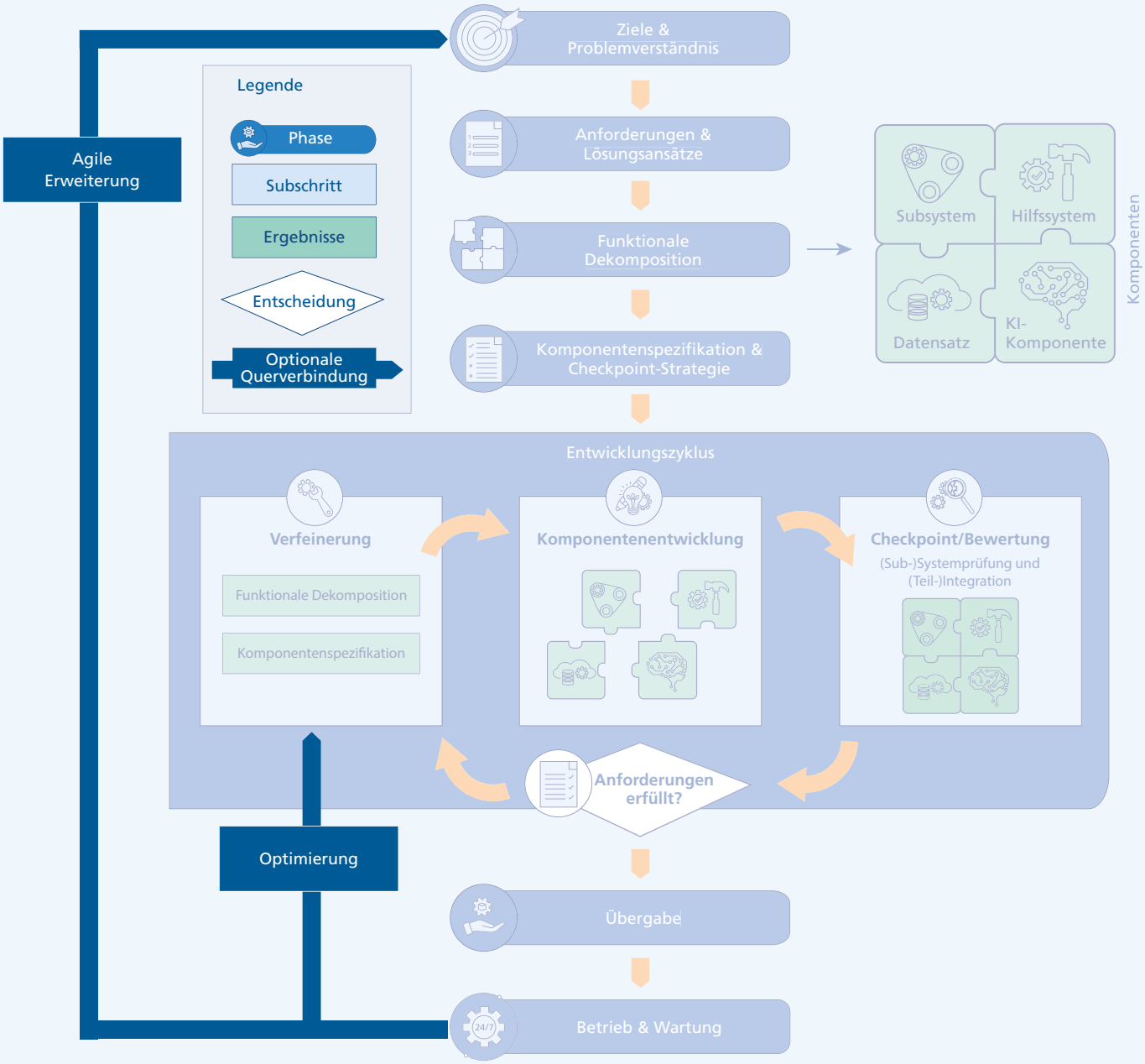


Abbildung 7: Erweiterung von PAISE® mit optionalen Querverbindungen.

Glossar

AI Systems Engineering
Übersetzung für KI-Engineering

Checkpoints
Synchronisationspunkt für die komponentenweise parallel ablaufende Entwicklung in einem iterativen Vorgehen.

Datenquelle
Sub- oder Hilfssystem mit wohldefinierten Schnittstellen, das Daten für den Betrieb und/oder die Entwicklung eines KI-basierten Sub- oder Hilfssystems liefert.

Datensatz
Eine Reihe von inhaltlich zusammenhängenden Daten, die für die Weiterverarbeitung zusammengefasst werden.

Durchgehendes Artefakt
Resultat, das im Verlauf eines Projekts produziert und kontinuierlich verfeinert und angepasst wird.

Funktionale Dekomposition
Zerlegung eines Systems in Subsysteme, die individuelle Funktionen übernehmen

Gesamtsystem
Die Anordnung individueller miteinander interagierender Subsysteme, die zusammen Verhaltensweisen und Funktionen aufweisen, die die individuellen Subsysteme nicht erreichen.

Hilfssystem (engl. Enabling System)
Ein System, das bei Entwicklung und Wartung eines Subsystems benötigt wird, jedoch im ausgelieferten Produkt, also dem Gesamtsystem nicht enthalten ist

Hochrisiko-KI-System
Eine Untergruppe von KI-Systemen, die laut aktuellem Proposal ARTIFICIAL INTELLIGENCE ACT der EU sicherheitskritische Aufgaben übernehmen. Diese Systeme sind einer strengeren Regulierung unterworfen.

KI-basiert
Die Funktionalität wird maßgeblich durch KI beeinflusst. Im Fall von Komponenten kann dies sowohl wegen einer Integration von KI-Verfahren in die Komponente sein als auch aufgrund des Einsatzes von KI-Verfahren, um die Komponenten zu entwickeln und zu warten.

KI-Engineering
adressiert die systematische Entwicklung und den Betrieb von KI-basierten Lösungen als Teil von Systemen, die komplexe Aufgaben erfüllen.

Komponente
Ein Subsystem oder Hilfssystem.

Künstliche Intelligenz (KI)
Die Eigenschaft eines IT-Systems, »menschenähnliche«, intelligente Verhaltensweisen zu zeigen (Deutsches Forschungszentrum für künstliche Intelligenz).

Maschinelle Lernverfahren
Methoden im Bereich des maschinellen Lernens.

Maschinelles Lernen (ML)
Ein Teilgebiet der Künstlichen Intelligenz, das Algorithmen umfasst, die Gesetzmäßigkeiten und Muster in Datensätzen erfassen und daraus Problemlösungen ableiten.

ML-Algorithmus
Ein Algorithmus, der ein maschinelles Lernverfahren umsetzt.

ML-basiert
Die Funktionalität wird maßgeblich durch Maschinelle Lernverfahren beeinflusst. Im Fall von Komponenten kann dies sowohl wegen einer Integration von ML-Verfahren in die Komponente sein als auch aufgrund des Einsatzes von ML-Verfahren, um die Komponenten zu entwickeln und zu warten.

ML-Komponente
Eine Komponente, in der maschinelle Lernverfahren zum Einsatz kommen.

ML-Modell
Abstraktion einiger Aspekte der Realität. Maschinelle Lernverfahren erstellen ein Modell anhand von Daten, um eine Aufgabe zu lösen.

Process Model for AI Systems Engineering (PAISE)
Ein Vorgehensmodell für KI-Engineering.

Prozess
Eine Anzahl an Aktivitäten, die zu einem definierten Resultat führen sollen.

Subsystem
Ein System, das sich hierarchisch in ein Gesamtsystem oder ein übergeordnetes System einordnet.

Systems Engineering
Ein interdisziplinärer Ansatz, um komplexe technische Systeme in großen Projekten zu entwickeln und zu realisieren.

Abkürzungen

AI	Artificial Intelligence
CC-KING	Competence Center KI-Engineering Karlsruhe
KI	Künstliche Intelligenz
ML	maschinelles Lernen/Machine Learning
PAISE	Process Model for AI Systems Engineering

Impressum

CC-KING
Kompetenzzentrum
KI-Engineering

CC-KING ist das Kompetenzzentrum für KI-Engineering der Karlsruher Forschungseinrichtungen Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB (federführend), FZI Forschungszentrum Informatik und Karlsruher Institut für Technologie (KIT). Es stellt eine Verbindung zwischen KI-Spitzenforschung und etablierten Ingenieurdisziplinen dar und soll so den Einsatz von Methoden der Künstlichen Intelligenz und des maschinellen Lernens in der Praxis erleichtern.

Editorin
Dr. Constanze Hasterok, Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB
Fraunhoferstr. 1, 76131 Karlsruhe
constanze.hasterok@iosb.fraunhofer.de

Autoren
Fraunhofer IOSB
Dr. Constanze Hasterok, Dr. Janina Stompe, Dr. Julius Pfrommer, Dr. Thomas Usländer, Jens Ziehn
FZI
Dr. Sebastian Reiter, Michael Weber
KIT
Dr. Till Riedel

CC-KING Konsortialleitung
Dr.-Ing. Thomas Usländer, Fraunhofer IOSB
thomas.uslaender@iosb.fraunhofer.de
Telefon: + 49 721 6091 480

CC-KING Technisch-wissenschaftliche Leitung
Dr.-Ing. Julius Pfrommer, Fraunhofer IOSB
julius.pfrommer@iosb.fraunhofer.de
Telefon: + 49 721 6091 286

Layout & Grafik
Anja Wollfarth M.A., Fraunhofer IOSB
anja.wollfarth@iosb.fraunhofer.de
Telefon: + 49 721 6091 346

Weitere Informationen
https://www.ki-engineering.eu

Gefördert durch:

Baden-Württemberg
MINISTERIUM FÜR WIRTSCHAFT, ARBEIT UND TOURISMUS

in Kooperation mit:

Fraunhofer IOSB
FZI
KIT Karlsruher Institut für Technologie