Survey

Constanze Hasterok* and Janina Stompe

PAISE® – process model for AI systems engineering

PAISE® – das Vorgehensmodell für KI-Engineering

https://doi.org/10.1515/auto-2022-0020
Received February 15, 2022; accepted May 25, 2022

Abstract: The application of artificial-intelligence-(AI)-based methods within the context of complex systems poses new challenges within the product life cycle. The process model for AI systems engineering, PAISE®, addresses these challenges by combining approaches from the disciplines of systems engineering, software development and data science. The general approach builds on a component-wise development of the overall system including an AI component. This allows domain specific development processes to be parallelized. At the same time, component dependencies are tested within interdisciplinary checkpoints, thus resulting in a refinement of component specifications.

Keywords: AI systems engineering, artificial intelligence, systems engineering, process model, machine learning

Zusammenfassung: Die Anwendung von Methoden der Künstlichen Intelligenz (KI) im Kontext komplexer Systeme stellt neue Herausforderungen an den Produktlebenszyklus. Das Vorgehensmodell für KI-Engineering, PAISE®, adressiert diese Herausforderungen durch die Kombination von Ansätzen aus den Disziplinen Systems Engineering, Softwareentwicklung und Data Science. Der allgemeine Ansatz basiert auf einer komponentenweisen Entwicklung des Gesamtsystems einschließlich einer KI-Komponente. Dadurch können domänenspezifische Entwicklungsprozesse parallelisiert werden. Gleichzeitig werden die Abhängigkeiten der Komponenten untereinander in interdisziplinären Checkpoints getestet. Dadurch werden die Komponentenspezifikationen Schritt für Schritt verfeinert.

Schlagwörter: KI-Engineering, künstliche Intelligenz, Systems Engineering, Vorgehensmodell, maschinelles Lernen

***Corresponding author: Constanze Hasterok,** Fraunhofer IOSB, Karlsruhe, Germany, e-mail: constanze.hasterok@iosb.fraunhofer.de
Janina Stompe, Fraunhofer IOSB, Karlsruhe, Germany, e-mail: janina.stompe@iosb.fraunhofer.de

1 Introduction

Due to the progress in computing power over the last decades and the increasing availability of data, the use of data-driven methods is favored and expanded in many industrial domains. Therefore, machine learning (ML) algorithms are advancing to the practical forefront as a subset of artificial intelligence (AI). The ML algorithm programs a software for a given use case by analyzing so-called training data and identifying patterns and correlations. The functions of the created software are therefore largely determined by the training data.

The importance of finding a systematic approach for the development of intelligent systems making use of novel AI and machine learning methods is widely recognized in Germany and Europe – in industrial development (e. g., in the projects of the "AI family" of the VDA flagship initiative), in standardization (e. g., in the AI standardization roadmap of DIN and DKE [1]), and by institutions such as the European Commission. In addition to the great potential of AI, the European Commission sees the need for regulations as described in the "EU AI Act" [5]. Legal foundations should protect markets and the public sector as well as people's security and fundamental rights. This goal of creating trustworthy AI applications in Europe requires a high level of systematic and interdisciplinary approaches.

Classical procedure models assume a systematically testable system and are therefore only conditionally compatible with the use of ML methods. Currently, systems that apply ML methods can only be tested by empirical methods, regardless of whether ML is built into the final system itself or whether ML methods are used to derive design specifications for the final product. In both cases, the results of complex ML-based methods can influence the functionality of the end product in a critical way. Therefore, it is necessary to be able to trace this influence back to principles of the ML-based methods used and the underlying data.

The above explanations can be summarized in the following two challenges for the use of ML methods in the development process of complex technical systems:

1. The ML component functionalities crucially depend on data from operating components and
2. the performance of ML components is not predictable in advance.

Popular and well-established process models like the Waterfall Model, Scrum, Crisp-DM and the V-Model do not address these challenges in a consequent way.

The Waterfall Model, originally defined for the domain of software development in 1970, consists of a fixed number of phases that are run through in a predefined sequence with clearly pre-defined results [10]. While this process model supports good time planning during development, it lacks iterative elements that allow an explorative approach. Therefore, challenge 2 is not addressed.

The V-Model, which was designed in 1979 for software development as well, is nowadays widely used in systems engineering. In order to ensure product quality, design phases are characterized with different levels of detail and are accompanied by respective test phases. In contrast to the Waterfall Model, the V-Model allows for an iterative approach since a design phase can be repeated if the test phase at the corresponding level of detail fails [9], [6].

The process model Scrum originates from the agile software development approach and is an iterative model as well. It was designed in 2002 [7] and development cycles aim at fulfilling all tasks from a backlog that is refilled from a requirement list after every cycle. The essential ideas of agile software development are the following main value preferences: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contact negotiation and responding to change over following a plan [2]. It is these premises that have been heavily incorporated into the field of AI development and the discipline of data science. Software engineering has thus departed from the waterfall-like and V-Model approaches. However, particularly agile development contradicts the needs of specification- and documentation-driven development of complex technical systems. Also the V-Model lacks a description of how to synchronize the parallel development of data-driven software and hardware components and ensure compatibility. For ML-components compatibility severely depends on the quality of the data that was used during the component's development. Thus, challenge 1 is met neither by the V-Model nor Scrum.

The Cross-industry standard process for data mining (Crisp-DM) was designed in 1996 for data mining projects [4]. The process focuses primarily on the understanding, exploration and modelling of the data. The subsequent data evaluation is matched to the business understanding. The results decide on whether the modeling cycle is started again or whether the model is deployed. Crisp-DM assumes presence of data and does not consider data sources, i. e., system components that can influence the data and therefore the model. Hence, the process model lacks the capability to describe the interplay between data quality and the functionalities of the ML-component and therefore does not address challenge 1.

Especially in the area of software-intensive systems of systems (SISOS), there has been a return to more structured process models since the 2010s. Standards such as ISO 12207 and ISO/IEC 15288 are paving the way for the development of increasingly complex systems. The goal is to make AI engineering integrable into very large development contexts in particular. This is where the discipline of AI systems engineering and the associated **P**rocess model for **AI** **S**ystems **E**ngineering (PAISE®) aim to make a significant contribution.

The standard ISO/IEC 15288 describes the life cycle processes of a system developed according to the established disciplines of systems engineering and software engineering. These processes are equally important for AI engineering. As a process model for AI systems engineering, PAISE® is seen as an example of a procedure that picks up the processes from the standard, focuses on the technical processes and highlights the special features that arise through the use of AI.

The main application domains, that originally motivated PAISE®, are mobility and production. However, PAISE® is in principle domain-independent and may be used in a wide range of engineering disciplines. Various usage scenarios include the customer-specific development and implementation of AI-based systems, as well as a completely new development of AI-based products that are manufactured in series.

In the following, we will focus on the foundations of PAISE®. More detailed explanations together with instructive examples can be found in [3].

2 PAISE®

The Process Model for AI Systems Engineering views the development of a product as a system that can be decomposed into subsystems which can be either software (e. g., ML algorithms) or hardware (e. g., mechanical parts). The subsystems provide individual functionalities, have

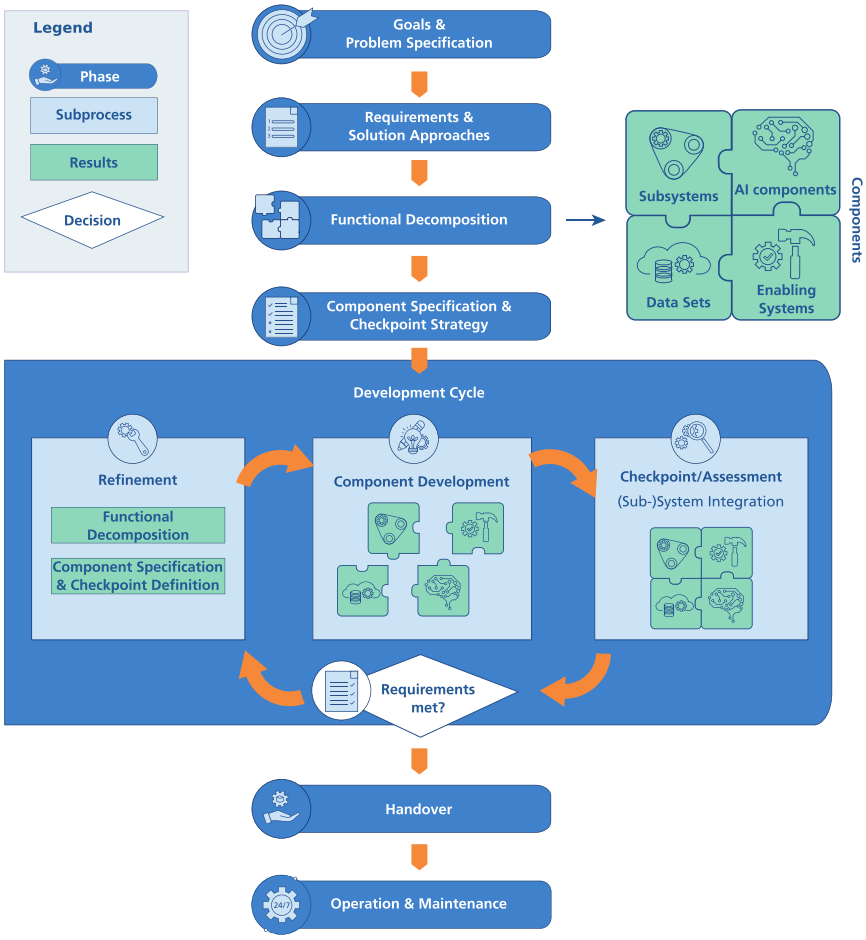

Figure 1: Schematic view on the overall structure of the Process model for AI Systems Engineering (PAISE®).

clearly defined interfaces and can themselves be decomposable. ML methods can be integrated directly into subsystems or into so-called enabling systems that are not part of the overall system and are therefore not delivered to customers, but enable the development and maintenance of other subsystems. As pointed out in Section 1, data sets and their quality play a crucial role in the process of AI systems development. Hence, data sets are developed individually and are treated as a kind of subsystem. Subsystems, enabling systems, and data sets are referred to as components in PAISE®.

The schematic view on the overall structure of PAISE® is presented in Figure 1. The process model consists of seven phases that are arranged in a waterfall-like structure.

The first two phases, *Goals & Problem Specification* and *Requirements & Solution Approaches*, adopt the processes "Business or Mission Analysis Process", the "Stakeholder Needs & Requirements Definition Process" and the "System Requirements Definition Process" from the standard ISO/IEC 15288. Overall project goals are defined, product requirements are derived and first ideas of how to approach the problem are developed. It should be pointed out that all considerations in theses phases should address the overall system and should not go into the level of detail of e. g., a functional decomposition. If the decision has been taken to apply AI methods, requirements from legal regulations must be considered [5]. Additionally, corresponding data has to be integrated into the requirements and the solution approach in that case.

The artifact of *role distribution* is initialized during the phase *Requirements & Problem Understanding* to have a clear distribution of responsibilities. It lists the responsibilities required in each phase and is extended, if needed, in all following phases.

Not AI specific is also the component specification part in the phase *Component Specification & Checkpoint Strategy*. Together with the *Functional Decomposition* it adopts the "Architectural Definition Process" from the standard ISO/IEC 15288. Components defined during the

Functional Decomposition are initially specified including their interfaces. The definition of the checkpoint strategy, however, prepares the structure of the AI specific development cycle and will be explained in Section 4.

The phase *Handover* covers the "Transition process" from the standard ISO/IEC 15288 where the product is transferred from the development team to the organizational units that realize operation and maintenance. For this purpose, an operation and maintenance concept is developed together with the corresponding documentation for the user (e. g., user manual) and the service team. The concept includes the definition of normal operation modes of the product as well as escalation levels with corresponding measures and maintenance triggers. Ideally, these aspects were already considered during the development cycle and only have to be documented at this stage.

In contrast, the characteristics of the phases *Functional Decomposition*, *Development Cycle* and *Operation & Maintenance* are specific to the application of AI methods in addition to general aspects from the "Design Definition Process", "Implementation Process", "Validation and Verification Process" and "System Analysis Process" from standard ISO/IEC 15288.

Their corresponding activities of theses phases and how they address the two challenges of AI systems engineering introduced in Section 1 are described in the following sections.

3 Functional decomposition

During the phase of *Functional Decomposition*, the functions of the overall system are initially distributed onto subsystems. The result is a mostly hierarchical subsystem specification with well-defined interfaces. This is supplemented by the specification of additional enabling systems. The granularity of the subdivision depends strongly on the complexity of the system.

It is important to emphasize that relations between components can be of very different nature, such as data exchange, electrical currents or mechanical forces. Hence, several system models can exist depending on the kind of relation. The same holds for the type of subsystem, which can be hardware (e. g., mechanical parts) and/or software (e. g., algorithms). Depending on the complexity of the use case this distinction can require several system models.

In the context of AI systems engineering, it is crucial to incorporate data sources into the system model. Data sources are understood to be subsystems or enabling systems that provide data for development and/or for operation and thus significantly influence the functionality of AI components. This consideration, together with a dedicated process on how to acquire data from the data sources and evaluate data quality, addresses challenge 1 (see Section 1).

The decision as to which subsystem should be AI-based can be made either from experience during the initial *Functional Decomposition* or during the *Development Cycle* in which refinement of the initial decomposition is envisioned. It is important to note that the decision to use AI is part of the solution approach, not the requirements. Additionally, the need for additional enabling systems or subsystems may arise during development, e. g., when additional data sources are needed. Similarly, components may be dropped if they are no longer needed. It is therefore important to note that the initial functional decomposition is not final, but should be used for the first iterations of the approach and may be subsequently adjusted. The same holds for the component specifications that are documented based on the initial functional decomposition and the system requirements in the phase *Component Specification & Checkpoint Definition*.

Figure 2 shows an example of a system model for an emergency breaking system that autonomously brakes a car in the event of danger. This is realized by detecting objects in camera images. The decision maker estimates the distance and relative speed of the objects and makes an emergency braking decision based on this information. Both detector and decision maker can be developed as AI-based subsystems. Enabling systems in this scenario are, for example, the software that performs an optimization of the attachment position for the given camera or a database in which recordings of the camera can be stored.

4 Development cycle

To address challenge 2 (see Section 1), PAISE® defines an iterative development cycle which is supported by checkpoints to synchronize component development. It allows switching between an exploratory approach on the one hand and a goal-oriented approach on the other. By iterating through the cycle, the maturity of the components and therefore of the overall system is continuously increased. Each cycle consists of a *Refinement* step, a phase of parallelized *Component Development* and a *Checkpoint* including assessment of the development results. Finally, the results of the evaluation lead to the decision whether the overall system is complete with respect to the re-

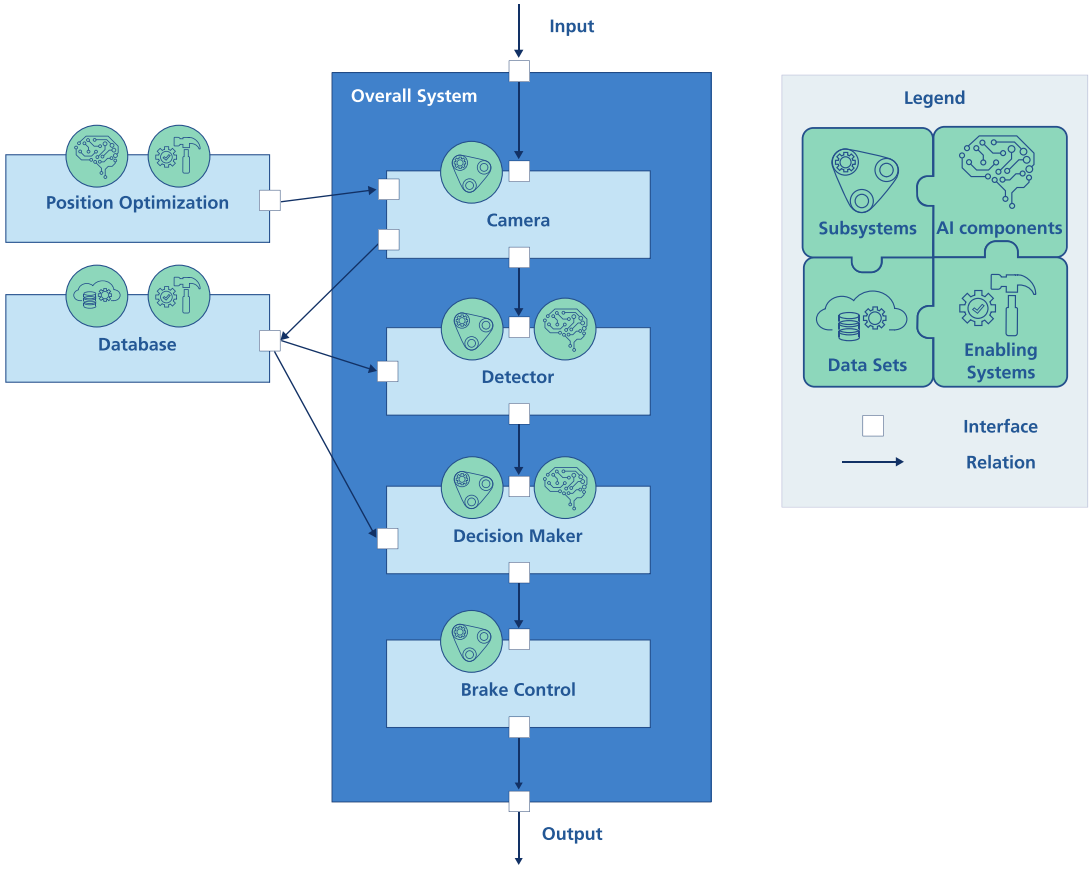

Figure 2: Example of a functional decomposition for an emergency braking system.

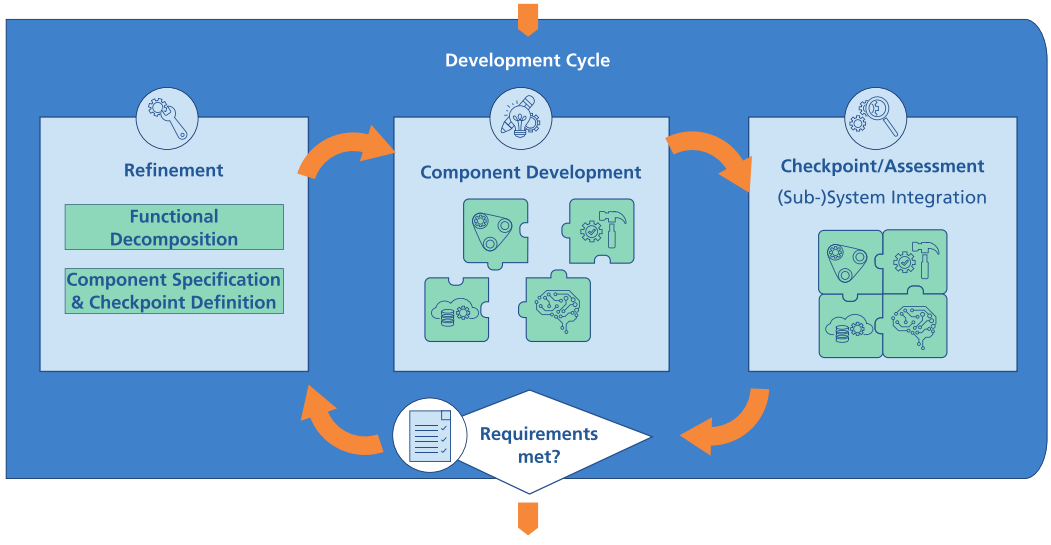

Figure 3: Substructure of the Development Cycle.

quirements. While the processes *Refinement* and *Component Development* take place within the respective disciplines and are parallelized for all components, the subprocess *Checkpoint/Assessment* takes place across components and domains. Figure 3 shows a schematic view of the development cycle and its sub-processes.

In the *Refinement*, the problem solution approach of each component is detailed and varied if necessary. The

solution approach is translated into the appropriate adjustments regarding the system model as well as the component specification. At this point, a further decomposition of components may be needed, e. g., in order to avoid bottlenecks during the development, or additional components, e. g., with inclusion of new data sources. Improvements to individual subsystems must always be considered in the context of the overall system.

Basis for the parallelized *Component Development* are the component specifications, which are to be fulfilled and validated. Development takes place for each component according to an individually appropriate and domain-specific procedure. For classical components such as mechanical or electrical subsystems, for example, procedures from systems engineering can be used. The prerequisite is that these can be integrated into the cyclic structure described here. PAISE® specifies its own procedures for *ML component development* and *data provisioning*. Once a solution approach for a component has been implemented (whether prototypical or refined), it can be integrated into the surrounding subsystem and validated and verified within integration tests.

Checkpoints are used to synchronize the development status of all components and to test the interaction of the subsystems within the overall system. For this purpose, (partial) integration of the subsystems takes place combined with verification and validation tests with regard to the requirements of the overall system. Since the component development allows an individual procedure for each subsystem, progress can be expected at different speeds. Hence, not all components need to actively participate in each checkpoint.

Different strategies can be followed in determining when and according to which criteria a checkpoint takes place. Although classic milestone planning is possible and has many advantages in terms of predictability, a more agile approach is recommended where the goals for the next checkpoint are defined during the *Refinement* step. Strategies how to define the next checkpoint can be the following:

- Feature-based: Definition of features that should be implemented for each component until the next checkpoint
- Maturity-based: Definition of degree of maturity defined with respect to the requirements of the overall system
- Time-based: Definition of a fixed time interval after which the next checkpoint takes place

Overall, the checkpoint serves to focus on interdisciplinary cross-sectional aspects. In addition to considering functional safety or even costs, this may also include open discussion of potential ethical conflicts that can be generated, for example, by the use of incomplete data or data with bias included. Such aspects are specifically addressed by the role of the data officer and are also anchored in the data provisioning procedure.

At checkpoints the (partial) integration and evaluation of components with respect to requirements takes place. Checkpoints therefore serve as a point of synchronization of all components. For ML components, this means evaluation against validation metrics that assess the component's functioning within the overall system. The results of a checkpoint can lead to refinements as well as adjustments in the solution approaches used to target component functionality. By iterating through *Component Development*, *Checkpoint*/*Assessment* and *Refinement*, the maturity of all components and thus the overall system is continuously increased.

The checkpoint-based cyclic procedure aims at a continuous improvement of the overall system. In doing so, it contains three properties that are in our view essential for AI systems engineering:

- It respects the fact that the development of some components might depend on others.
- It makes an explorative procedure possible, which is particularly necessary for the development of ML-based components, since no guarantees can be given in advance regarding the fulfillment of the requirements (see challenge 2).
- It offers the framework for a risk-based development which permits alternative solutions, allows prototyping and weighs risks against gains.

5 Data provisioning

The data provisioning procedure has the purpose to generate, prepare and evaluate training, test and validation data sets. In the refinement step, the specification of the data is adapted including data sources, which may be different for the stages training, testing, and runtime.

The data requirements comprise technical aspects relevant to the accomplishment of the AI component's tasks. Examples are the amount of data (how many measurements are available), its quality (e. g., how much missing or incorrect information) and its representativeness (whether the training data represents the data that will be generated at runtime). In addition, there are high-level non-technical aspects such as bias in the distribution of data, which can lead to unfair decisions, costs in data acquisition, and legal aspects of personal data, which in

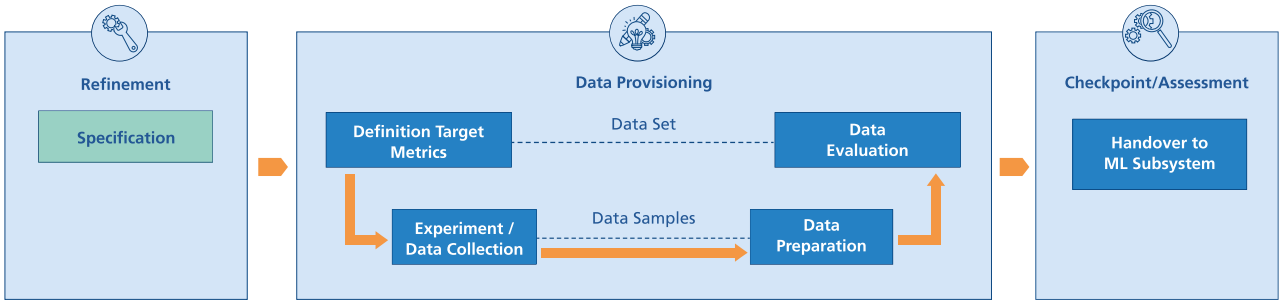

Figure 4: Scheme of the data provisioning process.

turn require additional steps such as anonymization or pseudonymization.

The data provisioning procedure is based on the V-Model introduced in Section 1. Each development step has a testing and verification step at the same level of detail. A scheme of the data provisioning process is shown in Figure 4. Two levels of detail have been proven to be useful: The *data set level* and the *data sample level*.

At the data set level target metrics are defined which are used later to evaluate if the data requirements are met. This is followed by the data collection step which can range between a simple gathering of available data, augmenting existing data, executing a simulation or conducting experiments and measurements under well-defined conditions. The acquired raw data are prepared with respect to the problem definition. This step comprises for example the derivation of features, i. e., measurement characteristics that serve as input to the ML component, data annotation (labelling), the aggregation of multiple data points, noise removal, filtering of incomplete data points or imputation of missing information. Furthermore, multiple features can be combined into a new feature to reduce the dimension and thus complexity of the data points. If the requirements demand it, techniques of anonymization or pseudonymization of data are also applied in this step. While some of the preparation steps are only necessary for training, testing, and validation data (e. g., annotation), some methods must be applied to the data at runtime for consistency reasons (e. g., noise removal, data imputation, combination of multiple features, etc.). Hence, part of the defined data preparation steps has to be transferred and implemented into the ML-component. The data provisioning procedure finishes with the data assessment where the previously defined target metrics are evaluated. After successful completion, the processed and evaluated data are made available for the development of AI components at the next checkpoint.

It should be emphasized that the process of data preparation does not necessarily have to be manual, but that there is great potential for automating individual steps or work sequences.

6 ML component development

The procedure for ML component development is based on the V-Model as well. The goal of the procedure is the encapsulation of an ML model (i. e., the data-driven part) into a component. This facilitates the substitution of data sources and related enabling systems in order to be able to iteratively integrate and validate results within the checkpoints. This approach creates an organizational interface between the classical data science discipline and systems engineering. A scheme of the ML subsystem development process is shown in Figure 5.

In *Refinement*, the component is specified in the context of the surrounding system and the specification is detailed iteratively and adapted within each cycle. Among others, the specification comprises the ML method (e. g., neural network, decision tree, etc.) and the software framework (e. g., TensorFlow [11] or Pytorch [8]) as an approach to achieve the requirements. Both choices should take higher-level requirements for the overall system (e. g., traceability of decisions) and direct dependencies on other components into account (e. g., limited computing resources, availability of data, availability of target variables). For the process of ML subsystem development, we identified three levels of detail: Component architecture, learning architecture and model.

At the component architecture level, in the first step the data source interfaces needed for this cycle are integrated. It may happen that the data sources and thus the interfaces are not the same in every cycle, e. g., if data are initially available in tabular data files and later in a database. At the learning architecture level, test and validation metrics are derived. These are also sometimes denoted as global cost functions that respect the component requirements and are suited for data-driven evaluation.

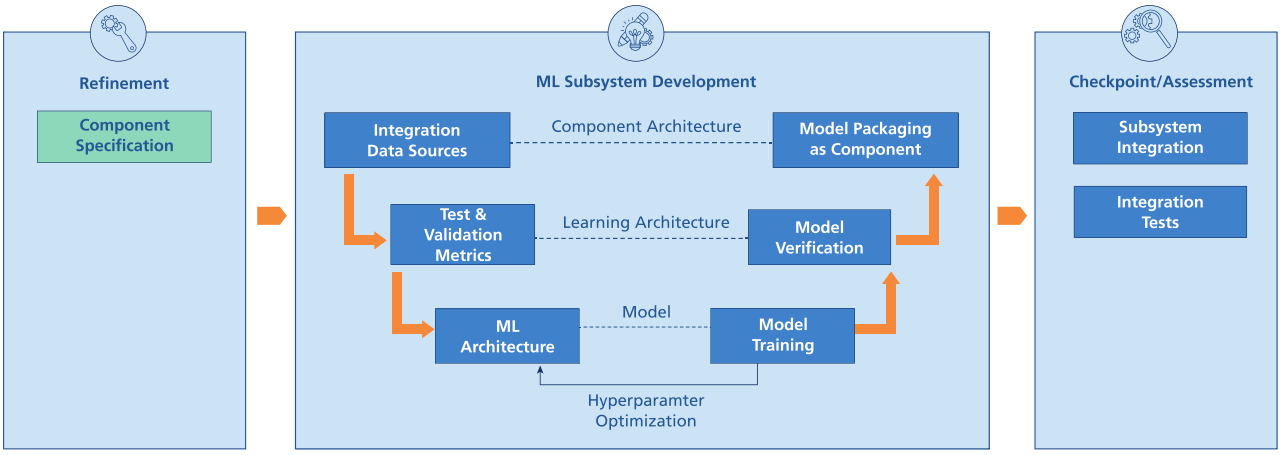

Figure 5: Scheme of the ML subsystem development process.

Domain knowledge should be incorporated at this point in order to ensure the correct functionality of the component. Afterwards, the selected ML method is implemented as a specific model architecture with defined hyperparameters. Examples of hyperparameters are the number of neurons, layers in artificial neural networks, learning rate as well as the definition of the loss function, also denoted as local cost function.

The model is trained using the training data. Since the results of the model strongly depend on the ML architecture, i. e., the chosen hyperparameters, those parameters are iteratively varied in order to find the ideal configuration. In many cases, this hyperparameter optimization can be partially or completely automated by suitable tools (e. g., auto-ML systems).

In the subsequent model assessment, the test and validation metrics are evaluated based on a test data set in order to quantify the quality of the learned and optimized model with respect to the required component functionality.

The final step within ML component development comprises the model packaging as a component where the ML model is prepared for integration into the target platform. While previously the model was validated based on data only, it is now ensured that the model is executable on the target platform (e. g., embedded devices) with comparable results.

At the *Checkpoint* the actual integration of the components into the high-level system takes place. Tests show how the component behaves in the interplay with other components and if it achieves its functionality required by the overall system. If the tests fail the requirements, further cycles are necessary in which further solution approaches, such as other ML procedures, are realized.

7 Operation and maintenance

As soon as all requirements are met, the exit of the development cycle is triggered and the last two phases *Handover* and *Operation & Maintenance* follow.

The monitoring of the ML component functionalities is crucial for reliable operation of AI-based systems. Changes in the data processed during operation can degrade the performance of AI subsystems over time. Such changes can be caused by changes in latent variables (e. g., temperature, humidity, etc.) as well as by changes in the intended use (e. g., foreign traffic signs; new material being processed in the machine, etc.). Ideally, these issues were considered during ML component development leading to additional components whose purpose is the monitoring of the ML component and the incoming data. The trigger for an update of the ML component can be set statically (e. g., time-based) or based on evaluations during operation, e. g., ML model performance or metrics for data distribution shift detection.

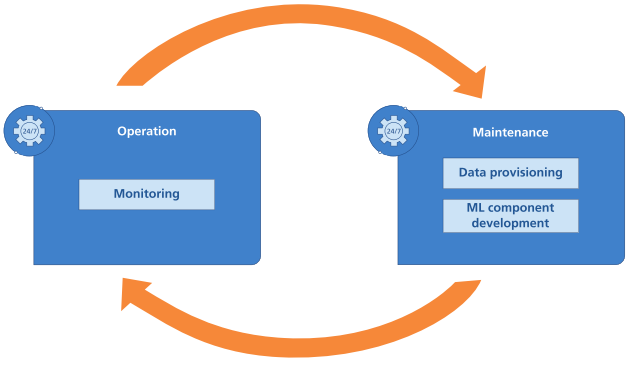

Figure 6: Schematic view on the substructure within the *Operation & Maintenance* phase.

If a model update is required, a new training data set is collected and processed following the data provisioning procedure (see section 6). In the next step, the ML component development procedure (see section 7) is applied in order to re-adjust the ML model. Finally, the updated component is reintegrated into the overall system, tested and put into operation.

Depending on the system's level of autonomy and the corresponding risk assessment, additional boundary conditions from reporting obligations and declarations of conformity, must be addressed during *Handover* and *Operation & Maintenance*. This specifically concerns high-risk AI systems.

8 Conclusion and outlook

Process model for **AI** **S**ystems **E**ngineering (PAISE®) contributes to the aim of integrating AI development into development contexts of increasingly complex systems.

The goal of PAISE® is a – from a technical point of view – high quality product. Therefore, the focus is on the technical processes of the product life-cycle standard ISO/IEC 15288 which are covered by PAISE®. Organizational project-enabling processes, technical management processes and agreement processes are beyond the scope of our considerations.

PAISE® addresses the two challenges of AI systems engineering, i. e.:

1. The ML component functionalities crucially depend on data from operating components and
2. the performance of ML components is not predictable in advance,

by the core ideas:

- definition of data sources within the *functional decomposition* (challenge 1),
- iterative checkpoint-based development cycle (challenges 1 and 2),
- development of data sets as components in the individual process of *data provisioning* (challenge 1),
- development of ML components in a structured process where key performance metrics are constantly monitored and compared to requirements (challenge 2),
- monitoring and update of ML components during operation (challenges 1 and 2).

The iterative checkpoint-based development cycle on the one hand allows for an explorative approach and on the other hand ensures the consistent integration of all components into the overall system in order to respect and react to dependencies (e. g., the dependency of ML component on incoming data from other components). In the spirit of the Dev-Ops approach in software development, it supports the prototypical integration and testing of ML components into the overlying subsystem of hardware and software at an early stage and, conversely, hardware-software subsystems are used at an early stage to generate data that can lead to an optimization of the AI component. Thus, instead of a local optimum only for the AI component, a global optimum for the entire system can be achieved.

Developing the ML-component in a V-Model approach integrated into the checkpoint-based development cycle creates an organizational interface between the classical data science discipline and systems engineering. In particular, the validation and verification approach ensures the verified implementation of requirements and can also respect severely restrictive boundary conditions such as limited computing and memory resources.

PAISE® is a process template that can and should be adapted depending on the organizational framework in companies. As with any process model, the elements of PAISE® must be transferred to the respective use case in order to derive specific actions.

It should be emphasized that the checkpoint-based cyclic development procedure together with the processes of *data provisioning* and *ML component development* represent the core of PAISE®. The described seven phases can also be arranged in other structures. The basic "checkpoint-based" concept of PAISE® within the development phase with preceding system division into subsystems is still applicable.

PAISE® has been successfully employed in cooperation with industrial partners, e. g., the company Hypercon Solutions GmbH to structure a project for optimized production of ultra high performance concrete (UHPC) building materials. Further verifications of PAISE® with other partners in other branches of production and mobility are ongoing and subject to current research.

Future efforts will go into specifying and establishing profiles of PAISE® tailored to selected application domains or problem spaces. Furthermore, the following aspects will be considered: How does the checkpoint-driven development cycle scale with the size of the development team? How can a project time schedule be developed despite the cyclic development cycle? How can certification bodies be involved in the development process? Hence, there are promising results expected in the future leading to an even more optimized and adapted process for AI systems engineering.

Acknowledgment: The authors thank the following persons for their support and help during the development of PAISE®: *Fraunhofer IOSB* – Dr. Julius Pfrommer, Dr. Thomas Usländer, Jens Ziehn; *Research Center for Information Technology* (*FZI*) – Dr. Sebastian Reiter, Michael Weber; *Karlsruhe Institute of Technology* (*KIT*) – Dr. Till Riedel.

Funding: This work was supported by the Competence Center Karlsruhe for AI Systems Engineering (CC-KING, https://www.ai-engineering.eu) sponsored by the Ministry of Economic Affairs, Labour and Tourism of Baden-Württemberg (http://wm.baden-wuerttemberg.de/).

References

1. Adler, R., T. Andersen, M. Anton, A. Aschenbrenner, Y. Babar, A. Bahlke, et al. 2020. In: (W. Wahlster and C. Winterhalter, eds) *Deutsche Normungsroadmap Künstliche Intelligenz*. DIN / DKE, Berlin.
2. Boehm, B. 2006. A view of 20th and 21st century software engineering. In: *ICSE '06: Proceedings of the 28th international conference on software engineering*, pp. 12–29.
3. Hasterok, C., J. Stompe. 2021, December. *PAISE® – Vorgehensmodell für KI-Engineering*. Retrieved from https://www.ki-engineering.eu/de/wissen-tools/paise.html.
4. Chapman, P., et al. 2000. *CRISP-DM 1.0: Step-by-step data mining guide*.
5. European Commission. 2021. *Coordinated Plan on Artificial Intelligence 2021 Review*. COM(2021) 205 final, Brüssel.
6. Friedrich, J., M. Kuhrmann, T. Ternité. 2009. *Das V-Modell XT. Informatik im Fokus*. Springer, Berlin, Heidelberg.
7. Beedle, M.A., K. Schwaber. 2002. *Agile software development with Scrum*. Prentice Hall.
8. Paszke, A. 2019. PyTorch: An imperative style, high-performance deep learning library. In: (H.W.-B. Garnett, ed.) *Advances in Neural information processing systems 32*. Curran Associates, Inc., pp. 8024–8035.
9. Ropohl, G. 2009. *Allgemeine Technologie: eine Systemtheorie der Technik*. Universitätsverlag Karlsruhe, Karlsruhe.
10. Royce, W. 1970. Managing the development of large software systems. In: *Proceedings of IEEE WESCON 26 (August)*, pp. 1–9.
11. Tensorflow. 2021, November 4. An end-to-end open source platform for machine learning: 10.5281/zenodo.5645375.

Bionotes

Dr. rer. nat. Constanze Hasterok
Fraunhofer IOSB, Karlsruhe, Germany
constanze.hasterok@iosb.fraunhofer.de

Dr. Constanze Hasterok is a research associate and project manager at Fraunhofer IOSB in the department "Information Management and Production Control".
She pursued her dissertation at the Max Planck Institute for Nuclear Physics in Heidelberg in the field of statistical analysis and modeling of data from particle detectors. At Fraunhofer IOSB, her research focuses on the application of machine learning methods for the optimization of industrial production processes.
Dr. Constanze Hasterok is head of the PAISE (Process Model for AI Systems Engineering) working group within the Competence Center for AI Systems Engineering (CC-KING) in Karlsruhe.

Dr. rer. nat. Janina Stompe
Fraunhofer IOSB, Karlsruhe, Germany
janina.stompe@iosb.fraunhofer.de

Dr. Janina Stompe received her PhD in applied mathematics from the Chair of Applied and Numerical Mathematics at the Karlsruhe Institute of Technology (KIT). The topic of her PhD lies in the area of inverse scattering theory. At Fraunhofer IOSB, Dr. Stompe focused on the optimization of industrial production processes based on the application of machine learning. In addition, she was head of the Training Laboratory of the Competence Center for AI Systems Engineering (CC-KING) in Karlsruhe. In June 2022, Dr. Stompe left the IOSB and is now working as project manager for funded research projects in the field of autonomous driving at understand.ai (UAI) in Karlsruhe.