CC-KING
Competence Center
for AI Systems Engineering

PAISE®

The process model for AI systems engineering

in cooperation with:

FZI

KIT Karlsruhe Institute of Technology

Fraunhofer IOSB

www.ki-engineering.eu

Contents

CC-KING
Competence Center
for AI Systems Engineering

Introduction

The process model for AI systems engineering, PAISE® for short, was developed at the Competence Center for AI Systems Engineering (CC-KING). PAISE® comprises the systematic and standardized development and operation of AI-based system solutions. Approaches from computer science and data-driven modeling are combined with those from traditional engineering disciplines, such as systems engineering.

The discipline of AI systems engineering seeks to integrate methods of ***artificial intelligence (AI)*** from an engineering perspective into the design, development and operation of technical systems consisting of hardware and software in a systematic, predictable and reliable manner. This objective is driven by the considerable potential offered by the use of ***machine learning (ML)*** in particular. In the field of data interpretation, these methods are sometimes able to achieve results (performances) that could not have been achieved with conventional methods despite decades of development work.[1]

In the past, AI was primarily based on conventional algorithms, such as knowledge-based or rule-based systems. These software systems were programmed by human operators and are specifically tailored to individual use cases. Advancements in computer processing power over the past few decades, coupled with increasing availability of data, have facilitated and progressively expanded the use of data-driven processes. Consequently, machine learning as a subcategory of AI algorithms is gaining more and more practical prominence. Simply put, the ***ML algorithm*** programs software to perform a given task. The automatic programming is done by analyzing what are known as training data in order to identify patterns and relationships. Therefore, the functionalities of the created software are largely determined by the composition and quality of the training data.

While this approach results in an immense gain in efficiency in development, it also necessitates a suitable process. Conventional process models require a manually specified and systematically testable system and are only compatible with machine learning methods to a limited extent. At present, the only testing methods available for machine learning are empirical ones — regardless of whether ML is built into the final product[2] or whether ML-based methods are used to develop a conventional final product.[3] As soon as the results of complex ML-based methods influence the suitability of the final product in a critical way, it is necessary to be able to trace this influence back to first principles. The ML-based methods used and the underlying data need to be monitored in a well-founded manner during system development. After an ML-based component has been delivered, it must also be ensured that the product continues to function reliably under changed environmental conditions, for example.

The challenges for project management for which the PAISE® process model offers solutions are presented below.

Notes on text formatting

The definitions of ***terms written in bold type and italics*** can be found in the glossary (p. 32). **Terms in blue** correspond to the phases of PAISE®

1 Deep learning methods are capable of outperforming human subjects when it comes to image interpretation, for example [D. Ciresan, U. Meier, J. Schmidhuber. (2012). Multi-column Deep Neural Networks for Image Classification. IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649.].

2 For example, when decisions learned during operation are made or when the program is modified to an even greater extent, i.e. it continues to learn on the basis of the data during operation.

3 For example, to select suitable materials or design parameters using ML-based methods.

The challenges of developing AI-based systems

The discipline of AI systems engineering[4] evolved from the discipline of systems engineering. While systems engineering methods and techniques are now applied successfully in developing complex technical systems, the use of AI within such systems poses new challenges for the development process.

Systems engineering offers formalized approaches to organizational challenges, such as interdisciplinary collaboration in the development of complex technical systems. Software solutions, such as embedded software on a microcontroller or programmable logic controller, are fundamental to many domains including medical engineering, mechanical and plant engineering and mobility.

Nowadays, for example, process models that were originally used for software development and were transferred to systems engineering are used for the development of complex technical systems. Examples include the waterfall model,[5] the V-model[6] and SCRUM.[7]

Two specific challenges in the development of ***AI-based*** systems have been identified and need to be addressed during the development process:

1. It is often the case that the performance of an AI-based approach cannot be estimated in advance, but must instead be determined empirically.
2. Data-driven methods such as ***machine learning*** require operational data as early as the development stage.

These two aspects are explained in more detail below, along with their respective impact on the technical development process (see ISO/IEC/IEEE 15288:2015). The aim of PAISE® is to provide a solution that meets a high standard of technical quality. Commercial aspects and company-specific processes are not addressed in this document.

1. It is often the case that the performance of an AI-based approach cannot be estimated ahead of time, but must be determined empirically instead.

In a development process based on the waterfall model, a high-level architecture[8] is derived from the requirements and refined step by step. In the context of a certification of critical systems, these steps of derivation and refinement must also be documented in a comprehensible manner.

In many traditional engineering disciplines, it is possible to design a high-level architecture for a system according to requirements without having to test functional aspects using prototypical means. This is made possible with modeling that is based on physical models, empirical values and simulations.[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), (pp. 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 In PAISE®, the high-level architecture corresponds to the system model created in the functional decomposition phase and is refined and adapted during the development cycle phase.

In the case of AI processes, however, it is more difficult to estimate performance theoretically or on the basis of empirical values, since the high-level architecture can only be conclusively determined to a limited extent. Particularly when the performance is governed by rare special cases, implementation details can result in significant differences.

Therefore, when AI-based algorithms are tightly integrated into an ***overall system***, it is often the case that they must be implemented in advance, at least in prototype form. The performance of the implementation in relation to the requirements is then empirically tested before the high-level system architecture can be finalized. This aspect can be taken into account by means of an iterative procedure that is incorporated into the **development cycle** phase in PAISE®. The high-level architecture is iteratively refined and adapted.

In some cases, it is not possible to refine the high-level architecture iteratively because of certain framework conditions, for example because the development of individual ***components*** is to be outsourced to external companies. In this case, preliminary developments of ***AI-based components*** can be useful for estimating the performance, so that the high-level architecture can be determined reliably and iterative adaptation is no longer necessary. Then again, such an approach leads to a loss of flexibility, which can mean that the solution that is technically optimal is ruled out from the outset.

2. Data-driven techniques, such as machine learning, often rely on operational data during development.

Data-driven methods such as machine learning require high-quality data to learn their behavior from. Quality is determined, among other things, by how representative the data used for learning (training data) are for the intended use case. The challenge here is that the training data required during development should, if possible, come from the actual application of the system, which has not yet been fully developed.

In the ML4P (Machine Learning for Production[10]) process model of the Fraunhofer-Gesellschaft, the starting point is an existing production facility into which machine learning methods are to be integrated (brownfield development). In this case, it is possible to obtain high-quality training data from the existing facility. If, on the other hand, an AI-based system is developed from scratch (greenfield development), real data from the system itself are not available until after commissioning. However, since development is supposed to be carried out with the aid of these data, other approaches must be taken into account. Possible alternatives include:

- *Phased implementation:* The data are generated by an existing technical system into which AI is to be integrated.
- *Measurement campaigns:* Data are generated under controlled conditions in dedicated measurement campaigns and series of experiments, possibly under slightly different conditions from those in the use case (laboratory conditions).
- *Simulations:* Data are generated by simulations of the technical system.
- *External data sources:* Data from external providers are incorporated.[11]

Data sources, both for development and operation, must therefore be taken into account from the outset during system development. In PAISE®, datasets are also included in the system model, and time and resources must be allocated for their development. This aspect is taken into account in PAISE® in a separate process called "data provisioning."

9 An example of a supporting tool is the Modelica modeling language, in which multi-physics simulations can be easily combined with predefined libraries of reusable building blocks.

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

11 External data sources are used in the field of image recognition on a frequent basis. Additional image data are either available free of charge or can be purchased from commercial providers.

PAISE® — the process model

PAISE® highlights the development of a product as an overall system that can be broken down into subsystems. PAISE® is characterized by a cyclical progression through refinement steps, component development and checkpoints. This makes it possible to alternate between an explorative approach on the one hand and a goal-oriented approach on the other.

The primary application domains for PAISE® are mobility and production, whereby different application scenarios are addressed. These scenarios include both the one-off customer-specific development and implementation of ***AI-based*** systems and the development of entirely new products that are to be manufactured and sold in multiple versions. The PAISE process model is shown schematically in Figure 1.

The ***subsystems*** of an ***overall system*** provide their own individual functionalities that are independent of one another, have clearly defined interfaces and can also be broken down into smaller parts. ***Machine learning (ML)*** can, on the one hand, be integrated into subsystems directly and, on the other hand, be used in the development of ***enabling systems***. Subsystems and enabling systems can be classified as AI-based and/or as a ***data source***. In addition, ***datasets*** are developed individually. Subsystems, enabling systems and datasets are referred to as ***components*** in PAISE®.

The use of machine learning poses inherent risks during the development process.[12] Such risks arise, for example, from the dependence on data quality, which can lead to a limitation of functionalities, or from unpredictability regarding the performance of AI-based systems. In order to take these risks into account during the development process, the **development cycle** passes through what are known as ***checkpoints***. It is at these checkpoints that (partial) integration and assessment take place in relation to the requirements. For ML components, this entails an evaluation against validation metrics in order to assess the function within the overall system. The outcomes of a checkpoint can prompt refinements as well as adjustments in the solution approaches taken to achieve component functionality. By cycling through **component development**, **checkpoint/assessment** and **refinement,** the maturity level of all components and thus of the overall system is continuously increased.

In parallelized **component development**, it is possible to combine different process models in a hierarchical manner. While preferential procedures can be followed for the development of conventional system components, "ML component development" is defined for ML components and links the development phases in a standardized manner according to engineering standards. Given the fact that sourcing the training data required for developing the ML component can sometimes be very complex, PAISE® addresses the process of **data provisioning** (pp. 19–21) separately.

A waterfall model was chosen for the overall system development procedure in this description. However, it should be emphasized that the seven phases described can also be applied to other models. The underlying "checkpoint-based" concept of PAISE® is still applicable.

PAISE® is a process template that can and should be adapted to the organizational framework conditions at the company. As with any process model, the elements of PAISE® must be applied to each use case in order to derive specific action steps from it. PAISE® focuses on the technical process (see ISO/IEC/EEE 15288:2015) and does not address commercial aspects and company-specific processes.

12 This relates to risks that threaten the project and that can be minimized using iterative approaches in line with Boehm's spiral model. [B. Boehm. (1986). A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 11 (4): 14–24.]

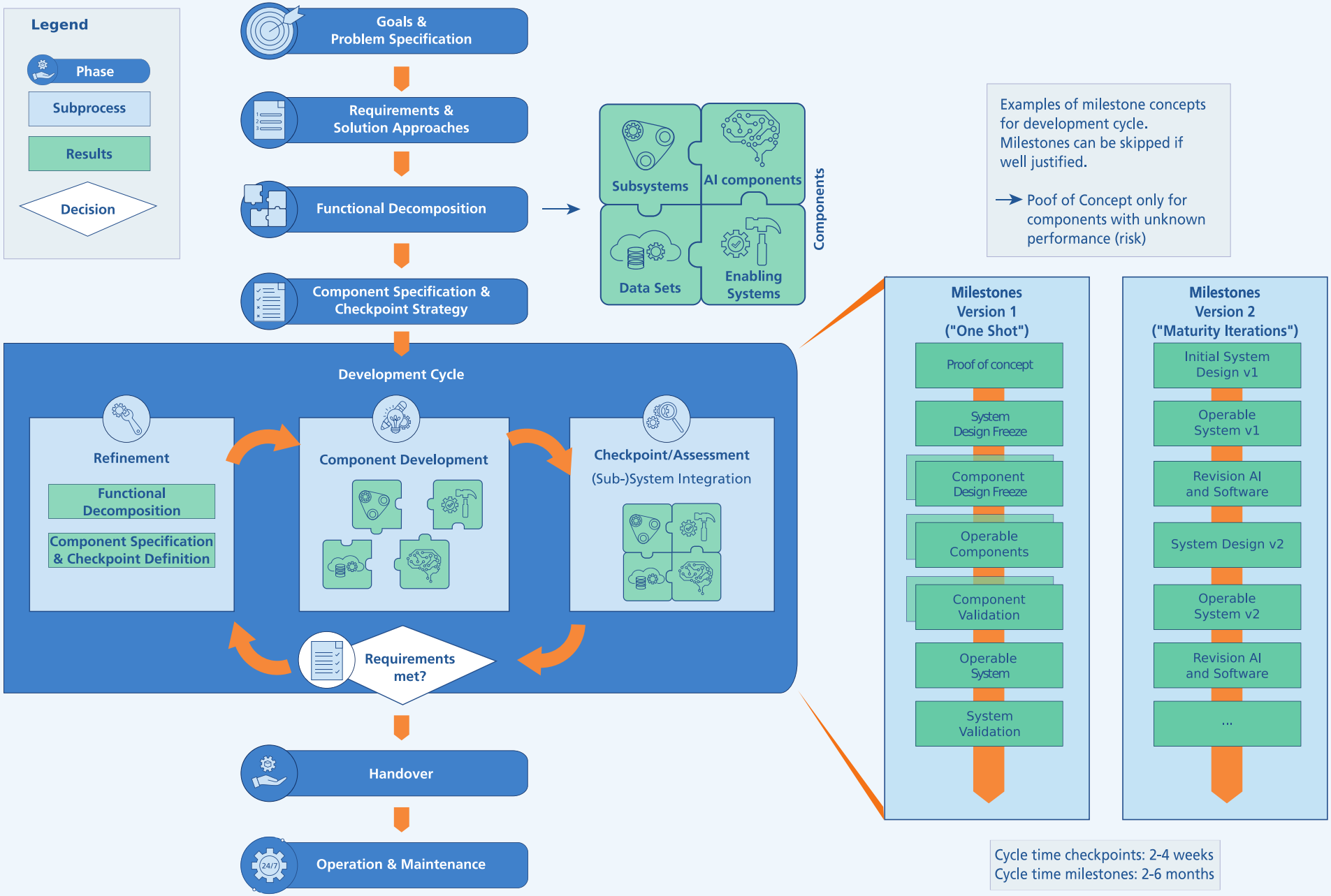


Figure 1: Schematic representation of the PAISE® process model

Permanent artifacts

PAISE® has four permanent artifacts. The artifacts are initiated at specific stages and are continuously expanded and adapted during the course of development.

System model

The system model describes the dependencies between the ***components*** (i.e. ***subsystems***, ***enabling systems*** and ***datasets***) and their interfaces.[13] It is based on the model for technical systems published by Ropohl in 1979 as part of the systems theory of technology (Ropohl, 2009). The system model is initially created in the **functional decomposition** phase and defines the components that are developed with the aid of the individual disciplines in the **development cycle**. An example of such a system model can be found on page 13.

Role allocation

Role allocation defines which responsibilities are required in which phase. This artifact is initiated in **requirements & problem specification** and is employed and adapted in all further phases of the process model. A more detailed description of the aspects of role allocation can be found on pages 28–29.

Documentation for external testing

Documentation for external testing records the characteristics that the ***overall system*** or individual components must fulfill in order to be tested and accepted by external parties (e.g. authorities). The documentation also comprises indicators of this fulfillment or reasoning amassed in the course of development and testing. Typical examples of such guarantees are functional safety and IT security, which can also include aspects of data protection provided by the system. Furthermore, the documentation may include information on explicability, manageability or judicial enforceability.

Data documentation

The data documentation is a description of the data that have been used for the development and testing of the ***AI-based*** components and are decisive in determining their function. Documentation is created during the **development cycle** and continuously expanded and adapted during **operation & maintenance**. Data used should, on the one hand, be categorized in terms of their source (e.g. the public datasets used, collection and annotation methods, measurement methods, environmental conditions), and, on the other hand, be presented in terms of their quality (technical errors, uncertainties, etc.), scope and initial processing (estimation of missing values, enhancement for significantly underrepresented populations). The European Commission's proposal for statutory regulations concerning AI, for example, calls for appropriate qualities.[14]

By archiving all data used, requirements such as those set by the European Commission can be met to some extent. However, this is not always sensible, for example, in the case of particularly large volumes of data or in the case of online learning systems. Methods can then be used to reduce datasets to samples or metadata, for example, or for version changes that may use hash values.

13 This consideration is reflected in numerous approaches to how AI or ML-based systems are currently handled. For example, in the mobility sector, with ISO 21448 "Road vehicles — Safety of the intended functionality," there is a transition from conventional faults in individual components to complex risks of the overall system.

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

Goals & problem specification

In the first phase of PAISE®, the goals to be achieved with the product being developed are defined. In addition, a problem specification is established. This will be developed in greater detail and refined in the subsequent phases.

In this phase, it is particularly important when organizational complexity is high to ensure that the problem specification is consistent across all teams and organizations involved.

The goals can also include business models to be pursued, such as whether a product is to be developed and then mass-marketed, or whether a specific service is to be rolled out.

Example:[15]

The development of a camera-based emergency braking system for passenger cars is commissioned. The system should be able to detect the vehicles driving in front on the highway using a single front camera, estimate their distance and relative speed, and trigger emergency braking if a rear-end collision is imminent. In the first phase, it is established that the system is required to prevent rear-end collisions with a high degree of reliability, that the entire processing chain from camera selection to the implementation of electronic brake control needs to be specified, and that the company commissioning the project does not currently use AI processes or have datasets on which to base the development.

Guiding questions	**Results**
• What is the issue to be solved? • What will be sold to customers? • What is the initial state? • What characterizes the desired end state? • Which data are to be utilized? • Are AI methods already in use? • Can existing ***ML models*** be used in the course of transfer learning?	• Documentation of the answers to and conclusions drawn from the guiding questions in the form of a presentation of the issue and the goals, e.g. a project profile

15 The example of an automotive emergency braking system was chosen here specifically to improve comprehensibility, given the widespread everyday experience in this area.

Requirements & solution approaches

In this phase, the requirements for the overall system are analyzed and possible solution approaches for implementation are defined.

At this point, ideas for possible solution approaches are derived from the product requirements for the first time. This still takes place at the high level. This phase can also yield several possible solution approaches, which are then assessed in terms of their feasibility. In the subsequent development process, work is initially carried out on the approach that seems most realistic, which is then fine-tuned.

If the decision is made to use ***AI-based*** solution approaches, requirements from legal regulations will also need to be taken into account in this phase going forward.[16] Examples of such additional requirements are documentation requirements, such as conceptual decisions regarding procedures for ensuring data sovereignty, data management, data collection and data preparation or the introduction of a risk management system.

Example:

In the course of this phase, it is established that the system is required to work reliably enough to resolve critical situations 99 percent of the time without the driver's intervention. Damage can occur as a result of both false-negative activations (failure to avoid rear-end collision) and false-positive activations (accidents caused by unnecessary emergency braking).

The system should not continue to learn during driving, but should only be updated as needed through manufacturer updates at annual maintenance appointments. The system has a high degree of autonomy over the longitudinal guidance of the vehicle in order to assist drivers if they are not paying attention, and it does not rely on drivers themselves to brake in time. By the same token, drivers are usually unable to cancel false activations in time. The system does not take over the regular task of driving the car, however (distance control, lateral guidance). It is assessed that AI can be used for object recognition and distance estimation, but also that detailed traceability of the quality of the results must be provided for approval.

16 See the European Commission's proposal for legal regulations regarding AI: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206

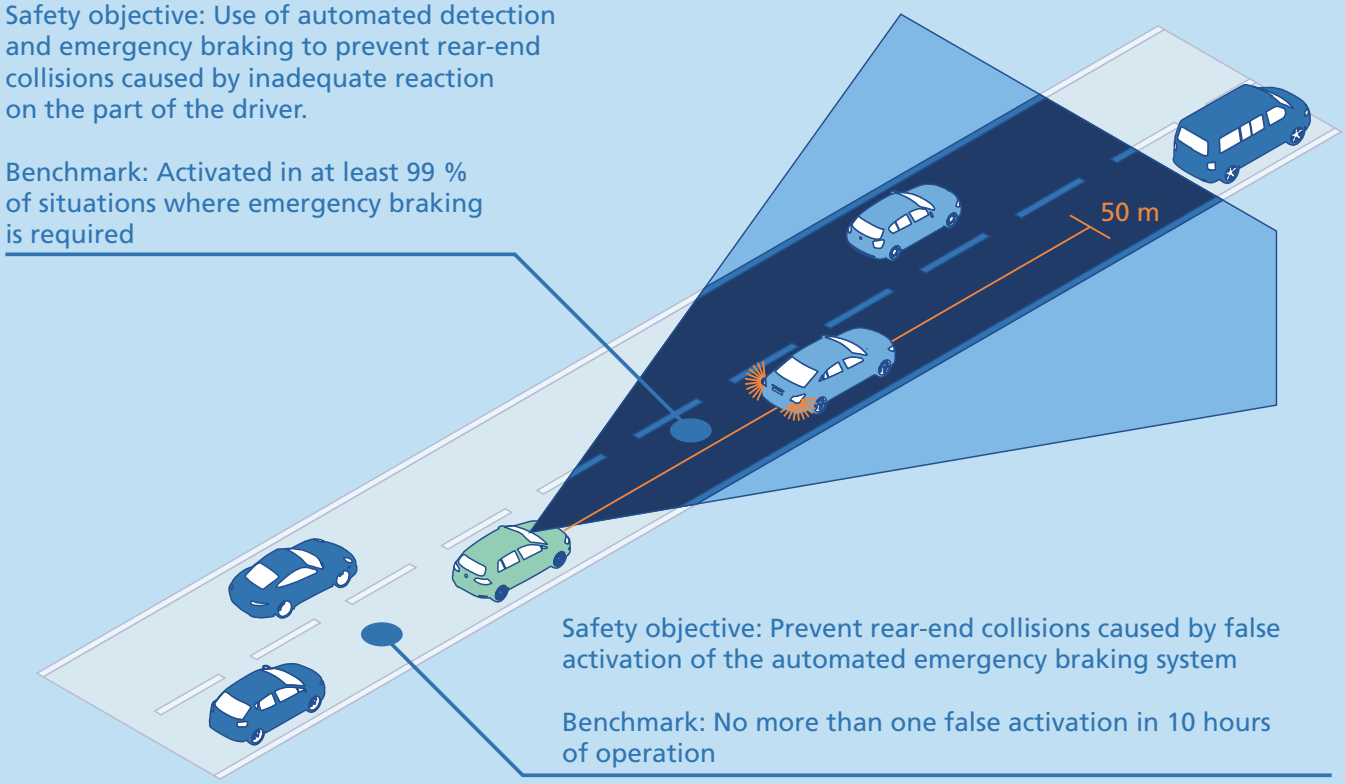
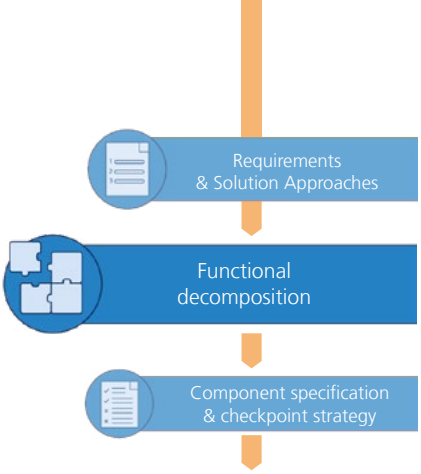


Figure 2: Illustration of the key requirements for the emergency braking system introduced in the example.

Guiding questions	Results
• What risks of harm to people and the environment need to be considered? • What data should the system use as a basis for providing information or making decisions? • Who should have the rights to the data? • At what point should the system learn from the data? • Should AI be able to actively control processes? • For which requirements can AI be used as solution approach? • Is a greater degree of traceability (explicability) of the AI application desired/needed?	• Prioritized system requirements • Requirements for the development process • Selection of the most realistic solution approach for the overall system • Determination of the benefits and primary focus of the AI used in the solution approach and the project • Initial risk assessment of the system • Classification of the system according to degree of autonomy[17] • Contractual regulations for use of data

17 Use cases for (partially) autonomous driving can be divided into five levels according to the SAE J3016 standard by the organization SAE International. A classification into five levels can also be carried out for production-related use cases, as suggested by Plattform Industrie 4.0 [Plattform Industrie 4.0. (2019). Technology Scenario "Artificial Intelligence in Industrie 4.0," working paper].



Functional decomposition

In the third phase, the functions of the overall system are initially broken down into subsystems, resulting in a mostly hierarchical subsystem specification with well-defined interfaces. This is supplemented by the specification of any additional enabling systems required.

The functions defined in the requirements for the ***overall system*** are allocated to ***subsystems***. The granularity of this subdivision depends heavily on the complexity of the system. In addition to this decomposition process, clearly defined interfaces are established.

In addition to considering the overall system, this phase specifies ***enabling systems*** that are not part of the delivered system but are necessary for its development.

It is important to emphasize that the nature of relations between ***components***, i.e. between enabling systems and subsystems, can vary greatly. Energy flows (e.g. electricity, hydraulics, compressed air), information flows (e.g. data), power flows and material flows are examples of such relation types.

Figure 3 shows an example of this kind of decomposition. It should be noted that components can also act as data sources. Data sources are considered to be subsystems or enabling systems that provide data for development and/or for operation and thus have a significant influence on the functionality of the AI components.

The decision as to which subsystem is ***AI-based*** can be made either with the benefit of experience during the initial functional decomposition process or during the **development cycle**, which is the stage earmarked for refinement of the initial decomposition. It is important to note that the decision to use AI is part of the solution approach, not part of the requirements. In addition, the need for additional enabling systems or subsystems may arise during development. Likewise, components can be omitted if it can be shown that they are no longer needed. It is therefore important to mention that the initial functional decomposition is not definitive. It is used for the first iterations of the process and should be adapted in the following ones. The ***functional decomposition*** supports the specific purpose of both the conventional components, such as mechanical and electrical systems, and the AI-based components.

Example:

The decomposition process yields the result shown in Figure 3. In the detector, objects are identified in the images captured by the camera. In the decision maker, an estimation of the distance and relative speed of the objects is made, and an emergency braking decision is made on that basis. If the decision is made to trigger emergency braking, a signal is sent to the brake control system, which monitors the hydraulic mechanical brake trigger. The brake is not part of the overall system supplied, but is connected to the brake control system via a predefined interface. The detector and the decision maker are AI-based subsystems. The mounting position for the camera on the car should be optimized using an AI-based enabling system. In addition, data from the camera, when available, should be stored in an internal database to make training data available for the development of the detector. However, since the camera is not available from the outset, the detector is initially developed on the basis of synthesized camera images and with an externally available dataset of traffic situations called "Cityscapes." The Cityscapes dataset is also used in the development of the decision maker. In the example shown, "Camera," "Internal database," "Synthesis of camera images" and "Database with Cityscapes dataset" are data sources.

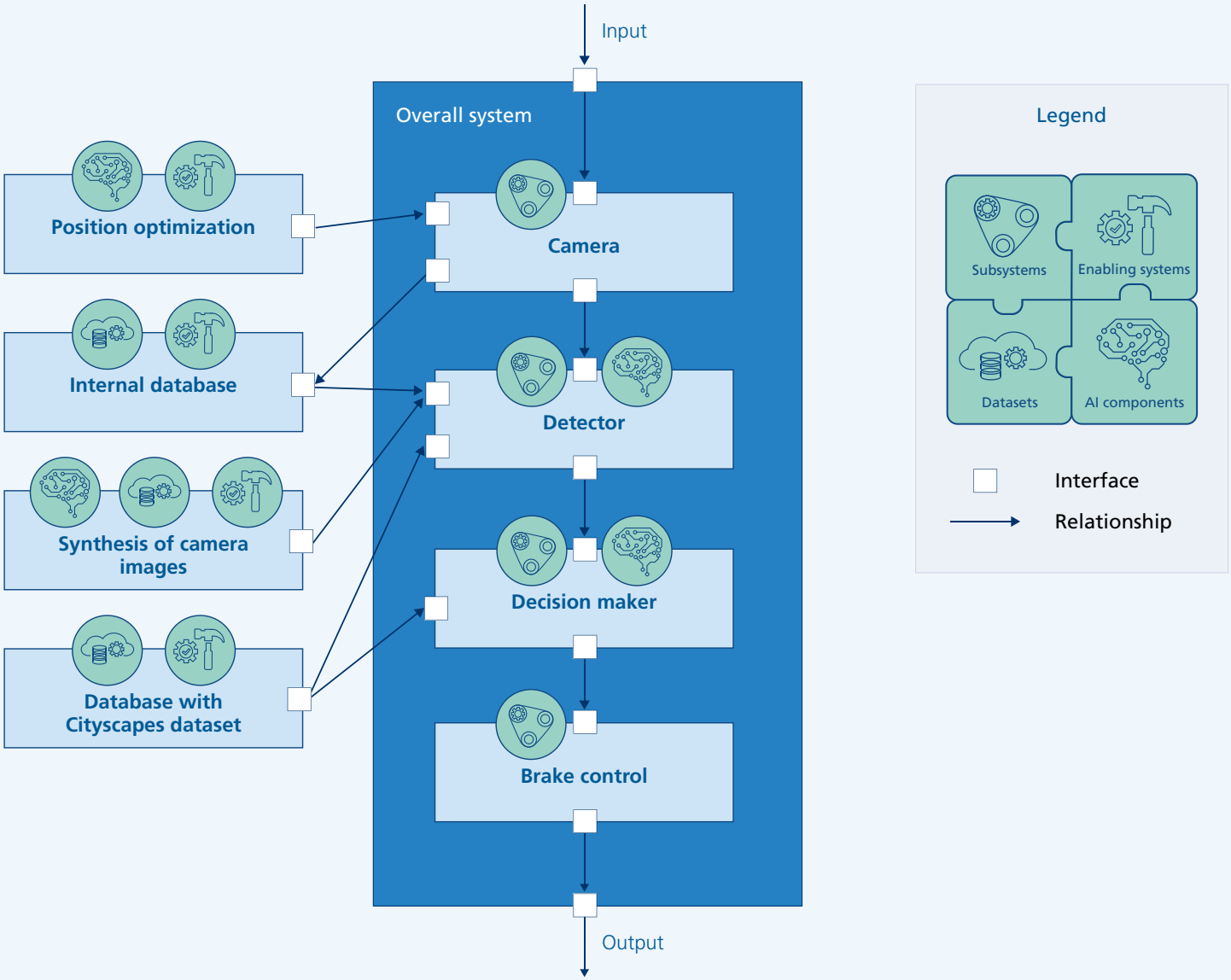
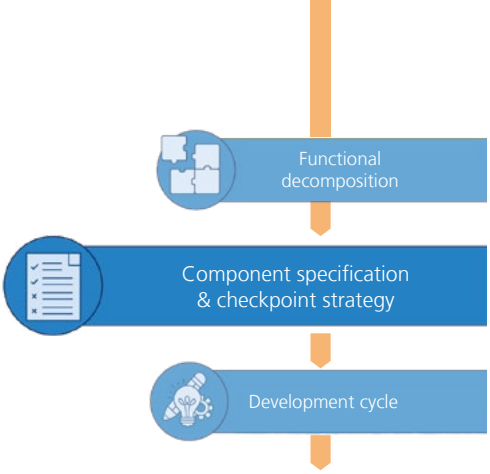


Figure 3: Schematic representation of the system model for an emergency braking system as an example.

Guiding questions	Results
• Which subsystems comprise which functionalities? • What enabling systems are required for subsystem development? • What task is likely to be performed by an AI component? • What data sources are there for training, testing and operating AI components?	• (Hierarchical) system and enabling system specification • Location of data sources and datasets for training, testing and runtime of AI subsystems or AI enabling systems • Definition of interfaces between subsystems and enabling systems • Schematic representation of the development environment



Component specification & checkpoint strategy

In this phase, a preliminary version of the component specification is created, and a strategy for the checkpoints of the concurrent subsystem development is defined.

The component specification is derived in the first instance from the requirements for the ***overall system*** and from the system model. Specific requirements for the ***subsystems*** are drawn up and potential component-specific solution approaches are devised. These solution approaches are reviewed and refined during the **development cycle** phase.

Checkpoints are used to synchronize the development status of all ***components*** and to test the interaction of the subsystems within the overall system. To this end, (partial) integration of the subsystems takes place at this point, together with corresponding verification and validation tests with respect to the requirements for the overall system. Since the development process can be a different one for each component, one can expect the rate of progress to differ between components. Not all components are required to actively participate in each checkpoint.

Different strategies can be used in determining when and according to which criteria a checkpoint takes place. Although traditional milestone planning is possible, and offers many advantages in terms of predictability, a more agile approach is recommended. The difference between checkpoints and formal milestones is that a specific stage of development does not necessarily have to be defined. In terms of checkpoint planning, the following strategies are possible:

- *The feature-based strategy:* A specific feature or requirement is to be implemented by the next checkpoint. This strategy is based on the implementation of what is known as the "user story" in the agile process model SCRUM. It is important that only design decisions that are necessary for implementing the feature and achieving a minimum viable product are made during the development stage. Ultimately, this saves costs when it comes to making decisions.
- *The maturity-based strategy:* A checkpoint is reached whenever a minimum of two subsystems have reached a certain level of maturity. In this case, maturity levels could be:
  - Preliminary analysis (proof of concept)
  - Guarantee of basic functionalities
  - Achievement of performance metrics (KPI)
  - Performance enhancement/optimization
  - Optimization of user friendliness

  Depending on the application, the maturity levels can be further specified and subdivided. They therefore constitute interim goals on the way to the finished product.

- *The time-based strategy:* A checkpoint is reached at regular time intervals, e.g. one week. The challenge here is to select the work packages up to the next checkpoint in a way that makes it possible to (partially) integrate them with all the new features at that point. However, given the fact that not all components have to participate in a checkpoint together, it is also possible to use different time intervals for each component.

Both when pursuing a strategy and when combining different strategies, it is important that the next checkpoint in each case is clearly defined in the refinement step. This includes the issues of which subsystems participate in (partial) integration and when a checkpoint is reached.

Example:

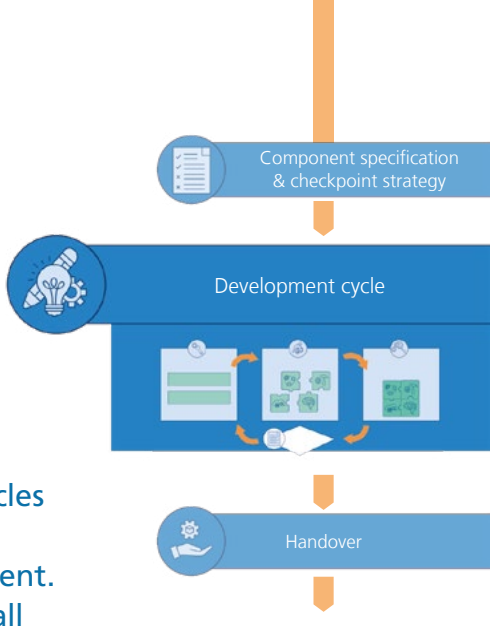
The following table compares the aspects of the maturity-based and feature-based strategies for checkpoints.

Feature-based strategy	Maturity-based strategy
By way of example, 2–3 features to be implemented within the initial development cycles are shown below for each component. A checkpoint takes place as soon as it is possible to integrate one of the developed components.	The following is an exemplary description of a few maturity levels for the overall system using bullet points.
Camera: • Elaboration of suitable specifications (resolution, frame rate, etc.) • Procurement of prototype • Determination of optimum installation position	Proof of Concept: • Test vehicle with camera prototype • Detector and decision maker were developed using Cityscapes dataset • Brake intervention using series AEB interface • Test drives at the test site
Datasets: • Selection of dataset (Cityscapes) • Interface connection to detector • Increase in consistency with target application through synthetic generation of new camera images	Implementation of 70% of the requirements: • Switch to camera target system • No embedded computer in vehicle yet but dedicated computing unit on passenger seat • Use of real-world target vehicle in development stage • Test drives at the test site
Detector: • Comparison with camera specification • Interface connection to decision maker • Verification with real-world camera data	Implementation of 90% of the requirements: • Hardware and software available in target architecture • Performance of emergency braking system still unknown • Real-life traffic tests with safety driver
Decision maker: • Interface connection to detector • Verification against real-world data from detector	Implementation of 100% of the requirements • Fully developed overall system with known performance that meets requirements
Brake control: • Elaboration of suitable specification • Verification of brake implementation on test bench	

When dealing with the example in subsequent phases, the feature-based strategy is chosen. At each refinement step, goals and features that are to be implemented by the next checkpoint are developed for each component. Examples of the initial component specifications for the "Database with Cityscapes Dataset" and "Detector" components are detailed on pages 20 and 24, respectively, in the context of the corresponding procedures during development.

Guiding questions	Results
• Which component-specific solution approaches should be pursued? • What information is available to the AI subsystem as an input (feature vector)? • What quality level does the AI subsystem have to achieve and how is it verified? • What quality of data needs to be available, and at which checkpoint, for AI subsystem development to move forward? How is the quality level verified?	• Documentation of all initial specifications • Documentation of the strategy for the checkpoints

Development cycle



Component development takes place in iterative cycles that continuously increase the maturity of the overall system. The cycles consist of a **refinement** step, a phase of concurrent **component development** and a **checkpoint** that includes a progress assessment. Eventually, the results lead to a decision as to whether the overall system has been completed in accordance with the requirements.

Refinement takes place on the basis of the results of the **checkpoint/assessment**. The solution approach chosen for implementing the respective component specification is elaborated in more detail or, if necessary, varied. A variation of the component-specific solution approaches is sensible in the first cycles of the **development cycle** in order not to exclude any solutions from the outset. In later cycles, work should only be done on the detailed design of the solution approaches in order to continuously increase the maturity of the product. The **refinement** step takes place in an interdisciplinary manner to take into account the dependencies between the ***components***. Subsequently, appropriate adjustments are made with regard to the system model as well as the component specification. In particular, this may involve further decomposition of components, for example to avoid bottlenecks in development, or additional ***components*** may be added if, for example, new ***data sources*** and ***datasets*** are included.

The component specifications that must be met and validated are fundamental to concurrent **component development**. Development takes place for each component according to an individually suitable and domain-specific procedure. In the case of conventional components such as mechanical or electrical ***subsystems***, for example, a ***systems engineering*** procedure can be used. The prerequisite is that this approach can be integrated into the cyclical principle described in this work. PAISE® specifies the processes for ML component development (p. 22) and data provisioning (p. 19).

Once a solution approach has been implemented for a component (whether prototypical or refined), it can be integrated into the surrounding ***overall system*** or the surrounding subsystem and validated and verified as part of integration tests. ***Checkpoints*** synchronize this integration of subsystems. Components can also passively participate in checkpoints if, for example, the development status does not allow for integration. In this case, the development status of the previous cycle is used for this component, simulations are used, or only the interfaces are tested. Finally, an assessment takes place, with corresponding documentation of the development status of all components and the results regarding the overall system. The documentation process should be supported by a versioning process that includes the data used, particularly in the case of ML components.

Overall, the checkpoint serves to focus attention on interdisciplinary cross-sectional aspects. In addition to considering functional safety or costs, this may also include an open discussion on potential ethical conflicts that may arise, for example, from the use of incomplete or biased data. Such aspects are addressed specifically by the data officer and are also incorporated into the data provisioning process.

The "checkpoint-based" cyclical approach outlined above aims at ensuring the continuous improvement of the overall system. It contains three properties that are essential for AI systems engineering from our point of view:

- It takes into account the fact that the (further) development of some components depends on the results of others. For example, the development of an ML component cannot take place effectively until initial data are available. Likewise, the design of a subsystem to be optimized using ML-based methods can be started only after the associated ML-based ***enabling system*** has been developed. The interdependencies of the components arise from the system model. In addition, a time dependency representation

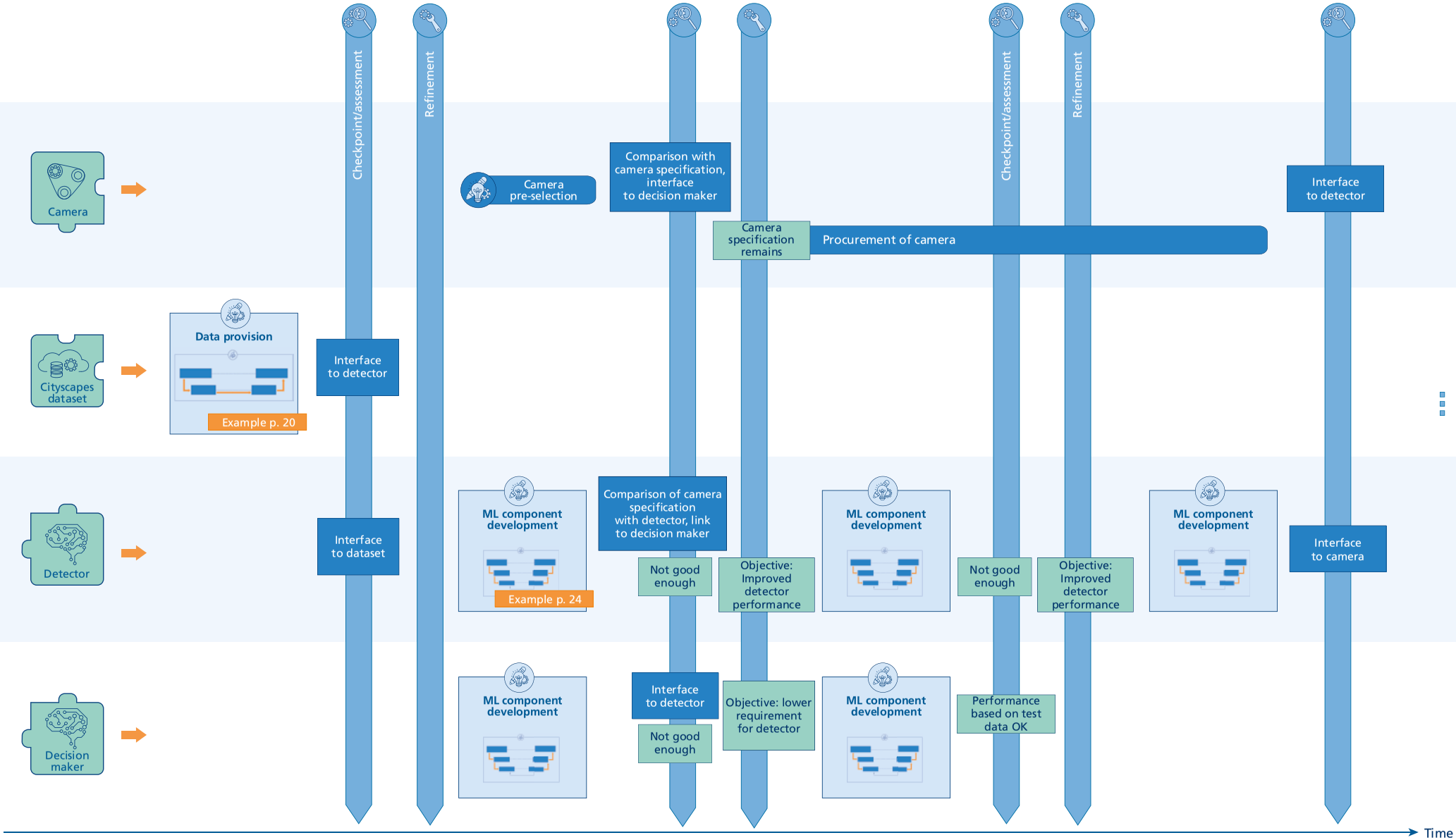


Figure 4: Example of a development cycle

similar to that of a Gantt chart can prove useful here, although it should be noted that in an agile approach it is not possible to use absolute time intervals for planning.

- It enables an explorative approach, which is especially necessary for the development of ML-based components, since it is often impossible to give prior guarantees as to whether all requirements can be met.
- It provides the framework for risk-based development that permits alternative solution approaches, weighs them against each other on the basis of prototypes and evaluates them in terms of their risks. Thus, it is not essential to assume that a component is developed as an ML component. For example, it may turn out that ML-based methods are not suitable and conventional statistical methods should be used.
- Particularly during the initial development cycles, it can be worthwhile to take a rough look at various alternatives, even if they involve a greater risk, as the risk is reduced further and further in later phases.[18] Risk analysis is therefore an essential aspect of this phase.

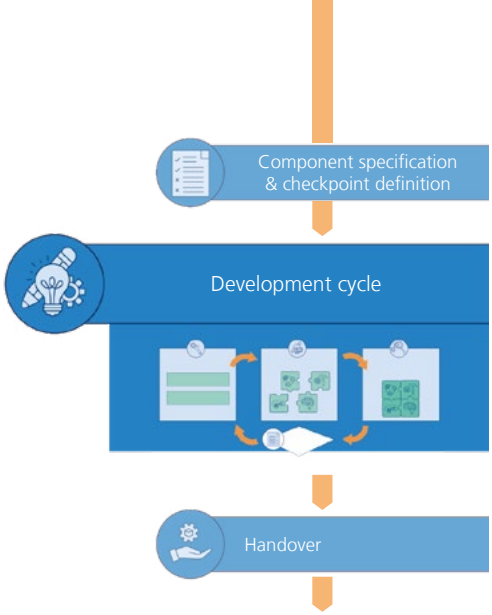
Not all cycles need to be the same length; rather, they can adapt to organizational circumstances.

Example:

All subsystems and enabling systems are developed during this phase. To provide a clear representation in the adjacent figure, we have limited ourselves to the camera, detector and decision maker subsystems, as well as to the Cityscapes dataset that is provided for developing the detector. The individual actions are only shown in an abstract manner and will be explained in more detail with regard to ML component development and data provisioning in the following sections.

At the outset, the Cityscapes dataset is prepared (see detailed example in the following section) and encapsulated as a component with a defined interface. At the first **checkpoint**, the test of the interfaces between the Cityscapes dataset and detector, as well as between the Cityscapes dataset and decision maker, takes place. This is successful, so no adjustments need to be made to the specifications, decomposition or interfaces during the **refinement step**. Furthermore, the development of an initial prototypical version of the detector (see detailed example on p. 22) and decision maker is defined as the objective for the subsequent **component development** process. At the same time, the selection process for the camera subsystem is started.

The camera, detector and decision maker are involved in the second **checkpoint**. The detector and decision maker have been developed prototypically based on indicative specifications. The specified camera parameters are compared with the assumptions of the detector; it is found that the selected parameters are broadly consistent with the training and test data from the Cityscapes dataset. However, the detector does not reach the required level of detection reliability required by the decision maker. In the refinement process, the decision is made to retain the camera specification, but to specify new target values for both the detector and the decision maker: The detector needs to improve its performance, and the decision maker needs to increase its robustness against detection weaknesses, thus lowering its detector performance requirements. At the same time, the camera procurement process begins, and this will run for the duration of the next two cycles. In the course of the subsequent **component development** process, the decision maker achieves the specified target properties based on the previous test data. However, during the integration tests at the **checkpoint**, it emerges that in the interaction between the detector and the decision maker, less than 99% of the critical situations contained in the dataset are actually evaluated as such. In the **refinement** process, the decision is made that the greater potential for optimization lies with the detector and the decision maker is left in its current state for the time being. **Component development** now focuses on the further development of the detector. At the subsequent **checkpoint**, the interaction between the detector and the decision maker is tested again. In addition, the camera that has been delivered in the meantime can be connected to the detector so that initial functional tests can be performed. Subsequent steps include collecting and annotating real data with the target camera system to assess the detector's level of development within the overall system.



Data provisioning

The **data provisioning** process aims to generate, prepare and evaluate training, test and validation datasets. In doing so, requirements regarding the relevance, representativeness and correctness of the data are to be met.[19] The data form the basis for the development and functionality of AI components.

The specification of the data is adjusted in the **refinement** step. These include both ***the data sources***, which can be different for training, testing and runtime, and the requirements for the data themselves.

Requirements may include technical aspects relevant to the AI component's ability to perform its set tasks, such as the data volume, quality, i.e. whether information is missing or incorrect, and representativeness, i.e. whether the training data are representative of the data encountered during runtime. In addition, there are overarching non-technical aspects such as bias in the distribution of data that can lead to unfair decisions (e.g. gender-specific decisions in personnel selection), costs for data acquisition or legal aspects regarding personal data, which in turn may necessitate additional steps such as anonymization or pseudonymization.

The subsequent procedure for data provisioning is based on the V-model.[20] At a higher level, the first step is to define the target metrics against which the data will later be evaluated. The target metrics in this case are derived from the requirements mentioned above.

This is followed by the experiment or data collection. The term "experiment" is used here to refer to data collection under controlled conditions. For example, for anomaly detection, data collection — with representatives of known anomalies and the normal state — can be performed using guided experiments. In the case of supervised learning, this facilitates the labeling process, i.e. assigning target values that are to be predicted by an ML-based algorithm. The data collection step also includes the recording of data under realistic conditions and the collection of artificially generated data, e.g. from simulations, or the augmentation of an existing ***dataset*** using data augmentation techniques. Similarly, a selection of public datasets may be considered at this stage.

The raw data acquired are screened and prepared during the **data preparation** process with regard to the issue presented. The specification is used as a basis for deriving features, i.e. input characteristics, for example. These features form the basis for the correct functionality of the ML component. Feature selection is based primarily on domain or expert knowledge, but significance considerations may also be relevant here.

In addition, the techniques of aggregation of multiple data points, noise removal or even filtering of incomplete data points can be used in this step. Another useful method is that of data imputation, i.e. estimating missing values. For example, if a sensor fails for a brief period, the corresponding feature and the incomplete data points can still be used. Furthermore, multiple features can be combined into a new feature to reduce the scale and thus the complexity of the data points.

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

19 These three aspects are stipulated in the European Commission's proposal for statutory regulations concerning AI [European Commission. (2021). Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonized Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Act, Brussels: 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.]

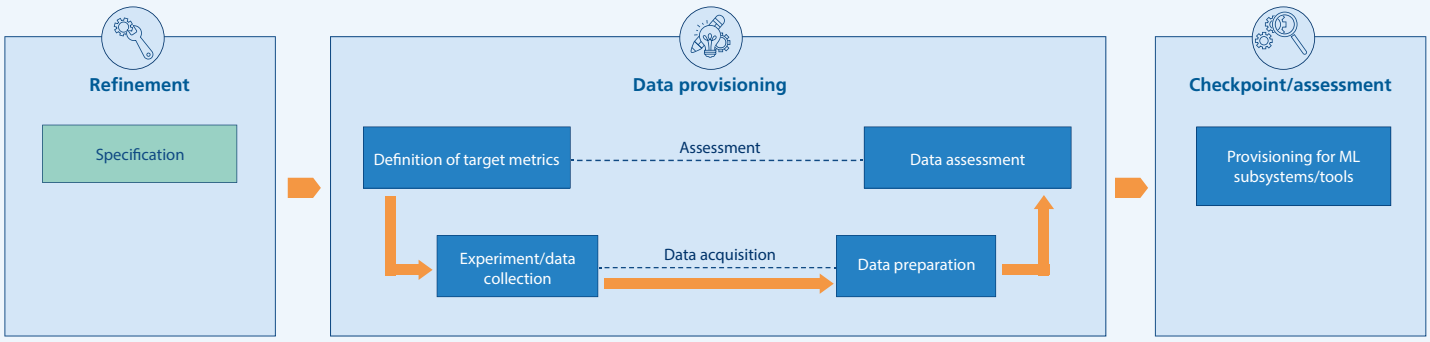


Figure 5: Schematic representation of the data provisioning procedure

In the case of supervised learning in the context of classification, the data are annotated in this step, i.e. the data points are assigned to their respective classes, if this has not already been done automatically when the data were recorded. As part of the development, the AI component can also be furnished with information regarding the relationship between data points and classes. Depending on the requirements, data anonymization or pseudonymization techniques are also applied in this step. While some of the processing steps are only necessary for training, testing and validation data (e.g. annotation), some methods must also be applied to runtime data for consistency reasons (e.g. noise removal, data imputation, combining multiple features, etc.).

Data assessment is the final step in the data provisioning process. This is where the previously defined target metrics relating to the data requirements come into play. In addition to the technical aspect, data assessment also plays an important role from a legal point of view when evidence has to be provided to external organizations. According to the EU proposal, appropriate governance and data management procedures are to be used for this purpose.

The processed and evaluated data will be made available for AI component development upon successful completion of the step.

It should be emphasized that the data preparation process does not necessarily have to be a manual one. In this case in particular, there is huge potential for the automation of individual work steps or sequences of steps.

Example:

For the subsystem developments of the detector and decision maker, vehicle camera image data are required in which the vehicles driving in front are annotated. Since these data are not available at the start of the project, the publicly available Cityscapes dataset is used, which contains actual vehicle camera images with manually annotated objects. As part of the data provisioning process, it must be checked to what extent the dataset is compatible with the target application, i.e. how representative it is (for example, in terms of camera resolution, driving scenarios, etc.). In the following, we will limit ourselves to the aspect of driving scenarios by way of example. For this purpose, the first step is to define target metrics for the traffic situation in the Cityscapes dataset. In the next step, the dataset is downloaded from the provider for evaluation. In the data provisioning process, images that only show people walking or cycling are filtered out, as these are irrelevant for the target use case of highway driving. In the final step of the data assessment process, the entire dataset is assessed once again in terms of the representativeness of the scenarios. It emerges that the scenarios are acceptable for preliminary development, but contain insufficient data from the target use case. Cityscapes places considerable focus on urban scenarios, whereas the target application is primarily aimed at highways. Cityscapes is deemed to be sufficient as a data source for the initial prototype development of the detector. However, later in the project, images will be artificially generated using an enabling system — a vehicle simulation — and then post-processed using AI techniques to make them more realistic. Finally, as soon as it is available, measurement data from public transport will be collected and manually annotated using the actual target camera system. In addition to the issues of representativeness and data quality, there are also data protection issues for this kind of data.

a) Real data from the Cityscapes dataset.

b) Data from the actual target camera system.

c) Training and testing data must also cover the scope of the target challenges.

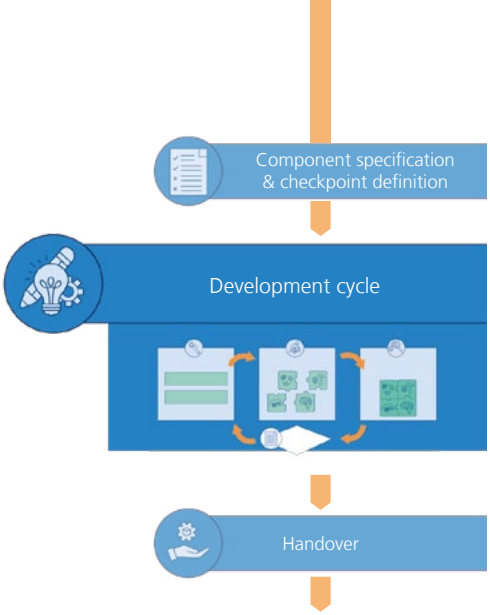
d) Simulated image data from a 3D rendering engine.

e) Simulated image data, enhanced with machine learning techniques such as generative adversarial networks.

f) Simulated data can provide annotations automatically.

Figure 6: Potential data sources and challenges in the sample application.

ML component development



The process of ML component development is based on the V-model[21] as established in the field of software and systems engineering. The aim is close integration with ML enabling systems and data sources to allow iterative integration and validation of results within checkpoints.

This process is intended to encapsulate an ***ML model*** (i.e. the data-driven learned part) in a ***component*** that is specified as precisely as possible. This creates an organizational interface between the traditional discipline of data science and ***systems engineering***.

The starting point is the specification of the component in the context of the surrounding system, which is detailed and adapted iteratively during the **refinement** process. The specification includes, among other things, the ML method (e.g. neural network, decision tree, etc.) as a possible solution approach based on the component requirements. In choosing this, both higher-level requirements for the ***overall system*** (e.g. the traceability of decisions) and direct dependencies on other components (e.g. limited computing resources, availability of data, availability of target variables) are taken into account.

The first step deals with the integration of ***data sources*** for training, testing and validation. It may well happen that the data sources and thus the interfaces are not the same in every cycle, e.g. if data are available in tabular form initially and later in a database.

In developing the test and validation metrics, global cost functions are derived from the component requirements that lend themselves to data-driven assessment. Domain knowledge should be incorporated here in order to be able to test ML components individually, but with reference to their function within the overall system.

The ML procedure is implemented as a concrete ML architecture in the next step, with specified hyperparameters. Examples include defining the number of neurons and layers in artificial neural networks and defining the local cost function and learning rate.

In model training, the ML architecture is transformed into an ML model suitable for the functionality to be fulfilled. Since the results of the learned model depend heavily on the ML architecture, i.e. the chosen hyperparameters, these are varied several times. The aim is to find the hyperparameter configurations that yield the best model on the basis of the local cost function and a validation dataset. In many cases, this hyperparameter optimization can be partially or fully automated using appropriate tools (e.g. auto-ML systems).

In the subsequent model assessment, a test dataset and the previously defined test and validation metrics are used to evaluate the quality of the learned and optimized ML model. Thus, the accuracy and performance of the model can be evaluated in terms of previously unseen data and metrics tailored to the component functionality to be fulfilled.

Model packaging as a component is the final step in ML component development. At this point, the trained and validated ML model is prepared in such a way that it can be deployed on the target platform. While previously the model was validated on the basis of data only, now it is ensured that the model is capable of running on the target platform, for example on a resource-constrained embedded system, where it provides comparable results.

Actual integration of the components into the higher-level system takes place at the **checkpoint**. It is only at this point that tests reveal whether the specification and the derivation of the architecture have been successful and whether the metrics achieved on a component-specific basis also have

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.]

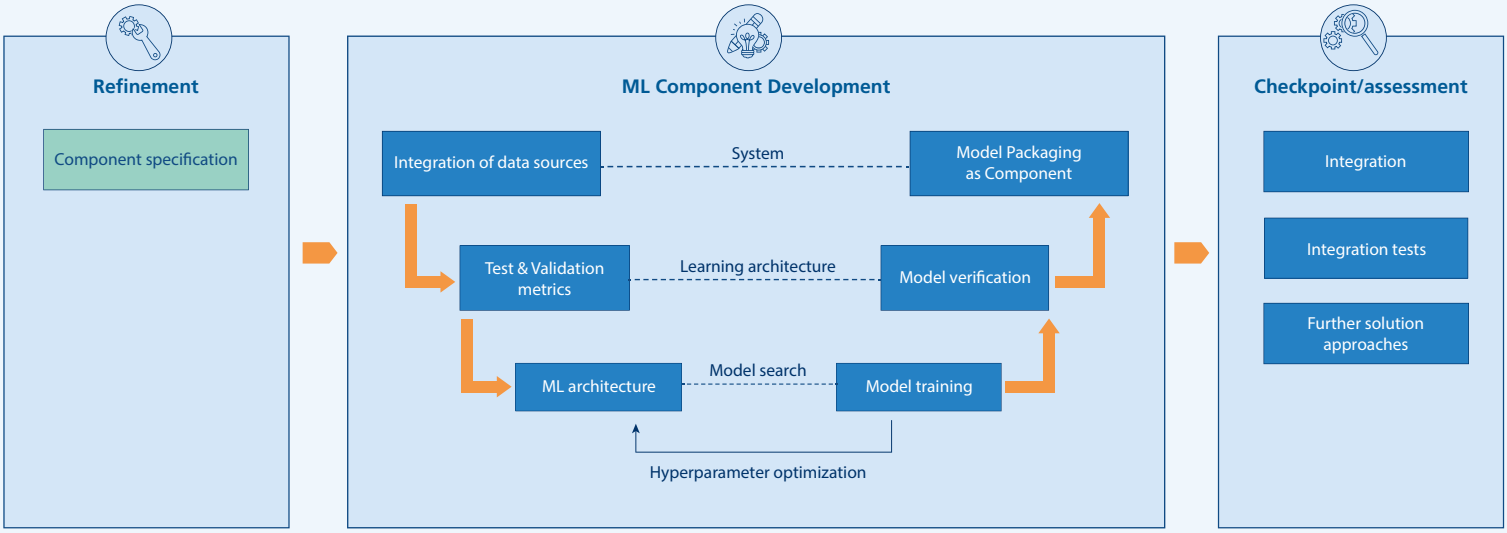


Figure 7: Schematic representation of the ML component development process.

a positive effect on the overall system. If this is not the case, additional cycles are necessary. In these cycles, further solution approaches, such as other ML methods, are tested in coordination with the other components. The goals of additional cycles could also be optimizations that go beyond the requirements, such as increasing the prediction accuracy of the ML system. To control the development process, it is important that metrics are translated into specific estimates of costs and risks within the checkpoint on the basis of the reports created, so that decisions can be made regarding further development.

Guiding questions	Results
• How can requirements be translated into test and validation metrics? • Which cost functions are suitable for model training? • Which ML method is suitable for producing an optimal component in terms of the overall system? • What improvements can be made as a result of changes to other subsystems (e.g. data)? • Do external providers offer AI components that can be used and further developed?	• ML component with clear documentation of requirements, data and tools used • Results from the evaluation of test and validation metrics • Assessment of potential for improvement depending on other subsystems • Continuous testing and monitoring approach for the ML component using test and validation metrics

Example:

For this insight, the focus is on the subsystem development of the detector. The aim is to detect vehicles driving in front in the video stream of a single camera. By way of example, the development is illustrated here using an existing external dataset called "Cityscapes."

In the context of component specification, the following solution approach is rated as promising: The task is to be solved using an artificial neural network for the recognition of objects in individual images, which is run on a small GPU-based computing unit. The neural network receives camera images via an Ethernet interface, and forwards results to the decision maker via a RAM interface, which is also run on the same computing unit. The aim is to achieve at least two detections per vehicle within one second of visibility at a distance of up to 50 meters. In the case of less frequent detections, the tracker is allowed to dismiss the object as a false detection. The error rate must be less than 0.2 percent for this detection task.

During the process of integrating the data interfaces, the technical interfaces in the target system (Ethernet streams of camera images) are converted into operational interfaces (video images in RAM), and the AI-related software portions of the component are detached from the embedded hardware so that they can be developed, trained and tested on a PC or in a computing cluster, for example. The decision is made that the system will initially be developed in the programming language Python and, due to the lack of datasets from the real-world system, will use an existing annotated vehicle dataset, Cityscapes,[22] which is fundamentally similar to the target system.

As part of the development of test and validation metrics, component requirements (two detections per second) are adapted to the requirements of the specified ML process. Since this method only uses a single image evaluation, it is not possible to adopt time periods directly. A frame rate of 25 frames per second is assumed and it is a requirement that each object must be detected at least twice over the course of 25 consecutive frames. Furthermore, this requirement may only be breached for less than 0.2 percent of the objects in the dataset.

In the process of choosing the ML architecture, a "Mask R-CNN" approach[23] is selected that recognizes individual objects based on their image pixel regions. To train this neural network, the cost function must be broken down into individual images and their pixels. So, even though "recognized" vs. "not recognized" is supposed to be evaluated for an entire image, a corresponding cost function is unsuitable for training. It is more practical to "punish" each misclassified pixel in training — and evaluate the performance of the neural network using the common mIoU metric[24]. This is where a major disconnect in requirements occurs: We are training a network that recognizes object outlines as precisely as possible in order to obtain an ML component that recognizes objects as frequently as possible.

Thus, the model is trained on the basis of the mIoU metric using the Cityscapes dataset. If the results are not satisfactory, hyperparameters, such as the number of layers in the neural network, can be adjusted.

In the model assessment process, the ML model is evaluated on the basis of the frame sequences, i.e. object outlines from the Mask-R-CNN result are checked for sufficient size and this metric is applied to image sequences from the Cityscapes dataset to evaluate whether the target of two detections per 25 frames is met, in the Cityscapes dataset at least.

In the context of model packaging as a component, the system is transferred to the GPU computing unit, and the Cityscapes dataset images are imported as a simulation via the real-world Ethernet interface to mimic a real-world video stream. Here, the benchmark of two detections per second is tested on the real-world component platform, but (in this iteration) still without real-world data.

At the checkpoint, the component can now be assessed for compatibility with the components developed concurrently in terms of its specifications and the achieved quality of results.

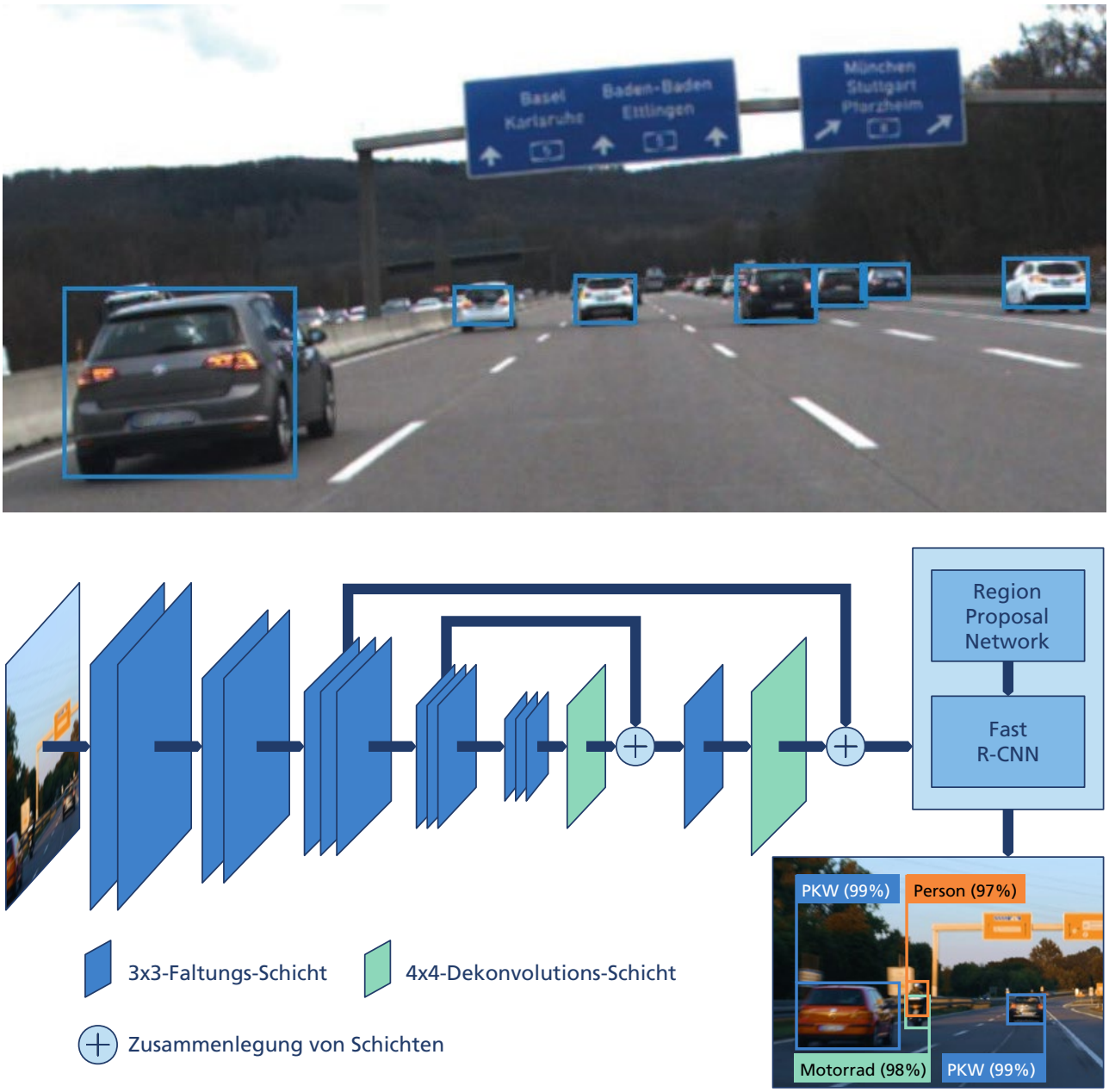


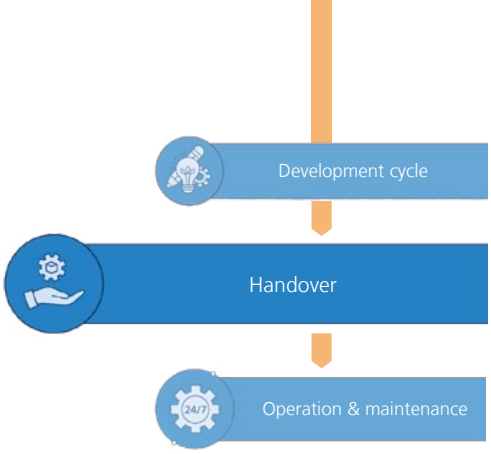
Figure 8: Schematic representation of an R-CNN. Adapted from[25].

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

23 A system based on a neural network used to recognize objects in images. [K. He, G. Gkioxari, P. Dollár, R. Girshick. (2017). Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969.]

24 The "mean intersection over union" metric evaluates the average overlap between the machine-detected object and the real-world known object.

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]



Handover

In this phase, the finished product is transferred from the development team to the organizational units that take care of operations and maintenance. To this end, documentation is prepared for the users and, if applicable, a service team. Specifically for AI systems, issues related to error-proneness and maintenance (retraining) of ML models are addressed.

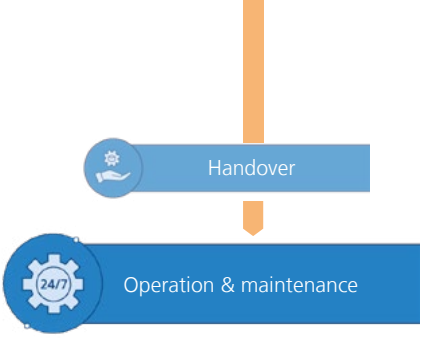
The information required for the **operation** phase is prepared for the users and the service team, e.g. in the form of an operating manual. This includes escalation levels when error states are detected in the system. Especially when using ML-based systems, issues concerning the consistency of the model and the concepts for detecting erroneous models must be clarified conclusively.

Depending on the type of system in question and on its level of autonomy and the assessment of risk to humans and the environment, additional constraints, such as reporting requirements and declarations of conformity, must be addressed in this phase. This specifically concerns ***high-risk AI systems.***

Example:

The system is handed over to the contracted company. The latter analyzes both the design principles and the quality of results on the basis of the artifacts and verifies acceptance. It is specified that ML-based object recognition is based largely on inventory datasets of vehicle images, and may not be able to capture future manifestations (e.g. the possible prevalence of shuttle minibuses, which are not included in the dataset). It is specified that the system monitors itself to a limited extent by running comparisons against simple non-ML processes. Firstly, regular data collection runs are necessary for providing updated datasets for re-testing and re-training, and secondly, vehicle parameters need to be updated, potentially on a yearly basis. The fact that deployment of the system must be registered and regularly checked in the course of maintenance work on account of its nature as a high-risk AI system is communicated to the relevant authorities.

Guiding questions	Results
• How can changes in data distribution that lead to changes in the behavior of ML components be detected? • What criteria should be used as a basis for initiating maintenance work, for example re-training an ML model? • How can the model be monitored for erroneous behavior? • Who is responsible for erroneous behavior if the system continues to learn during operation?	• Operating manual • Documentation for the company's service department • Maintenance concept (including updates to the AI component(s))



Operation & maintenance

In the final operation & maintenance phase, the service and maintenance concept defined in the previous step is implemented. The aim is to ensure full functional capability during operation.

With regard to ***AI subsystems***, this phase includes, in particular, the monitoring and regular review of the ML models. Some ***AI-based*** systems are tested (and possibly also certified) before they are delivered and are no longer modified during the operation phase. In contrast, other AI-based systems are continuously updated using data from operations. There are numerous gradations between these two extremes.

Changes in the data processed during operation can degrade the performance of AI subsystems over time. Such changes can be caused both by latent influencing variables (e.g. temperature, humidity, etc.) and by changes in the intended use (e.g. foreign traffic signs, new material processed in the machine, etc.).

The trigger for updating the AI subsystem can be set both statically and on the basis of data collected during operation. In the latter case, metrics concerning model quality or changes in the distribution underlying the data are used. The collected data are processed in the context of data provisioning (see p. 19) and prepared for the further development of the AI subsystem, which takes place in a subsequent step with the aid of the ML component development procedure (see p. 22). Finally, the updated ***component*** is reintegrated into the ***overall system***, tested and put into operation.

Example:

During operation, the system parameters are checked as part of the annual maintenance intervals, and abnormalities that have occurred during self-monitoring of the system are analyzed. Regular data collection is carried out, and new registrations of other vehicles with unusual shapes (e.g. new types of shuttle minibuses) are investigated specifically to determine whether the braking system achieves a sufficiently high detection rate for these vehicles.

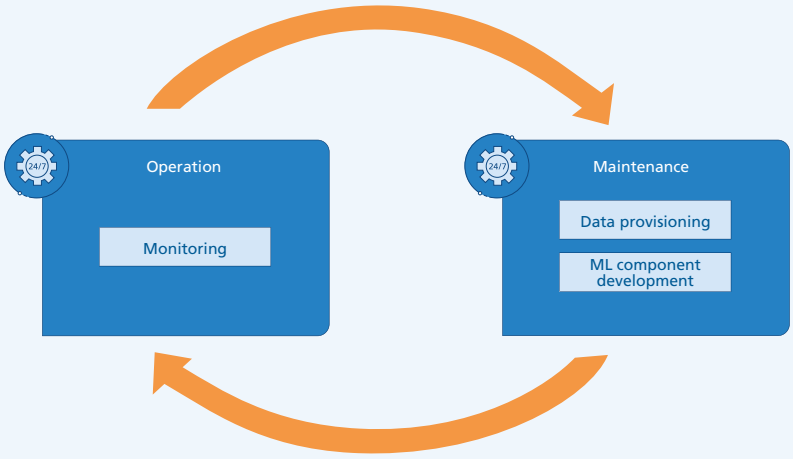


Figure 9: Schematic representation of the substructure of the operation & maintenance phase.

Role allocation

Projects and undertakings in the field of AI systems engineering are usually interdisciplinary and can be very complex. Therefore, a distribution of roles is required that clearly defines competencies and responsibilities.

The roles and expertise that we consider most important are listed below. In the **requirements & solution approaches** phase, the need for each role should be assessed and a person should be assigned. It is also possible for several roles to be assigned to one person.

Project sponsor/commissioning company: The project sponsor or the commissioning company approves the budget for the project and specifies the assignment to be worked on. From this point onwards, the framework conditions, goals and requirements are defined and regular information on the project status is requested.

Project management: Project management organizes and structures the project. It establishes or requests the resources needed to meet the goals and maintain contact with the project sponsor. Project management is also responsible for mitigating any significant organizational complexity by creating teams, assigning responsibilities and ensuring communication to make the project a success.

Domain experts: Domain experts demonstrate a high level of understanding of the application domain that the data for AI component development come from. The expert has a particularly active role to play in applications with a high degree of relevance to physical reality.

Safety officers: They deal with the issue of functional safety in highly critical systems, prepare risk assessments and assessments and are responsible for ensuring that the required critical limits are met.

Other tasks performed by safety officers include assessing the risks that arise in the event of an application failure and initiating appropriate preventive measures.

Users/operators: Users assess the practicality of the application and point out deficiencies, and therefore have an advisory function. In particular, they have experience in the human-application interface.

Automation engineers: Automation engineers ensure that the commands generated by the software are implemented by deploying the process control system. This expertise is required more than ever when there is a high degree of autonomy, since decisions made by AI-based ***components*** must be passed on automatically to the actuators.

AI experts: AI experts create data-driven models and validate and verify them. They have a basic understanding of the relationships underlying the data.

IT security officer: The purpose of this role is to ensure the "confidentiality of data" and "integrity of the application." If ***AI-based*** processes are to be used, data will need to be processed, both in the development and in the operation of the system. According to GDPR, personal data in particular must be safeguarded against misuse, but internal company data can also become a target for hackers. Ensuring the invariability of an application's function is an integral aspect of preserving the application's integrity. This is about preventing a situation where the application is used for purposes other than those for which it was designed, for example, teaching the wrong things to a continuously learning system.

Software development: Software developers create the software architecture, coordinate the development process and carry it out. Throughout the process, they work closely with AI experts to integrate the AI models that they create.

IT infrastructure experts: IT infrastructure experts create the architecture of the IT system, including the necessary interfaces, required computing resources, communication channels, etc. They liaise closely with the AI experts.

Data officers: They deal with legislative issues that affect the data. This includes, for example, assessing whether the data used are personal and therefore subject to specific protection.

Responsibilities in PAISE®

In the following, the roles described are classified according to the RACI matrix in each phase of the overall system development process[26]. Thus, the respective role is categorized as Responsible (R), Accountable (A), Consulted (C) or Informed (I).
In the component specification & checkpoint strategy and the development cycle phases, all roles except project management, commissioning company and users are required to fulfill their responsibilities with regard to the components that relate to their expertise.

Responsibilities according to RACI Responsible (R), Accountable (A), Consulted (C), Informed (I)

Role/phase	1	2	3	4	5	6	7
Project sponsor/ commissioning company	R	R	I	-	-	A	R
Project management	R	R	R	A	A	R	I
Domain expert	I	C	C	R	R	C	-
Safety officer	I	C	C	R	R	C	-
Users/operators	I	C	-	C	C	C	-
Automation engineer	I	C	C	R	R	C	-
IT security officers	I	C	C	R	R	C	-
AI expert	I	C	C	R	R	C	-
Software development	I	C	C	R	R	C	-
IT infrastructure expert	I	C	C	R	R	C	-
Data officers	I	C	C	R	R	C	-

Legend:

1st phase: Goals & problem specification, 2nd phase: Requirements & solution approaches, 3rd phase: Functional decomposition , 4th phase: Component specification & checkpoint strategy, 5th phase: Development cycle, 6th phase: Handover, 7th phase: Operation & maintenance

Optional links

The previous description of PAISE® followed an acyclic approach, with the exception of the development cycle. In the following, optional links are discussed. These enable you to leave a phase, for example, the operating phase, and return to previous phases.

In the following, potential application scenarios that generate optional links between phases are explored. In particular, this relates to optimization and agile enhancement based on **operation**.

Optimization of the AI subsystem, based on operational data:

The system is validated with respect to the specified requirements and operation domain by means of careful testing. However, unforeseen environmental conditions, user behavior or system interactions that were not considered in the original specification may occur during operation. Special cases that usually result from a combination of several extreme boundary conditions are important in this respect. These new findings are essential for continuous system optimization. If these data are available to the development team, individual ***subsystems*** can be optimized. The updates can be fed back directly into operations or result in new releases. This link differs from regular maintenance in that regular maintenance is scheduled and performed either by the operator of the ***overall system*** or by a service team. During optimization, the development team takes action again to improve the general behavior of individual ***components***.

Agile enhancement of the overall system, based on operational data:

Agile enhancement makes use of the DevOps approach.[26] As with optimization, the experience gained during operation can provide impetus for further developments. While optimization improves existing product functions, agile enhancement goes one step further. At this point, we ask ourselves which other problems can be solved by the product, and which functionalities can be added to the overall system. For that reason, agile enhancement requires a re-run of the entire process model, albeit perhaps in a simplified manner, since existing systems and documentation can be used as a basis.

26 The term is derived from “development” and “operations” and describes an approach that is intended to improve collaboration between software development and IT operations. A continuous transition between development and operations is envisioned, with small incremental updates being made and lessons learned from their operations being fed back into development.

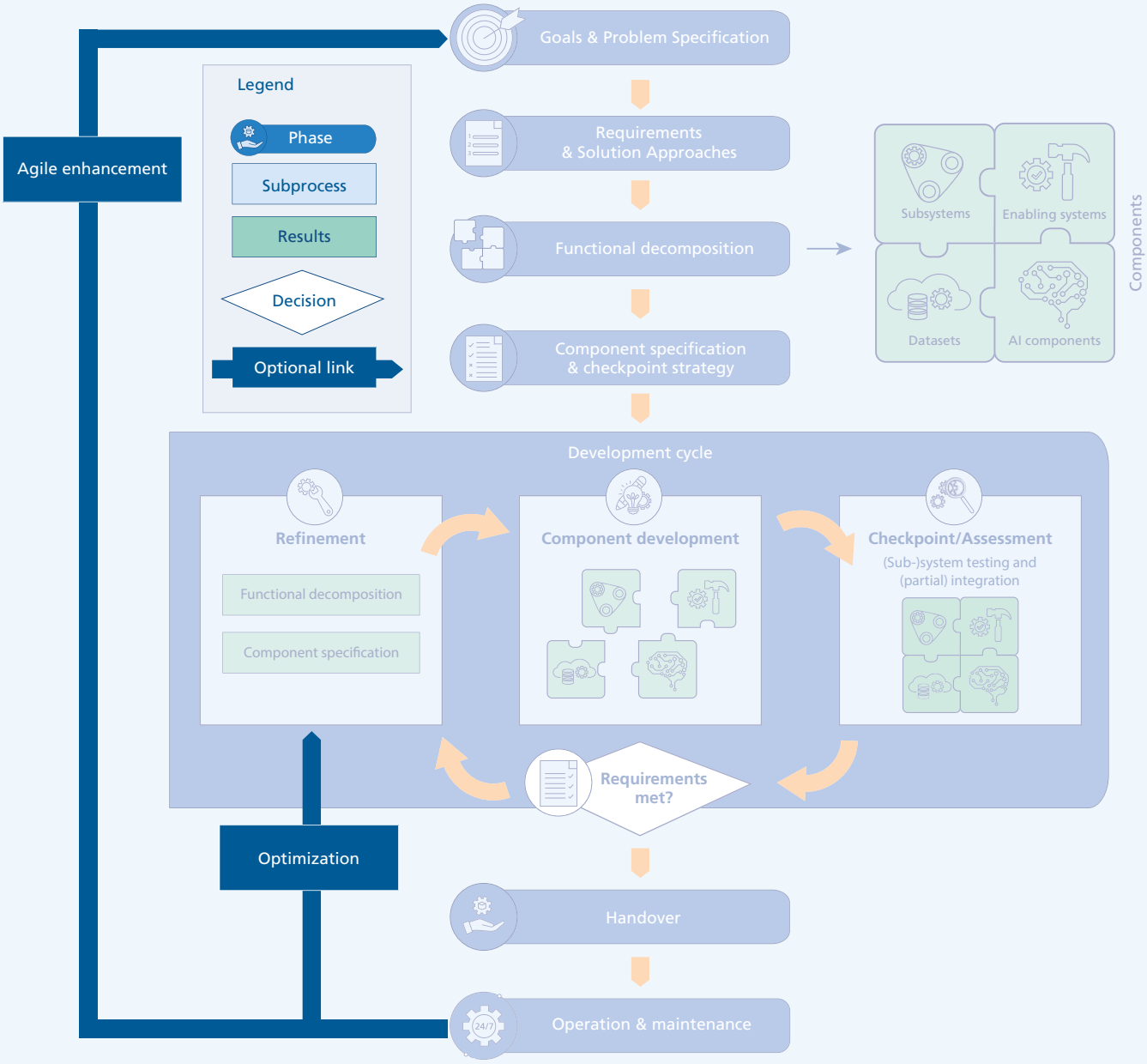


Figure 7: Enhancement of PAISE® with optional links.

Glossary

AI Systems Engineering
Translation of KI-Engineering

Checkpoints
Synchronization point for component-by-component concurrent development in an iterative approach.

Data source
Subsystem or enabling system with well-defined interfaces that provides data for the operation and/or development of an AI-based subsystem or enabling system.

Dataset
A set of data that are related in terms of content and are grouped together for further processing.

Permanent artifact
A result that is produced in the course of a project and is continuously refined and adapted.

Functional decomposition
The decomposition of a system into subsystems that perform individual functions.

Overall system
The arrangement of individual interacting subsystems that together exhibit behaviors and functions that the individual subsystems do not achieve.

Enabling system
A system that is required during the development and maintenance of a subsystem, but is not included in the delivered product, i.e. the overall system.

High-risk AI system
A subset of AI systems that perform safety-critical tasks according to the EU's current ARTIFICIAL INTELLIGENCE ACT proposal. These systems are subject to stricter regulation.

AI-based
AI has a significant influence on functionality. In the case of components, this can be due to the integration of AI methods in the component as well as the use of AI methods to develop and maintain the components.

AI Systems Engineering
Addresses the systematic development and operation of AI-based solutions as part of systems that perform complex tasks. Translates the term KI-Engineering.

Component
A subsystem or enabling system.

Artificial Intelligence (AI)
The property of an IT system that exhibits intelligent behavior akin to that of a human (German Research Center for Artificial Intelligence).

Machine learning processes
Methods in the field of machine learning.

Machine learning (ML)
A branch of artificial intelligence that involves algorithms that detect regularities and patterns in datasets and use them to derive solutions to problems.

ML algorithm
An algorithm that implements a machine learning procedure.

ML-based
Machine learning processes have a significant influence on functionality. In the case of components, this can be due to the integration of ML processes in the component as well as the use of ML processes to develop and maintain the components.

ML component
A component in which machine learning techniques are used.

ML model
An abstraction of some aspects of reality. Machine learning processes create a model based on data in order to solve a task.

Process Model for AI Systems Engineering (PAISE)
A process model for AI systems engineering.

Process
A number of activities that are intended to lead to a defined result.

Subsystem
A system that fits hierarchically into an overall system or a higher-level system.

Systems Engineering
An interdisciplinary approach for developing and implementing complex technical systems in large projects.

Acronyms

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

Publishing notes

CC-KING
Competence Center
for AI Systems Engineering

CC-KING is the Competence Center for AI Systems Engineering of the following Karlsruhe-based research institutions: the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB (in charge), the FZI Research Center for Information Technology and the Karlsruhe Institute of Technology (KIT). CC-KING is the interface between cutting-edge AI research and established engineering disciplines, and thus aims to facilitate the application of AI and machine learning methods in practice.

Editor
Dr. Constanze Hasterok, Fraunhofer IOSB
Fraunhoferstr. 1, 76131 Karlsruhe, Germany
constanze.hasterok@iosb.fraunhofer.de

Authors
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 consortium management
Dr. Thomas Usländer, Fraunhofer IOSB
thomas.uslaender@iosb.fraunhofer.de
Phone: + 49 721 6091 480

CC-KING technical and scientific management
Dr. Julius Pfrommer, Fraunhofer IOSB
julius.pfrommer@iosb.fraunhofer.de
Phone: + 49 721 6091 286

Layout and graphics
Anja Wollfarth M.A., Fraunhofer IOSB
anja.wollfarth@iosb.fraunhofer.de
Phone: + 49 721 6091 346

More information
https://www.ki-engineering.eu

Fraunhofer IOSB is a legally non-independent institute of the Fraunhofer-Gesellschaft e. V., Munich.

Funded by:

Baden-Württemberg
MINISTRY FOR ECONOMICS, LABOR AND TOURISM

in cooperation with:

Fraunhofer IOSB

FZI

KIT Karlsruhe Institute of Technology