



Funded by
the European Union



HORIZON EUROPE PROGRAMME: HORIZON-CL4-2022-DIGITAL-EMERGING-02

SolidAIR

Solid, rapid and efficient adoption of Data, AI & Robotics applications in production

Step by step guide on deploying technical systems in the field of Visual AI, Data and AI and Robotics and AI

Primary Author(s)	Emil Andreas FHG Thanasis Mastrogeorgiou THL Martin Wifling VIF
Deliverable Type	Others
Dissemination Level	Public
Due Date (Annex I)	30.11.2025
Pages	62
Document Version	Final
GA Number	101120276
Project Coordinator	Andreas Frommknecht Fraunhofer IPA (FHG) (andreas.frommknecht@ipa.fraunhofer.de)

Contributors	
Name	Organisation
Emil Andreas, Toni Ly, Andreas Frommknecht	FHG
Panagiotis Mavridis, Panagiota Moraiti, Thanasis Mastrogeorgiou	THL
Martin Wifling, Sophia Bastidas	VIF
Jean Pierre Allamaa	SISW

Formal Reviewers	
Name	Organisation
Andreas Frommknecht	FHG
Anesa Begovic	I2M

Project Abstract

SoliDAIR aims to accelerate the uptake of Artificial Intelligence (AI) and Robotics in European manufacturing, using Data as an enabler. It will co-develop and demonstrate tailored solutions to digitalise and automate visual inspection and physical testing, enable predictive quality control and process optimisation. The SoliDAIR project tackles the problem of AI & Robotics systems not being extensively used in the production industry, because it is not clear whether they are safe and when or why they will fail, by researching, developing and testing methods that are as solid and trustworthy as possible to be adopted by the European industry, while being cost-efficient to develop and replicate.

New methods and tools will be developed by research and technology providers, which leverage the current state of the art in visual AI, AI for process data, and smart & collaborative Robotics. The developed technologies will be applied and demonstrated in 4 industry use cases to prove their functionality and applicability in real production environments. The objective is to improve production processes through digitalised and automated quality control for high volume, high rate and flexible manufacturing. The developed methods shall be efficiently and easily adaptable and replicable, so they can be easily applied to new use cases outside the consortium.

Table of Contents

Public Summary	5
1 Guides.....	5
1.1 Theme 1: Visual AI.....	6
1.1.1 Motivation.....	6
1.1.2 Scope.....	6
1.1.3 Data Acquisition	7
1.1.4 Data Augmentation	8
1.1.5 AI Architecture Selection	12
1.1.6 Model Training	12
1.1.7 Testing/Validation.....	12
1.1.8 Explainable Artificial Intelligence (XAI) (Module: XAI for image-based quality control) 13	
1.1.9 Potential Actions for Performance Improvement (Module: Image-based quality control) 13	
1.1.10 Deployment (Module: HMI).....	15
1.1.11 Available Datasets, Pretrained Models and Code.....	16
1.1.12 Expertise needed	16
1.2 Theme 2: Data and AI	18
1.2.1 Motivation.....	18
1.2.2 Scope.....	19
1.2.3 Data Acquisition and Preprocessing.....	23
1.2.4 High-Fidelity Simulation and Reduced Order Modeling (ROM).....	27
1.2.5 Feature Engineering and Identification of Key Properties	33
1.2.6 Synthetic Process Data Generation and Model Validation.....	36
1.2.7 Hybrid Model Development and Training.....	41
1.2.8 Model Performance Monitoring and Continuous Improvement	43
1.2.9 Available Datasets and Code	46
1.2.10 Expertise needed	46

- 1.3 Theme 3: Robotics and AI50
 - 1.3.1 Motivation.....50
 - 1.3.2 Scope.....50
 - 1.3.3 Requirements Identification and Robotics Platform Selection.....50
 - 1.3.4 Simulation and Digital Twin54
 - 1.3.5 Physical Deployment.....57
 - 1.3.6 Available Datasets and Code59
 - 1.3.7 Expertise needed60
- 2 Acknowledgements and disclaimer62

Public Summary

This step by step guide on deploying technical systems in the field of Visual AI, Data and AI and Robotics and AI is primarily designed for a collaborative, multi-disciplinary team within manufacturing organizations. The step by step guide aims to support this teams in the development and deployment of AI based systems in industrial production environments.

1 Guides

Targeted user group

This step by step guide and the outlined AI approaches are primarily designed for a collaborative, multi-disciplinary team within manufacturing organizations. The core user group consists of specialized domain experts who are deeply familiar with the product and process but are **not necessarily AI experts**. This includes **production engineers** and **machine/process experts** responsible for factory-floor operations, and **product development engineers** involved in defining component specifications and simulation models (e.g., the physics-based model). These domain experts will be **supported by data scientists and AI specialists** who focus on model development and optimization. The overall process is structured to bridge the gap between these disciplines. The outcome, developed through a generalized methodological framework, serves as a blueprint to give the entire team a transparent view of the process they are not experts in, enabling them to understand, trust, and contribute their domain knowledge effectively.

Modules

To ensure continuity within the SoliDAIR project, the modules referenced in this step-by-step guide correspond to the generalized modules defined in the SoliDAIR framework across Visual AI, Sensor Data & AI, and Robotics & AI. Figure 1 from Deliverable 2.1 (D2.1) is reproduced here as a visual anchor, so the multidisciplinary team can quickly relate each step to the established building blocks. While the module names used in this guide are pragmatic and may differ slightly from those in the figure¹, each listed module under a step maps directly to its counterpart in that figure.

¹ E.g., Blender-based high-fidelity synthetic image generation, High-fidelity simulator and GenAI synthetic image generation, Generative synthetic Image Generation are all part of the Module “Efficient synthetic data generation” visible in Figure 1

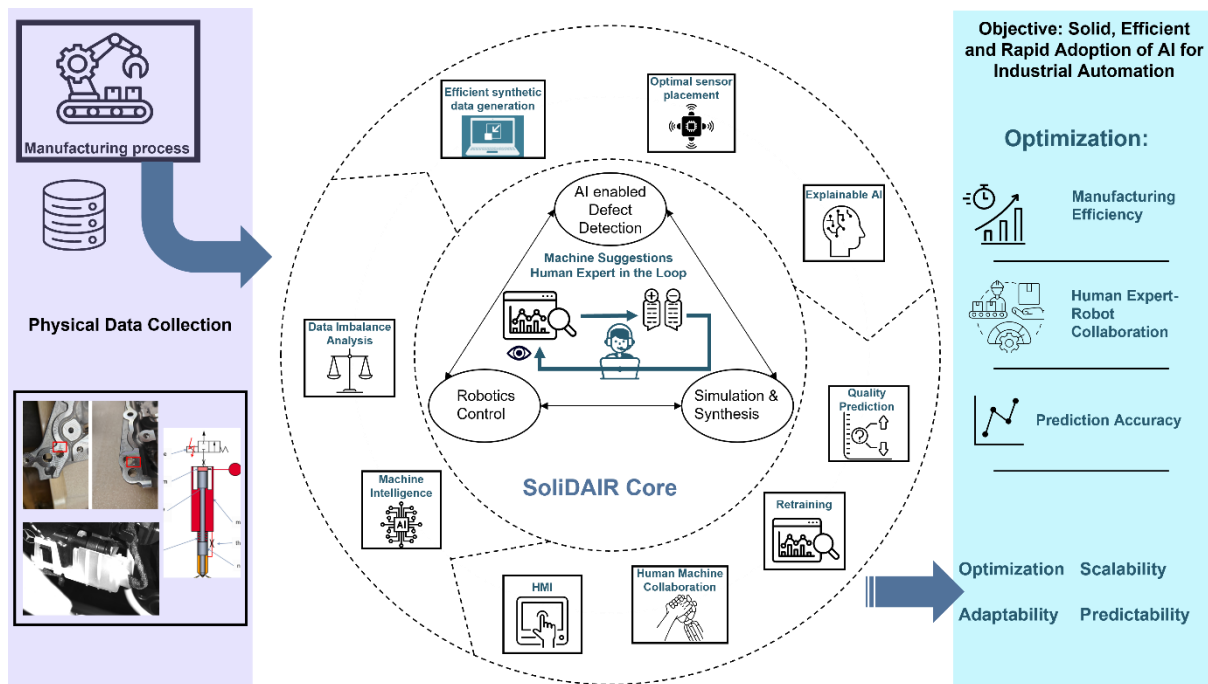


Figure 1 Objective of and developed modules within the SoliDAIR project.

The list of modules under each step refers back to the generalized modules presented in D2.1. For detailed descriptions, readers can consult D2.1; this guide focuses on how those modules are operationalized within the manufacturing workflow.

1.1 Theme 1: Visual AI

1.1.1 Motivation

Following the Grant Agreement, SoliDAIR aims at accelerating the uptake of Data, AI and Robotics in European manufacturing. Visual AI is one of the key points in this context. Traditional visual inspection methods typically rely on fixed thresholds, handcrafted features or human operators, which can be limited in consistency, scalability and adaptability. In contrast, AI systems using computer vision and machine learning can learn directly from data, enabling them to detect subtle defects, variations or patterns that might be overlooked by rigid rule-based algorithms or human operators.

Adopting AI-based solutions in mature manufacturing processes is heavily context-dependent. Nonetheless, the consortium is convinced that this guide provides general insights and actionable steps on how to implement visual AI in said environments, to lower entry barriers and enable repeatable deployment. A major characteristic of mature manufacturing processes is the high amount of in-specification (OK) data in comparison to the out-of-specification (NOK) data, that often occurs in the parts-per-million (ppm) range. Tackling data scarcity by simulation-based and/or purely data-driven image generation methods is a key contribution of the presented workflow.

1.1.2 Scope

Application Scenarios:

Generic image data from a manufacturing process, consisting of labelled pairs of images and annotations, either binomial (OK vs NOK) or multinomial class labels (i.e. OK vs Defect 1 vs

Defect 2) with optionally additional bounding boxes for detection related tasks must be available for a (supervised) machine learning-based approach. Task at hands should thus be industrial quality inspection by means of image classification, anomaly detection and/or object detection of known objects.

As a first step, the defect detection requirements or respectively error class specifications must be defined, such as defect types, classification criteria, acceptable tolerances, detection rates and inspection cycle time. The results of this step are crucial, as they directly determine the specifications and selection of the vision model and associated hardware, as well as the evaluation of the proposed solution and the quality of the annotations used to train the AI models.

Boundary Conditions:

To be applicable without a preliminary anomaly detection, all relevant classes, i.e. different defects and the non-defective class, must be present in the data. Application of anomaly detection, either as a standout solution or as a preliminary step for further downstream applications, is most promising when a relatively large and diverse set of non-defective images are available but very few or no defective images are available. Additionally, if defect modes are open-ended and/or fine-grained, such that previously unseen defects may arise, anomaly detection is advantageous because it models normality and flags deviations, enabling discovery of novel defects without exhaustive defect labels. As a standalone solution, anomaly detection cannot differentiate between specific defect types which needs to be considered when different defects demand different courses of actions. For this task classification models can be used. If not only the defect type (what) itself is relevant but also the location (where) of the defect, object detection comes into place.

1.1.3 Data Acquisition

Note: For a mature solution, the data acquisition of the training data should – unless strictly necessary – not differ from the acquisition process during inference, when the system is deployed.

Step 1: Image Acquisition Equipment

Task: Select and configure appropriate imaging equipment

Description: Choose between area-scan or line-scan camera depending on the geometry of the component and inspection requirements. Consider frame rate and interface type (USB, GigE, Camera Link) based on production speed and integration. Ensure sufficient resolution, defined as the number of pixels per millimeter (spatial resolution), to detect the smallest defect of interest.

The choice of lens depends on the total field of view (FoV) required to capture the entire inspection area while minimizing distortion and maintaining sharp focus. For example, inspecting defects inside cavities may require a different lens than examining a wide, flat surface.

The working distance (WD) should be carefully regulated according to the lens specifications and field of view to maintain proper focus and image clarity. Even small defects must be clearly visible in the image. If not, the working distance should be reduced.

Step 2: Lighting Setup

Task: Design and optimize the lighting setup for defect detection

Description: Proper lighting is crucial for reliable defect detection. The illumination should enhance contrast between defects and background while minimizing shadows and reflections (particularly for metal or plastic surfaces).

Flat dome lights are ideal for wide, flat areas to provide uniform illumination, minimizing shadows and reflections. Ring Lights are best for illuminating cavities or holes and can be combined with special equipment or diffusers to reduce glare on reflective surfaces. Bar lights are useful for highlighting surface textures. They can be angled to enhance contrast and reveal surface details more clearly. You should also consider ambient light to avoid interference or undesirable reflections and avoid direct, harsh illumination on glossy surfaces. In addition, you can use diffusers, multiple light sources experiment with angles, intensity, and polarization if necessary to optimize defect visibility.

In the captured images, defects should appear clearly and prominently, with no shadows or reflections. If this condition is not satisfied the lighting setup should be adjusted to ensure optimal image quality.

Challenges: Shadows and reflections on surfaces are a major challenge. Small or subtle defects also require special attention to ensure proper visibility. Additionally, achieving adequate illumination inside cavities can be difficult.

Step 3: Dataset Creation

Task: Collect, categorize, annotate, and split images to create a high-quality dataset for training and evaluation

Description: After ensuring clear and consistent images, proceed with data collection by capturing images of all possible defects.

Based on the inspection requirements consider different positions and angles if needed. Categorize different defects into separate classes. Ideally, if feasible, include at least 100 images for each defect type. While more images generally improve system performance, as long as variance across the dataset increases with more images. Thus, avoid duplicates and repetitive images. Ensure that all variations of a specific defect type are included. For example, if collecting images of surface defects, including bumps, scratches, or any other forms of surface irregularities that can be observed.

Next, annotate the collected data using an annotation tool suitable for the task. For object detection, ensure that the bounding boxes tightly enclose each defect. For classification and anomaly detection tasks, label the images accordingly.

Finally, split the dataset into three sets. Good practices are approximately 60-80% for training, 10-20% for validation, and 10-20% for testing. If the same defect is captured multiple times, ensure that the sets are independent, so that no identical defect appears in more than one set.

Challenge: Limited available defects can result in a small dataset, reducing the diversity and robustness of the model. For highly imbalanced datasets, every defect (class) needs to be present in every split, which can be problematic if only a few samples, i.e. less than three samples of said class are available. In this case, an anomaly detection approach can be considered as an alternative to classification.

Tools:

- **Open Source:** Any suitable imaging software can be used for data collection, and any annotation tool can be used for labeling. **Python** with its most common data science packages such as **NumPy**, **Pandas**, **Scikit-learn** etc. can be used for dataset splitting and data analysis.

Examples for such acquired datasets, which can be used for model training and testing by data scientists are provided by SoliDAIR Zenodo community (cf. Chapter 1.1.11).

1.1.4 Data Augmentation

Whenever image classification/object detection models are being trained, the standard list of augmentations (e.g., vertical/horizontal flips, (random) cropping and rotations, blur, noise, color jitter, grayscale) should be carefully considered and beneficial augmentations should be

incorporated in the training process. Augmentations, especially for small datasets or underrepresented classes can significantly enhance performance. Because data-driven methods often transfer poorly across domains (e.g., from production line 1 to production line 2), carefully designed augmentation is crucial for improving robustness and generalization.

Synthetic Data Generation:

In many mature manufacturing applications, the collected datasets are highly imbalanced as faults often appear in the ppm range. As deliberately creating defective pieces is often infeasible, to fully exploit the potential of the existing defective samples, synthetic defective parts can be created. Depending on the availability of different types of data (e.g., 3D models, CAD or blender data, raw image data), different approaches can be taken.

For purely data-driven approaches modern image generation models (mainly GANs and Diffusion Models) can be trained from scratch or fine-tuned using non-expensive fine-tuning techniques (e.g., SoRA). If a 3D environment can be set up to generate simulation data, style-transfer methods can be applied to add photorealism, adapting from the simulation into the real image domain and therefore reducing the style-gap of synthetic images. The synthetic data should be kept separated from the original training data to avoid confusion, even though it will be used to extend the training data corpus.

Generate Synthetic Data based on a 3D Simulation Environment (Modules: Blender-based high-fidelity synthetic image generation, High-fidelity simulator and GenAI synthetic image generation, Generative synthetic Image Generation)

A generalized procedure developed in the SoliDAIR project for generating synthetic images with defects will be presented. The proposed pipeline (Figure 1) is designed to be generic and applicable in different use cases. The primary requirement for using this approach is the availability and quality of a 3D model of the object. This 3D model must be provided in standardized 3D representation format such as *.stl*, *.obj*, or *.glb*.

Step 1: Importing and Creating Photorealistic Materials Using Blender

Task: Import 3D models and create photorealistic materials using Blender for rendering realistic images

Description: By using Blender, you can import 3D models and use materials to create textures on their surface. Then these models can be rendered through ray tracing, to create photorealistic images. These materials are created by adjusting the color, metallic and roughness parameters, through various noise and wave textures, which are generated procedurally. This can give a photorealistic result, when rendered by the Cycles Ray Tracing Engine used by Blender. The result is the creation of photorealistic 3D models of objects.

Tools:

- **Open Source: Blender** for modeling and rendering.

Step 2: Simulating Defects Using Geometric Operations

Task: Simulate realistic defects on 3D models using geometric operations to generate diverse synthetic examples

Description: The next step is the addition of characteristics (defects) to these 3D models. Small, irregularly shaped features can be generated and randomly placed on the model's surface to simulate defects. It is important to analyze real defects in your dataset and generate synthetic examples that capture similar patterns and variations, ensuring as much diversity as possible for robust model training. Geometric operations

(such as Boolean Difference) can be applied between the base model, and these defect features, surface irregularities or indentations can be created.

Step 3: Generating Synthetic Dataset and Annotations

Task: Generate a synthetic dataset by rendering defected 3D models and create corresponding annotated mask images for training

Description: By applying random position/rotation to one of, or if applicable both the defect objects and the camera, you can generate a diverse set of synthetic images. To use these images as a synthetic dataset, you can create the corresponding mask image that indicates the pixels where a defect exists (annotation). Mask images are generated by marking defective regions in black and non-defective regions in white, providing precise annotations that are essential for training defect detection models.

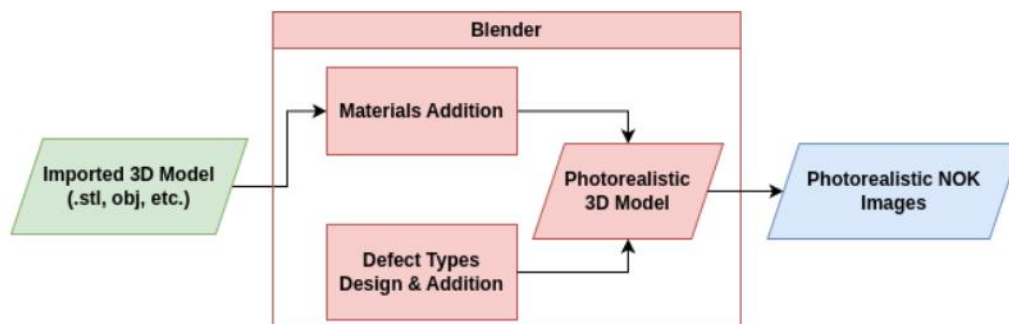


Figure 2 Synthetic image data generation pipeline using the Blender software

Step 4: Simulator-Based Data Augmentation and Realism Enhancement

Task: Enhance synthetic data by performing simulator-based augmentation and applying style transfer methods for improved realism

Description: The generated 3D Blender part can be imported to an environment simulator, Simcenter Prescan to allow further data augmentation through a variety of scanning configurations, high-fidelity camera models, as well as multiple lighting, material and reflectiveness conditions. Once the new synthetic images are generated, they are fed into the style transfer network to add realism.

Tools:

- **Open Source:** **Cycle-GAN**, or similar methods (i.e. **Style-GAN** or **USO** for realism enhancement via style transfer.
- **Proprietary:** Environment simulator, such as **Simcenter Prescan**.

If no 3D model of the part is available other methods adapted in SoliDAIR project come into place:

Generate Synthetic Data based on Image-to-Image (I2I) Deep Learning (Module: Generative synthetic Image Generation)

Step 1: Problem and Data Readiness

Task: Define the defect classes to be synthesized

Description: Gather all source images of the defective classes to be synthesized. Ideally, add an equal number of non-defective images to increase the dataset size used for training the Image-to-Image generation model. Ensure images reflect real acquisition conditions (camera, lens, lighting, working distance) to keep synthetic data

faithful to deployment. The smaller the set of available defective images is, the more important is the label quality. If feasible, verify label correctness.

Tools:

- **Open Source: Python** and its commonly used packages for image processing (e.g., **PIL**, **torchvision**)

Step 2: Choosing I2I Pipeline

Task: Select the model and training stack

Description: If the downstream application requires image resolutions upwards from 128×128 , a latent-space approach is recommended to reduce computational overhead. For that, a standard Variational Autoencoder (VAE) and a trainable denoiser (U-Net, Vision Transformer (ViT) or Diffusion Transformer (DiT) backbone) can be used. In most setups, keep the VAE frozen and train the denoiser end-to-end on your source data to preserve photometric fidelity. Confirm support for conditioning, i.e. on class labels to be able to target specific classes to be synthesized. Alternatively, large pretrained models such as SDXL from Hugging Face can be used.

Challenge: As of now, CUDA-supported GPUs are a necessity for this type of deep learning applications. Hardware resources such as VRAM need to be monitored, batch and model sizes may need to be adjusted accordingly. Unsupported conditioning mechanisms (e.g., Layout-to-Image for standard diffusion) may block progress.

Tools:

- **Open Source: PyTorch, Hugging Face Diffusers, CUDA toolkit**

Step 3: Train the Denoiser

Task: Optimize denoising latents with conditioning

Description: Prepare the images by resizing/cropping to model input size, normalize color space. Encode source images to latent using the (frozen) VAE. Typical knobs that greatly impact model convergence are learning rate and learning rate schedule, noise schedule, batch size, gradient norms, and conditional-free-guidance if available. Alternatively, when relying on large pretrained models adapt them by using special fine-tuning techniques such as LoRA.

Challenge: Training I2I models can be very finicky, especially with little data. Carefully monitor the losses during the training to quickly detect mode collapse and restart training with different parameters. Overfit may reduce realism or variety of generated images. Therefore, validate model performance on held-out images and use early stopping when the model starts overfitting to its training data.

Tools:

- **Open Source:** trained VAEs (e.g., VQ-F4) for the selected denoising model, model backbone (e.g., **Latent Diffusion Model (LDM)** or **ControlNet**), **TensorBoard** or **W&B** for logging and monitoring training, **PyTorch**

Step 4: Synthesize Target Defective Images

Task: Generate class-conditioned samples

Description: For a chosen defect class, set class conditioning. Sample by starting from random noise and iteratively denoising the latent vectors using the Denoiser. Decode refined latent vectors back to pixels using the VAE Decoder. A visual sanity check should be conducted on the first batches of generated images for each class. If there are no obvious discrepancies between the generated images and the expected results, large amounts of synthetic images can be generated. Control diversity and fidelity via random seeds, classifier-free guidance scale, number of inference steps, and sampler choice.

Tools: all the above-mentioned in Steps 1-3

Step 5: Evaluate Synthetic Images

Task: Assess the quality and usefulness of synthesized images.

Description: Monitoring the loss is not sufficient to be able to assess the quality of the generated images. Compute metrics such as Fréchet-Inception-Distance (FID), Learned Perceptual Image Patch Similarity (LPIPS) and Inception Score (IS) using the synthesized and real images. It is worth noting that not always the images with the best scores provide the best values for the downstream application, i.e., increasing model accuracy of the classification model that is trained on the synthetic images.

Challenge: Iterative nature of generating synthetic images, training the downstream model and evaluating the impact of the synthetic data (cf. sections 1.1.6 and 1.1.7). This evaluation requires ablation studies that compare downstream performance with and without synthetic data. These steps should be executed in a feedback loop.

Tools:

- **Python** libraries such as **torchmetrics** provide a broad range of evaluation metrics

All the standard image augmentations can also be applied to the synthesized data, further increasing dataset variance.

1.1.5 AI Architecture Selection

The model architecture should be chosen carefully based on the dataset size, inference requirements and available hardware resources. Smaller models offer faster inference for real-time inspection, while larger models provide higher accuracy. Other criteria or constraints to be considered can be latency, available hardware such as number of GPUs, CPU cores and VRAM, and training speed. State-of-the-Art pre-trained architectures usually can be found by thorough research. I.e., the **YOLO11** model can be a good choice for **defect detection** and offers several versions (nano, small, medium, large, extra-large) you can choose from. Baseline models to try first in **anomaly detection** can be **PatchCore** or **EfficientAD** model. For image **classification**, of the shelf **ResNet** and **ResNeXt** variants or **Vision Transformers** can be tested first.

1.1.6 Model Training

This step focuses on training the model effectively to achieve good downstream task performance

Task: Train and optimize the model using the annotated dataset with appropriate hyperparameter tuning

Description: Train the model using your annotated dataset, starting with pre-trained weights if available. Carefully tune hyperparameters such as image size, number of epochs, learning rate, batch size, and augmentation types to optimize performance on the training and validation sets. When considering image size, adjust according to capabilities of the inference set-up. Retaining a very high resolution may hinder generalizability of the model, as well as hit potential hardware bottlenecks.

Tools:

- **Open Source: Ultralytics YOLO Python framework** can be used for model training of YOLO networks, other suitable deep learning framework such as **timm** (Pytorch Image Models) and **pytorch-lightning**

1.1.7 Testing/Validation

Model evaluation ensures the trained system performs reliably, highlighting strengths, and weaknesses.

Task: Evaluate model performance using quantitative metrics and visual analysis to identify failures and validate reliability on the held-out test set.

Description: Evaluate the model performance using metrics such as Accuracy, Precision, Recall, mean Average Precision (mAP), false negatives (FN), and false positives (FP) on the test set to avoid overfitting to the training/validation data. To assess the impact of additional synthetic data (cf. 1.1.4 Step 5), ablate the performance increase on said metrics. If applicable, visualize the detections to identify and analyse any failures or missed defects.

Note: In this step by step guide, a positive test result means always the prediction of a defect or a certain class.

Tools:

- **Open Source: Python** packages such as **pycocotools**, **matplotlib**, **OpenCV** and **supervision** can be used for model evaluation and visualizations.

1.1.8 Explainable Artificial Intelligence (XAI) (Module: XAI for image-based quality control)

Beyond performance metrics such as mAP, FN and FP, XAI methods can be used to evaluate whether expectations of human operators align with decision driving features of the visual AI model.

Task: Verifying that potential sources of unwanted bias are not influencing the model's decision-making process and making sure AI decisions align with expectations of potential stakeholders such as shopfloor workers.

- **Description:** Define what “correct focus” means in the context of the task. Visualize the most influential regions on the model using XAI methods to sanity check if the model is not biased by unintentional information. Compare obtained heatmaps on a per sample bases with expected results. To avoid confirmation bias, especially wrong model outputs (i.e. misclassifications or missed detections) should be inspected to verify if insights about the wrong decision can be gained from the XAI method. Insertion/Deletion AUC curves can be computed to verify if identified regions truly strongly impact model output and should be compared with a random baseline. Walk through representative examples with shop-floor experts and record disagreements between model focus and human expectations.
- **Challenge:** XAI increases trust and aids debugging but is not a formal proof of correctness. It should always be combined with robust validation. Many methods are baseline dependent and results can strongly vary depending on the choice of the former.

Tools:

- **Open Source: Python** packages such as **pytorch-gradcam**, **SHAP**, **CAPTUM**
- **Proprietary: HALCONs** heatmap tools

1.1.9 Potential Actions for Performance Improvement (Module: Image-based quality control)

The following actions can be performed optionally to improve the performance of the model.

Step 1: Anomaly Detection Pre-Screening

Description: As not all occurring defects may be present in the training data and to avoid faulty setups such as camera defects, blocked lenses etc. an anomaly detection model might be used before the original downstream task such as classification or

detection. Doing so reduces the probability of overconfident predictions on out of domain data and reduces the mean time to detect faulty setups.

Step 2: Different models for different defect types

Description: You may use separate models for each defect type or class. If certain defects require different camera lenses or lighting setups, treat them as individual cases and train a model for each. That will result in more specialized models that are optimized for each defect type or class, improving accuracy and reliability but increasing integration efforts and hardware demand.

Step 3: Handling Small Defects and False Negatives in Defect Detection

Description: If small defects are being missed by the detection algorithm, you can utilize the SAHI (Slicing Aided Hyper Inference) framework. Extract slices (smaller cropped sections of the original image) from your training and validation images, retaining only slices that contain defects. Remove slices where defects are cropped by more than 70%. Train the detection model on these slices, carefully selecting the tile size and input image size (typically the input image size is equal to or larger than the tile size to make defects more prominent). Choosing slices that are too small may increase false positives, so tune these parameters carefully. Keep in mind that increasing the number of slices will also increase inference time.

During inference, use SAHI to slice the images into slices and detect defects on each tile, ensuring small defects are detected.

Tools:

- **Open Source: Python** libraries such as **SAHI (Slicing Aided Hyper Inference)**.

Step 4: Handling False Positives in Defect Detection

Description: If many false positives (FP) are observed in defect detection task, include a small portion (~10%) of images without defects and thus without any annotations in the training and validation sets, if OK samples were not represented in the validation and training data yet.

If consistent FPs arise from anomalies that resemble defects, such as dirt, consider ensembling detections from multiple models to improve results. For example, if FPs occur due to dirt or internal markings within the material, you could train the model using these images without annotations. If this degrades model performance (increasing false negatives), divide the FPs into distinct categories and train separate models for each type. Each model will specialize in ignoring a specific category, improving overall detection reliability. The key idea is to retain only overlapping predictions from the models, minimizing false detections while avoiding false negatives (ensemble learning).

Step 5: Defect Size Measurement

Description: If quality control requires measuring defect size, first define the threshold in millimeters that classifies a defect as significant. Then, calculate a pixel-to-millimeter ratio using a reference object with known dimensions positioned in a known distance from the camera, and use this ratio to convert defect sizes from pixels to real-world units.

Step 6: Utilize XAI to Debug Model

Description: XAI methods can be used to detect unintentional model behavior based on artifacts that allow the model to find a shortcut in its decision making (see chapter 1.1.8). When detecting such a bias, try to exclude it from the input image by i.e.

cropping more aggressively, converting RGB images into greyscale or masking the image by for example edge detection algorithms.

1.1.10 Deployment (Module: HMI)

Step 1: Implement Export for shadow mode

Description: Implement an export from the existing machine/hardware. Calculate the image to the size, value range and cutout the AI trainings process expects. Lot of errors and inconsistency could happen later, if the machine preprocesses the images differently than the training pipeline. Even small difference like a bilinear to a constant or bicubic interpolation can make small difference in the confidences in the network. To create the preprocess pipeline for the images before the first training process is crucial to avoid hard to track differences between the results in the training and the actual machine later.

Step 2: Human Labelling / Ground Truth (Module: Human Domain Knowledge)

Description: Give the opportunity to locally create a ground-truth to images by a human operator. If the machine already has a traditional way to determine the information on the image its result can be used to help the human operator to label images more efficiently.

Step 3: Use Confusion Matrix

Description: Use Confusion Matrix to show the difference of existing traditional toolchain and ai toolchain and to show the difference between either toolchain and the human label or the existing solution and the human label to enable analysis by the operator of how more efficient (Pseudo NIO/IO rates) the different toolchains are. Confusion matrices are one of the most useful tools to evaluate the performance of a deep learning classification over big dataset and time.

Step 4: Use a minimum confidence of classification (Module: Retraining)

Description: The confidence of a classification can be an indicator if the network is confronted with images that are vastly different to the known images during training. Especially typical hardware errors (sharpness of the lens, lighting changes dramatically) has usually an effect on the confidence. It is an option to use those effects and set a confidence minimum. If the classification finds no class with the given confidence the machine should treat the part as failed (for unknown reasons) and the images should be saved separately to may be used later in a retraining of the network.

Step 5: Implement export in the running process

Description: The software in the machine should enable to export images for the (re)training of the model. A good strategy for this is to use the confidence of the classifier to export images that are most likely different from the current training set and additionally use a random process to export images that seem ok to stabilize the usual use case of the model.

Step 6: Use XAI (Heatmap) for explainability to the worker

Description: The heatmap of the classification with the appropriate architecture and stable training is usually a good indicator what part of the images was relevant for its

decision. This can be used to help the worker (the operator during the automated process) to understand the description of the machine and maybe fix the problem with the part by him/herself and let the machine retest the part. Since the heatmap by itself with its multiple colours is hard to interpreted by the untrained eye it can be useful to extract the relevant region with a threshold on the heatmap image with the value range 0 to 1. This way a clear region in the image can be shown.

1.1.11 Available Datasets, Pretrained Models and Code

The SoliDAIR project has already published and will in the future further publish several data sets, pretrained models and code for the usage in the AI community.

Two bounding-box annotated image datasets are available for defect detection in aluminum automotive components and can be downloaded from Zenodo ([Link](#)):

- **Surface Defects Dataset:** Annotated images of surface defects of aluminum components. Defect types include scratches, dents, etc.
- **Thread Defects Dataset:** Annotated images of thread defects, covering deformations, porosity, etc.

A dataset based on a latent representation of deep learning model features is available for evaluating the performance of classification algorithms under severe data imbalance ([Link](#)):

- **Latents Classification Dataset:** Labeled feature vectors obtained from a deep learning model trained on four classes of bowden sliders.

The code for training (see chapter 1.1.6), evaluation (see chapter 1.1.7), and performance improvement techniques (e.g., SAHI, ensemble learning, defect measurement) (see chapter 1.1.9), along with the corresponding models (YOLO11 for surface and thread defect detection) are planned to be open-sourced on GitHub and possibly shared on other platforms like Zenodo by the end of the project. These resources facilitate the training and evaluation of defect detection models across various datasets. By the end of the project the **I2I generation model** (see chapter 1.1.4) will be open sourced on **GitHub** and potentially linked to other relevant platforms such as Zenodo or AIoD. This repository lowers the entry barrier for other practitioners to train latent diffusion models on their own data.

1.1.12 Expertise needed

From a user perspective, the integration and operation of a Visual AI system are associated with specific requirements regarding the necessary expertise during the different phases.

Development Phase

For setting up the optical inspection system image processing experts, system integrators and production process experts are mandatory. When the system is set up the image data can be provided by the production company to the AI experts, who take care of training the machine learning model. Together with the process experts the trained model is then deployed to and integrated into the inspection system by the system supplier to be evaluated under real production conditions. Technical prerequisites need to be established to permanently integrate these models into an

inspection system. For this purpose, a container-based infrastructure based on Docker is preferred. Standard exchange formats like the Open Neural Network Exchange (ONNX) are helpful for smooth and easy integration.

On the user side, it is necessary to build internal understanding of the ML process for long-term stable and independent operation. This includes:

- the ability to evaluate training data regarding structure and quality
- the ability to assess model performance using appropriate metrics and to interpret confusion matrices

It must also be possible to internally decide whether retraining is required and whether the current performance meets production requirements. Deep algorithm-level knowledge is not mandatory; however, a fundamental understanding of the overall ML workflow is required.

Definition of Labels (Ground Truth)

Labeling criteria and ground truth creation should be defined by an employee with strong product and defect expertise. Domain knowledge regarding product characteristics and typical defect patterns is essential, as it forms the basis for valid model evaluation.

The analysis of confusion matrices and KPIs requires analytical skills and the ability to compare AI results with those of conventional inspection process.

Operational Phase

During operational phase, the system should be handled by AI key users. These individuals must be able to:

- interpret confusion matrices
- understand confidence values and interpret XAI heatmaps

In addition, they must detect performance deviations and initiate an internal improvement process if model performance is insufficient. Operators do not need to be AI specialists, but they must understand the evaluation mechanisms of the system to make informed decisions.

Comparison with Conventional Vision Systems

Compared to conventional rule-based vision systems, the focus of required expertise shifts. Traditional systems require:

- extensive knowledge of image processing
- camera and lighting setup expertise as well as manual parameter tuning of thresholds and rules

For complex modifications, support from the vision system supplier is often required. AI-based systems reduce the need for detailed rule-based parameter tuning and specialized image processing expertise. Instead, the focus shifts to:

- data quality and correct labeling
- performance evaluation and structured management of the model lifecycle

Overall, the required expertise shifts from in-depth rule-based image processing toward data-driven model evaluation, performance analysis, and structured management of the AI lifecycle.

1.2 Theme 2: Data and AI

1.2.1 Motivation

This comprehensive workflow addresses the challenge **of industrial data scarcity** stemming from low defect rates or generally small numbers of measurable parts by detailing the required steps, actions and best practices when tackling this by the integration of Digital Twins (physics-based simulations) with real sensor data.

Overcoming these challenges in industrial sensor data necessitates the use of advanced AI/ML methods, particularly Deep Learning (DL):

- **Data Imbalance:** Already established, highly mature production processes result in extremely low failure rates (e.g., in the parts-per-million range). The resulting dataset is often **heavily imbalanced**, predominantly comprising "in-specification" (OK) data. This makes it difficult for standard approaches to understand the underlying relationships between sensor measurements and product performance when tested, leading to poor control ability, especially for the rare "out-of-specification" (NOK) cases.
- **Need for High Precision and Reliability:** For sensitive processes, the predictive QC (Quality Control) system must achieve **high precision and reliability**, often meeting or exceeding the low error margins of physical measurements. Achieving this high level of accuracy and trustworthiness has proven challenging with existing approaches.
- **Computational Bottlenecks:** When simulation models are used, they can be **computationally heavy**, taking hours to complete, which is incompatible with the real-time needs of a high-rate production line. AI/ML is needed to create more efficient models.
- **Both computer-based simulation models and data-based models** derived from measurements rarely cover all process control performance requirements separately, as they **cannot take all influencing factors** into account **at the same time**. It is therefore crucial to develop complementary models.
- **Complex Root-Cause Analysis:** Multi-step manufacturing and assembly processes involve **complex, multilayer cause-and-effect interactions** that are difficult for humans or rule-based systems to capture. AI, particularly DL, is employed to perform **root-cause analysis** by correlating process data with end-of-line quality data, identifying the systemic causes of deviations for process optimization.
- **For new products during the ramp-up phase**, there is usually insufficient measurement data available due to the small number of components already produced to provide definitive data-based control models to meet the quality gates defined in the Production Preparation Process (PPP). Hybrid models based on measurement data from early (pre-)production parts augmented with simulation data can provide control models in time for relevant release dates.

Targeted user group

This simulation-augmented AI approach is primarily designed for a collaborative, multi-disciplinary team within manufacturing organizations. The core user group consists of specialized domain experts who are deeply familiar with the product and process but are **not necessarily AI experts**. This includes **production engineers** and **machine/process experts** responsible for factory-floor operations, and **product development engineers** involved in defining component specifications and simulation models (e.g., the physics-based model). These domain experts will be **supported by data scientists and AI specialists** who focus on model development and optimization. The overall process is structured to bridge the gap between these disciplines. The outcome, developed through a generalized methodological framework, serves as a blueprint to give the entire team a transparent view of the process they are not experts in, enabling them to understand, trust, and contribute their domain knowledge effectively.

1.2.2 Scope

To circumvent this limitation, a generic approach leverages physics-based modeling and simulation to overcome data scarcity. The strategy is twofold:

1. **Hybrid Model Optimization:** Transform existing computationally intensive, physics-based simulation models (representing product or process behavior) into faster, more accurate **hybrid models**. This is achieved by incorporating real production and testing data alongside the underlying physical knowledge.
2. **Synthetic Data Augmentation:** Utilize the optimized, computationally efficient hybrid simulation model as a **synthetic data generator**. This enables the purposeful creation of high-fidelity, labeled data points representing the rare/NOK scenarios that are underrepresented or missing in real-world production measurements.

By systematically augmenting the sparse real data with high-quality synthetic data, the resulting dataset is balanced and enriched. This comprehensive dataset then enables the development of robust, accurate, and trustworthy AI models—such as Deep Learning (DL) or anomaly detection algorithms—for predictive quality control and root-cause analysis, ultimately making these solutions acceptable and scalable in demanding industrial environments.

This guide establishes a structured, use-case-agnostic methodology for developing **credible hybrid AI models** in the manufacturing domain. The approach integrates real-world sensor data with physics-based models to overcome the fundamental challenge of **data scarcity and imbalance** related to rare product defects (NOK parts).

Navigating the Sensor and AI Theme Workflow

The development of high-performing AI models in manufacturing is frequently hindered by the scarcity of data, particularly concerning rare but critical product defects. This guide provides a **step-by-step methodology**, broken down into six major, manageable stages, to navigate this complex process successfully. The primary aim is to furnish members of Product Development, Production Preparation, and Data Science teams with a clear, shared vision and a **functional architecture** for hybrid model creation. Each section is tailored to allow the domain expert (e.g., the Simulation Engineer in Step 2 or the Data Scientist in Step

5) to understand their specific module in depth, while simultaneously explaining the bigger picture, defining the required cross-links and references, and detailing how to assess the information quality received from upstream and delivered to downstream stages. This holistic approach ensures that local experts can effectively adapt and optimize the overall workflow for their unique individual manufacturing conditions.

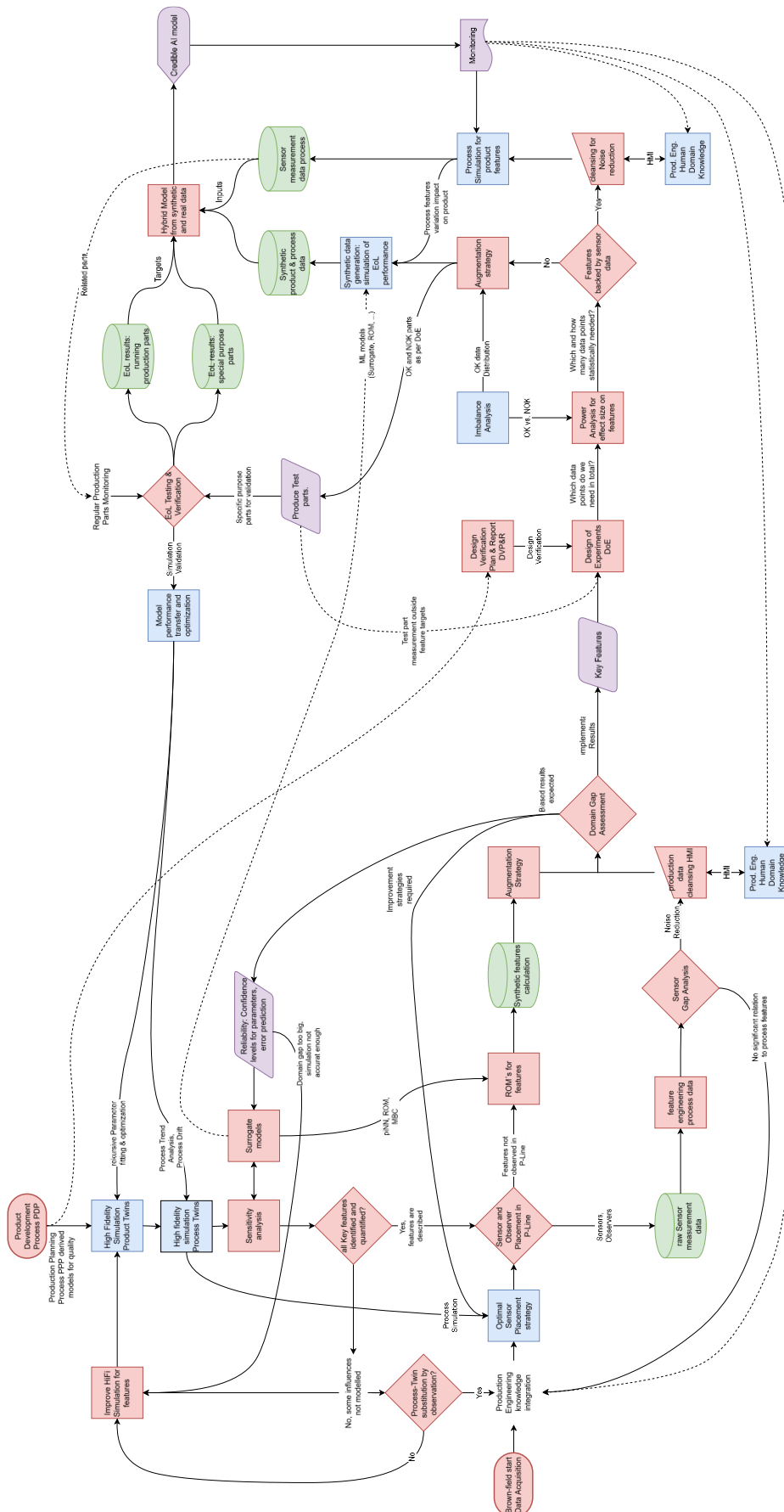


Figure 3 Detailed Functional Architecture for Data and AI Theme

The full functional architecture (modelled in the publicly available <https://app.diagrams.net/>) of the step-by-step guide is published in the SoliDAIR Zenodo community. Please be encouraged to download for better readability and reviewing, and adaptation to your requirements (NB: the workflow and sub-architectures will be described in detail during the step-by-step explanations)

<https://doi.org/10.5281/zenodo.17491810>

The overall functional architecture for developing a Credible Hybrid AI Model is deliberately structured into **six major steps** for **easier understanding** and operational efficiency.

The main steps of the Hybrid AI Model development workflow are:

1. **Data Acquisition and Preprocessing:**
Establishing (Measurement) Data Reliability.
2. **High-Fidelity Simulation and Reduced Order Modeling (ROM):**
Creating the Fast Physics-Based Digital Twin based on Product and Process Simulation(s)
3. **Feature Engineering and Identification of Key Properties:**
Statistical Validation and DoE Definition.
4. **Synthetic Process Data Generation and Validation:**
Managing the data imbalance problem.
5. **Hybrid Model Training:**
Data Fusion and hybrid model development.
6. **Model Performance Evaluation and Continuous Improvement:**
Credibility Assessment and Sustained Performance.

This breakdown serves several crucial purposes:

- it **simplifies complexity** by transforming a multi-domain, recursive challenge into sequential, actionable phases;
- it **improves manageability** by assigning clear ownership and responsibility for each phase; and
- it **facilitates quality control** by establishing defined inputs and verifiable outputs for every transition, ensuring information quality and traceability throughout the entire process.

1.2.3 Data Acquisition and Preprocessing

This foundational stage prepares real-world manufacturing data, focusing on data discovery, cleansing, and assessing sensor coverage in complex, historically grown (“brown-field”) environments.

Step 1

To **understand** the (often) historically grown data landscape, collect raw sensor measurement data, perform feature engineering, conduct Sensor Gap Analysis, and apply cleansing to prepare a reliable, noise-reduced dataset of process features.

Functional Sub-Architecture

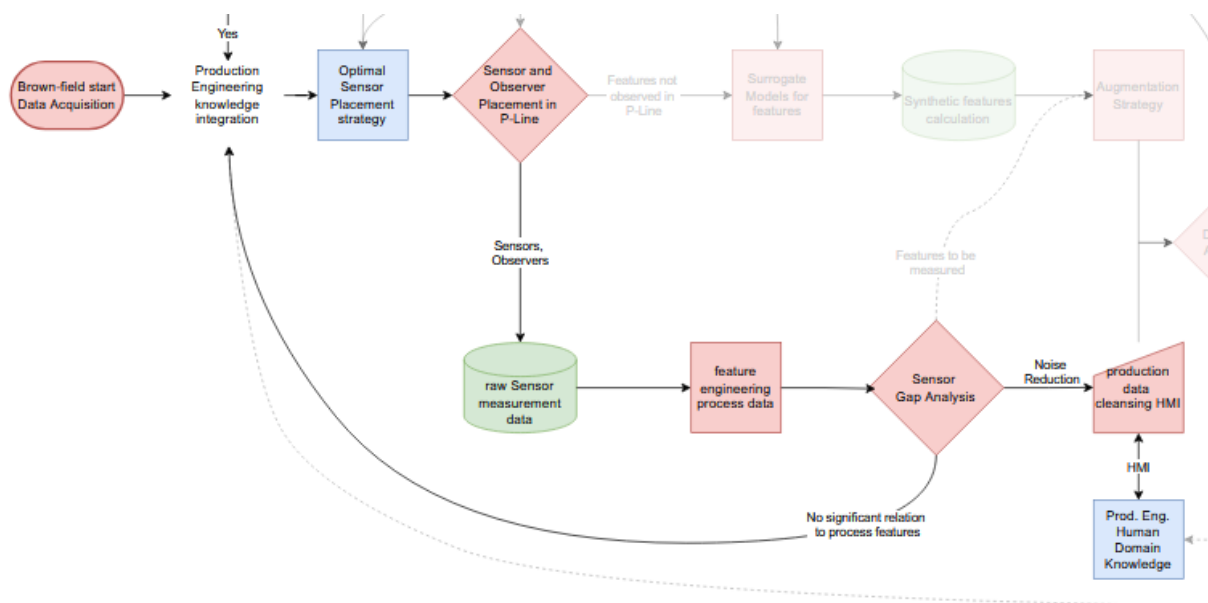


Figure 4 Functional Sub-Architecture of Data Acquisition and Preprocessing step

Detailed Sub-Workflow Description

This step guides the expert through the rigorous process of transforming raw manufacturing measurements into usable data, starting from a historically complex setup.

1. **Data Source Identification and Contextualization:** In a brown-field environment, production lines accumulate various measurements over years, often stored across fragmented systems like historians and legacy Manufacturing Execution Systems (MES). The initial task is to locate all raw sensor measurement data and establish the true purpose, reliability, and accuracy of each variable, as this information is frequently lost or poorly documented during historical buildup. A major technical hurdle is linking time-series sensor data to a specific manufactured part. Since part identification (e.g., a serial number) is often only logged at discrete stations, linking the continuous time-series data streams (from sensors and PLCs) to the part's actual processing window requires complex time-series alignment and interpolation. Furthermore, integrating this operational data with descriptive metadata from Product Lifecycle Management (PLM) systems is necessary to fully contextualize the

measurements. This requires the collaboration of production engineering and data scientists.

2. **Optimal Sensor Placement Strategy** (Module: Optimal Sensor Placement): By analyzing the main influencing factors of the part's performance features, it suggests sensor placement aimed at either direct measurement of the performance feature or indirect measurement via an observer strategy. If direct measurement is not possible due to the nature of the process (e.g., high heat, inaccessible location), this task suggests an Observer Strategy. This involves developing a soft sensor or state estimator (an inference model) to indirectly measure or infer the critical feature from a combination of available, correlated measurements. If either direct or indirect measurement is not feasible, additional physical sensors or software observers are suggested and integrated into the production line. In an information loop, further input comes from the Sensor Gap Analysis task, located later in the workflow, where the significance of the current measurement strategy is evaluated.
3. **In-Line Observability Check**: This critical sub-step determines if the product features necessary for the final quality prediction model are observable during the production process in the line. This is paramount, as subsequent modeling of the influences on part performance can only be made if those influences are captured in the process data. The functions not covered by the process measurements are suitable for later augmentation of the data set through simulation. It is recommended to establish a list/overview of all features to be covered, whether in the process or via the simulation.
4. **Feature Engineering and Standardization**: The raw measurement data is processed by the feature engineering process data module, aimed at consolidating the raw data files into a single, coherent dataset suitable for subsequent preprocessing, analysis, and integration with simulation data in the hybrid model approach. This step ensures that all relevant measurements are uniform in terms of structure, units, and column headers, enabling efficient and reproducible data handling throughout the subsequent stages. This step involves an initial data quality check performed on each data file to verify consistency in terms of, e.g., the number of measured parameters and column headers, which enables a smooth and accurate joining of the datasets. It also includes techniques such as data synchronization (via event triggers, timestamps) and performing preliminary basic statistical analysis (mean, variance) over defined process windows to transform raw sensor outputs into meaningful process features.
5. **Sensor Gap Assessment**: The output features are subjected to Sensor Gap Analysis, which assesses their statistical correlation to established product quality parameters. If the analysis detects "No significant relation to process features", this indicates the current sensing infrastructure is inadequate. A feedback loop is then triggered to the **Production Engineering knowledge integration** module to revise the feature definition or identify unmeasured critical parameters. The identification of key features represents a crucial step in the development of a highly accurate ML/AI model. This step ensures that only relevant, informative, and high-quality parameters are used to train the prediction algorithms, thereby improving model accuracy, interpretability, and robustness.
6. **Noise Reduction and Cleansing** (Module: Human Domain Knowledge): Validated features proceed to the production data cleansing HMI. This human-in-the-loop module uses dashboards (e.g., Mendix) to facilitate manual review, allowing domain experts to identify and remove outliers, address missing values, and validate the

quality of the engineered features before the data is passed to the next step. This step ensures the dataset is as concise as possible and significantly reduces computation resources, as insignificant influences can be identified and eliminated.

The **Input** is raw, high-volume, often fragmented and/or time-series data, and the **Envisaged Target** is a structured, validated dataset of process features.

Challenges and Design Questions

Challenges:

- Data discovery and understanding require a significant amount of time due to historically fragmented data sources.
- Interoperability with legacy operational technology (OT) systems remains a major technical hurdle.
- Managing data volume, velocity, and variety (Big Data) necessitates robust streaming and distributed processing.
- Part Traceability Challenge: unambiguous allocation of process states data to the production part, especially when process data is streamed as time series, is a major challenge (“What exact temperature did this part see during its process?”). Working with time series during feature extraction and later in building the hybrid model is not recommended. When time series are unavoidable, try linking the time series data to the respective features.
- Keeping manually composed production data (such as Excel files) accurate.

Design Questions:

- Which statistical metrics should the Sensor Gap Analysis use to definitively flag a feature as having "No significant relation"?
This depends heavily on product design and robustness, hence collaboration with product engineers is recommended.
- Can a product performance feature be identified and controlled by sensor measurement? Which factors really influence the product performance during production and or/by the design?
A (hybrid) model can only predict what is in the data. It is a (very) common misconception (and hope) that product failures can be predicted by analyzing production data, when the cause is, e.g., in a tolerance combination that is not measured.

Technical Requirements for Infrastructure:

Every production environment has its own specific requirements for the IT infrastructure, which stem from technical reasons but may also involve historical legacy components. Therefore, only a very generic and exemplary overview of the involved infrastructure can be given.

Generally speaking, data about the produced goods is typically generated in a multitude of places (e.g., originating from different manufacturing steps conducted by various production equipment). This data needs to be stored in a central location, and the related data sources are joined to form (at least) one consistent large data set. This dataset serves as the starting point for all subsequent steps.

To make the data actually useful for a specific analysis, an extract, transform, and load step is implemented, and the specific analysis is conducted. The output of this analysis could be added to the original dataset as an additional feature. Only after that step can advanced methodologies, such as machine learning, be applied meaningfully. Exemplary software stacks for conducting these tasks are briefly outlined below.

Open Source stack:

With the wide availability of powerful open source tools IT infrastructure could involve the following: to connect (multiple) data sources (like SQL-based production databases) with data consumers Apache Kafka (real-time streaming/data bus) and/or Apache NiFi (big data processing) might be used and the fused data sets might be stored in SQL-based databases. For data processing and generic extract, transform and load (ETL) tasks Python with Pandas and NumPy (feature engineering) libraries can be employed and combined with StreamLit to provide a graphical user interface (GUI) so that the developed software becomes accessible for non-experts. For machine learning, several Python-based toolkits are available as open source software, namely TensorFlow, PyTorch and Scikit-Learn.

Proprietary software stack:

An even wider variety of proprietary software exists for all aspects related to data handling and processing and a full discussion of options is beyond the scope of this report. Siemens Digital Industries Software is part of SoliDAIR; from their portfolio Teamcenter Manufacturing (process data planning/management) and Insights Hub/Mendix (for data ingestion/visualization) might be suitable options. For data analysis and visualization often MS Excel is used as a starting point, but there exist many proprietary solutions like Originlabs, Tableau, KNIME, Stata etc. For machine learning mostly open source based libraries are used also in commercial software stacks.

Explanation of Deployment Steps

- **Setup OT/IT Bridge:** Install and configure ingestion connectors (e.g., Kafka Connectors or NiFi processors) to automate the flow from OT systems to the data lake.
- **Containerized Feature Pipeline:** Develop feature extraction scripts (Python) and deploy them as containerized microservices for consistent execution across environments.
- **Sensor Gap Analysis Implementation:** Implement the statistical analysis module (Python/SciPy) and define the feedback loop to Production Engineering knowledge integration.

AI Usage and Explanation

- **AI Module:** Automated Feature Selection (within feature engineering process data) and Statistical Correlation Models (within Sensor Gap Analysis).
- **Explanation:** AI/ML techniques (like clustering or classification algorithms) can be used to perform ML-driven feature selection, identifying features with the highest

predictive value for known OK/NOK states. This proactively addresses the curse of dimensionality, focusing system resources only on statistically relevant features derived from the massive sensor data pool.

1.2.4 High-Fidelity Simulation and Reduced Order Modeling (ROM)

This step requires a product performance theory, which is developed during the Product Development Process (PDP). This theory, expressed in mathematical models, establishes the **basis for setting tolerances** and the acceptable limits **on all relevant features** (product and process). Based on these models, High-Fidelity Simulations HFS can be developed, covering the impact of product features on the product performance (Product Twins) and the process features on the product features (and consequently on the performance).

Step 2

The aim of step 2 is to provide a **physics-consistent synthetic data generation engine** to purposefully produce specific data to augment the measured dataset, utilizing surrogate models and/or **Reduced Order Models (ROMs)** to achieve this in a computationally efficient way.

Functional Sub-Architecture

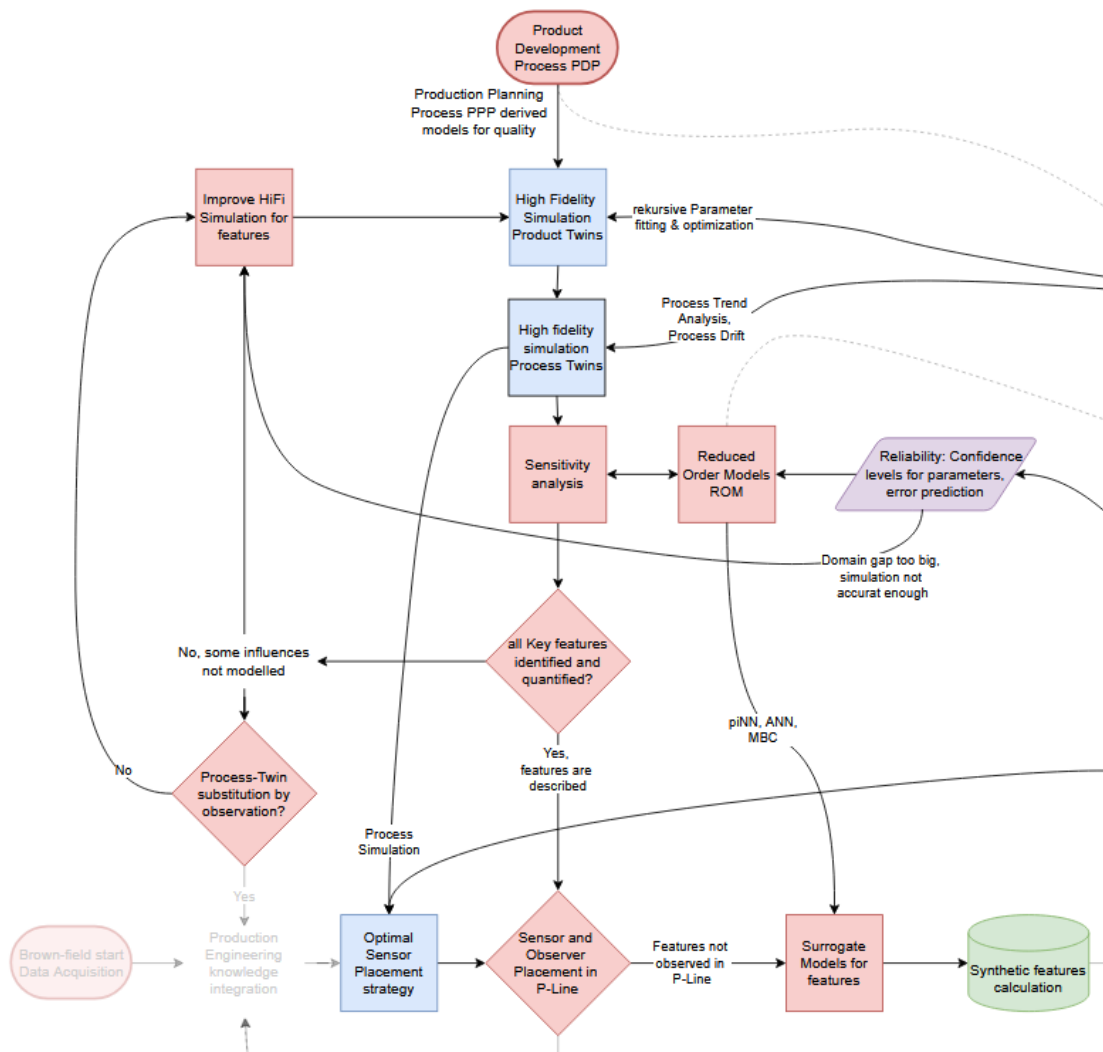


Figure 5 Functional Sub-Architecture of HFS and ROM step

Detailed Sub-Workflow description

1. Setup HFS Product Twins:

The High-Fidelity Simulation (HFS) uses **mathematical and physics-based models** to describe the **product performance features and their dependencies** on the required detail and accuracy. The HFS can also be adapted in a **data-driven fashion**, by for example parameter or system identification. The HFS is used to simulate all possible product features over their possible variations (depending on their tolerance definitions and limits setting). While State-of-the-Art variation can be covered by Design-of-Experiments (DoE) methods or statistical variation (Monte-Carlo), these concepts reach their computational limits with increased product complexity and/or the required combination of different physics domains. In this case, further specialization utilizing Reduced Order Models ROM and Surrogate Models is required (see next sub-tasks). Note that usually resulting from the PDP there is a typical set of features (parameter) variation simulation available, e.g., to proof that worst-case tolerance combinations are still within the targeted performance limits.

2. Setup HFS Process Twins:

The relation between process parameters (process features) and the influenced product features but not necessarily their impact on product performance is modelled, as this step is covered by the Product Twin. Same requirements as per Product Twins apply. The HFS Process Twins allow to monitor, predict, and simulate the internal state evolution of a Product, for example in time or frequency domain. The HFS Process Twins allows to capture complex internal interaction between hidden states/control actions of the subsystems, at every “instance” of the progress.

3. Conduct Sensitivity Analysis:

Utilizing the Product and Process Simulations as input, the next step is to conduct a Sensitivity Analysis (e.g., using Simcenter HEEDS) to quantify how changes in input parameters (features) affect performance, which is crucial for defining the training space (in later steps) for the simplified models. As an outcome, a clear picture and weighed list of features influencing the product performance (target features) is established. Secondly, a list of features where no significant relation is determined, may be used for complexity and noise reduction (see later steps).

4. Features Check from Sensitivity Analysis:

Although it seems obvious, but in reality, not all influencing factors are always considered in simulations. There can be many reasons for this, e.g., modelability within a tool, or existing expert know-how regarding their influence on product performance. Nevertheless, a check for completeness of the weighted feature list represented in the simulation is required, otherwise the HFS must be adapted accordingly.

5. Process-Twin Check:

An observation check on process influences (Process-Twin substitution by observation?) is performed and compared to the features list of the sensitivity analysis to understand if all relevant features are modelled in the process twin HFS. If the check fails, it indicates the HFS may not adequately cover key process physics, and a direct loop to **Improve HFS for features** is triggered for recalibration.

6. Reduced Order Model ROM development (Module: Reduced Order Models for sensor data):

ROMs for features are trained to replicate the HFS behavior. The training, typically performed using tools like Simcenter ROM Builder or Altair PhysicsAI, often employs methods like Neural Networks, Bayesian surrogates, or Regression Trees. Methods like Proper Orthogonal Decomposition (POD) or Principal Component Analysis (PCA) help in determining the dominant features amongst the list of all original features, to be used in the training of the ROM. Based upon the Sensitivity Analysis results for the weighted list of features, there should be models built for every product feature over their full parameter variation possible (defined by the tolerance window definition and the statistical quality requirement – see later step “Power Analysis”, which describes on much outside the tolerance window (NOK) variation is required).

ROM are built by first generating exciting data that cover a large range of features and targets (either from simulation or from real-world measurements). For tabular data (i.e. Product performance prediction), the ROM building tool can leverage those data to train a list of ML learning models, giving the user a scalable and quick way of testing and validating the different resulting ROMs. However, such an approach remains model-centric, as the focus is entirely on the ML model. Instead, we also propose a data-centric approach, which tackles the data-generation process. A

closed-loop data-generation approach to prepare the training dataset for ROMs. Two situations can be handles: 1) active learning to build-up a training dataset while continuously evaluating the effect of such a training dataset on the performance of the ROM, and iterating using a sensitivity-aware optimizer (Simcenter HEEDS), or 2) optimal data selection to down-select a large, pre-generated dataset (from simulation or from real-world measurements), into the smallest possible dataset that optimizes the performance of a particular ML model, evaluated on objective test datasets. This renders the training dataset not universal but rather application specific to exploit maximal performance. Additional techniques are also presented within SoliDAIR to reduce the dimensionality of predicting sequential data (i.e. those concerning process data), to rapidly obtain a multi-step response from the set features. We refer to them as low-order embedding of the sequence.

7. **Reliability Check, Error Prediction and Confidence Levels:**

This check **quantifies the uncertainty** in the ROM's predictions. Methods like cross-validation and residual analysis are used to calculate the mean prediction error and error bounds. Confidence levels for parameters are determined, often using Bayesian methods or Gaussian Process Regression, to estimate the reliability of the ROM when interpolating new data points. This task primarily relies on the built-in validation and export features of dedicated tools like Simcenter ROM Builder (which ranks models using performance metrics) and Python libraries (e.g., Scikit-learn or PyTorch) for advanced uncertainty quantification (UQ) models.

The quantification results indicate if the order reduction and hence a loss of accuracy is acceptable and/or if there should be separate models for the individual features developed, as it is a trade-off between accuracy and computational requirements on one side and accuracy over all covered features on the other side. However, related features must be covered by a single model. As further inputs for the Reliability check the results from the Hybrid Model and Evaluation (step 5) and from the Domain Gap Assessment (Step 3) are to be considered (see descriptions of step 3 and 5).

Accuracy maps can also be built in tools like Simcenter Testlab Process designer, to evaluate the performance of the ROM in comparison with the recorded data, and in comparison with the HFS. Those accuracy maps allow the user to visually pinpoint regions of lower-accuracy (due to interaction between the features), and understand the validity region of their ROM.

8. **In-Line Observability Check** (additional Overlay of Step 1):

This critical sub-step determines if the product features necessary for the final quality prediction model are observable **during the production process in the line** (as described in step 1).

From the Sensitivity Analysis, a comparison between the (weighted) features to be measured and observed in the production line (via their product features relation) and the product and process features from the HFS is made. This is relevant for the validation of the features of the HFS (see step 5: Hybrid Model and Evaluation)

9. **Surrogate Model Creation:**

Since running the full HFS can take an unacceptable amount of time on a high-performance computing (HPC) cluster, surrogate models provide a low-cost approximation. These are data-driven ROMs - statistical or machine learning models - that are trained to rapidly approximate the input-output behavior of a complex, computationally expensive HFS. They treat the HFS as a "black box."

Techniques include regression techniques like Polynomial Response Surfaces,

Kriging (Gaussian Processes), and various Artificial Neural Networks (ANNs). The model is built entirely from a set of input (features of the product and process) and output (product performance) data pairs generated by running the high-fidelity simulation a limited number of times. This approach ensures consistency with the later hybrid modelling approach, where real (measurement data) is augmented with the synthetically (computed) datasets provided.

The final ROM must be exported to a **tool-agnostic format** (e.g., FMI or ONNX) for use in the computationally inexpensive online augmentation module. The Envisaged Target is an accurate, exportable ROM model, typically achieving 95% fidelity (the exact acceptable percentage should be derived in the Domain Gap Assessment sub-step) with a significant **computational speedup**.

10. Augmentation Strategy:

This sub-step connects the HFS knowledge to the data gaps identified in Step 1. By comparing the list of all **relevant features** (established by the product performance theory) with the **measurable features** (validated in Step 1), the **Augmentation Strategy** indicates precisely which missing features must be covered by synthetic simulation to create a balanced, complete training set. This distinction ensures simulation resources are optimally utilized to fill critical knowledge gaps, primarily focusing on NOK scenarios.

Challenges and Design Questions

Challenges:

- **Availability of HFS for product and process** covering all performance-determining product features:
Often, only aspects of the product are being simulated, e.g., a specific sub-structure within a specialized tool or simulation domain, to understand its behavior. Linking this specific simulation tools via co-simulation environments (e.g., Simulink, Dymola, SimulationX, Amesim, and Ansys, as well as open-source options like the [open-source Python package PyFMI](#), OpenModelica, and [FMIGo!](#)). The usage of domain agnostic exchange formats like FMI is key to ensure interoperability.
- **Validation of the HFS** is often only done with the nominal values of the involved features, or based on existing physical models, representing a specific tolerance situation. This usually does not allow an extrapolation to the defined limits of the feature, although it is common practice. Improvement of the HFS is required to be accurate enough (see step 5: Validation and Improvement) to meet the Domain Gap and Accuracy requirements.
- **High upfront offline training costs** for ROMs; maintaining accuracy of the ROM within a defined error tolerance; the preference for non-intrusive ROMs which treat the HFS as a black-box.
- **Sensitivity Analysis** does not provide a clear picture of influential variables and features:
Complex products with many cross-interdependencies between features blur the picture for the product performance key factors. Usually, acceptable tolerances tend to get strictly limited, resulting in product cost increase. This is considered as a critical situation, as it does also indicate that the product design is prone to uncontrolled and

unforeseeable behavior which is difficult to link to specific root causes.

If the product design must be considered a given, it is recommended to deal with this situation by not prematurely removing factors that do not appear to have a major influence from the ROM derivation or considering them as noise. Unfortunately, this does increase the effort required for synthetic data creation.

- **Availability of training data:** in model-centric ML development, training data is often assumed to be abundantly available, or cheap to generate. However, this is false. Large amount of data does not necessarily indicate higher accuracy ROMs, particularly if the large amount of data fail to uncover scarce situations, or are too generic (i.e. not dynamically exciting to cover a large domain of operation). Moreover, generating data can be dangerous (real systems), sometimes impossible (for e.g., on real operating manufacturing lines), and costly/non-scalable (for complicated HFS). We treat this challenge through our closed-loop, optimization-based training data meta-learning pipeline.

Design Questions:

- Which dimensionality reduction technique (e.g., Proper Orthogonal Decomposition, POD, or Neural Networks) is best suited for the HFS data?
- How do we ensure the ROM maintains acceptable fidelity percentage while achieving a significant computational speedup?

Analysis of Technical Requirements for Infrastructure

- **Proprietary:**
Simcenter Amesim (1D/multiphysics system simulation for HFS);
Simcenter 3D (multiphysics CAE, pre/post-processing for 3D simulation);
Simcenter ROM Builder (dedicated platform for ROM creation using AI/ML methods and NN).
AVL CAMEO (for integrating HFS/ROM into testing/optimization);
AVL CRUISE M (multi-disciplinary system simulation).
- **Open Source Stack:**
Python with libraries like PySR (Symbolic Regression) for alternative ROM creation.

Explanation of Deployment Steps

1. **Offline ROM Training:** Run Design Space Exploration using tools like Simcenter HEEDS to generate a comprehensive input-output dataset from the HFS.
2. **Model Reduction:** Apply methods (e.g., Neural Networks within Simcenter ROM Builder) to find a low-dimensional latent space.
3. **Generic Export:** Export the trained ROM to a tool-agnostic format (e.g., **FMI** or **ONNX**) for use in the online augmentation module.

AI Usage and Explanation

- **AI Module:** Surrogate models and ROMs for features.
- **Explanation:** ROMs are AI-based surrogate models used to predict complex physical phenomena in quasi-real-time. They enable optimization routines to test thousands of design configurations, which would be impossible with the full HFS. ROMs also leverage physics-based principles, allowing them to work effectively with smaller datasets compared to purely data-driven models.

1.2.5 Feature Engineering and Identification of Key Properties

Step 3

To arrive at a **complete composition of key features** by harmonizing measured process data with synthetically described features, validating the total set for statistical power, and identifying parameters that contribute as noise for subsequent removal.

Functional Sub-Architecture

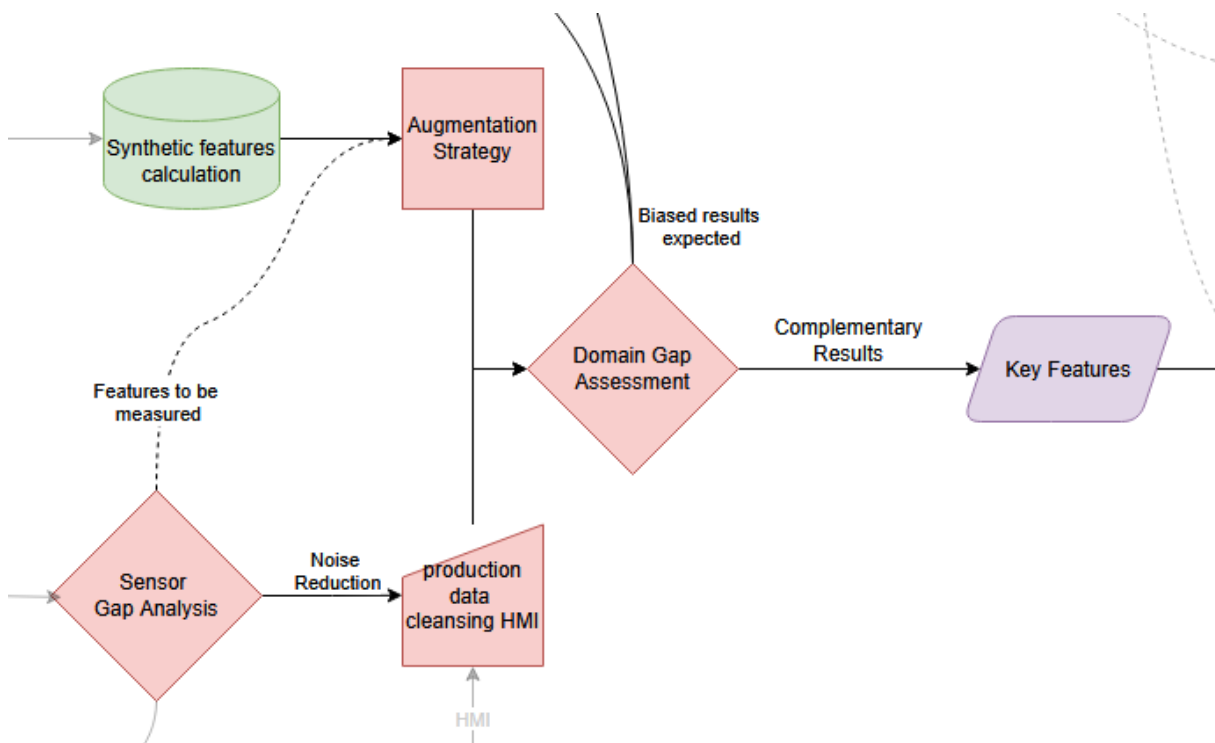


Figure 6 Sub-Architecture for Feature Engineering

Detailed Sub-Workflow Description

1. Synthetic Feature Calculation:

The ROMs are already based on the features depicted in the sensitivity analysis and provide simulation results by varying the identified features. By applying the same

feature engineering steps as in the real data (step 1), **data consistency** can be ensured.

2. **Augmentation Strategy Execution:**

This task identifies the gaps in the features represented in the existing cleaned dataset (primarily OK samples, but also from features dependent on the design or those not measured in the production line). It checks for the possibility of augmentation from synthetic feature calculation, covering missing features from measurements, as well as **extensions from measurements outside the target windows, that is, NOK samples**.

This involves merging the newly calculated synthetic features (NOK samples) with the existing cleaned real data (primarily OK samples) to create the first iteration of a balanced training dataset.

3. **Data Cleansing (Re-check):**

The merged dataset passes through the production data cleansing HMI. Although the real data was previously cleansed, this step serves as a validation check for the combined dataset, enabling domain experts to identify any obvious anomalies or inconsistencies introduced during the synthetic data generation and merging process.

4. **Domain Gap Assessment (DGA):**

The resulting dataset is subjected to the Domain Gap Assessment (DGA), a critical **validation checkpoint**. This task determines the difference in **statistical distribution** between the real-world data (OK and NOK) and the combined (OK + synthetic OK + synthetic NOK) data. By synthetic re-calculation of real-world data, the gap is statistically quantified.

- a. **Establishing Gap Size:** The acceptable gap size is directly dependent on the accuracy requirements of the final model. For safety-critical applications (e.g., aviation, medical), the DGA must enforce extremely strict tolerance bands (e.g., or specific Kolmogorov-Smirnov test p-values). For non-critical predictive maintenance, wider gaps may be acceptable. The threshold must be agreed upon by Data Scientists and Product Development teams.
- b. **Quantification Methods:** The DGA primarily uses statistical distance metrics:
 - i. Maximum Mean Discrepancy (MMD): Measures the distance between two probability distributions (real vs. synthetic).
 - ii. Kolmogorov-Smirnov (KS) Test: Checks if the synthetic data's feature distribution significantly differs from the real data's distribution.
 - iii. Unsupervised Anomaly Detection: Runs simple clustering or Isolation Forest algorithms to check if the synthetic data points fall completely outside the natural hyper-sphere of the real data.
- c. **Feedback Loop Action Proposals:** If the DGA determines the gap is too large, it suggests the synthetic data generation parameters may be flawed, resulting in "Biased results expected". The following actions are proposed, depending on the application scenario:
 - i. Parameter Recalibration (Simulation/Step 2): The primary action is to route feedback to Step 2 (Improve HFS for features), requiring the simulation engineers to recalibrate HFS parameters or adjust the ROM training domain. This is done when the bias is attributed to fundamental model inaccuracy.
 - ii. Feature Re-evaluation (Step 1): Propose the inclusion of previously defined noise parameters if the DGA indicates that a missing, un-

modeled real-world variable is contributing to the distribution shift. These parameters must be re-categorized from 'noise' to 'influential feature' and added back to the synthesis process.

- iii. **Modeling Technique Validation:** Suggest changes in the synthetic generation method itself (e.g., switching from Gaussian Process Regression to Deep Generative Models or applying explicit noise modeling to the synthetic outputs).
5. **Key Features Finalization:** The process concludes with the definition of Key Features. This final feature set is the definitive input for the training steps in Step 5, ensuring only reliable, validated, and statistically powerful variables are used to build the hybrid AI model.

The **Input** is the ROM model output (synthetic data) and the clean real data, and the **Envisaged Target** is a merged, validated, and quality-assured hybrid feature set.

Challenges and Design Questions

- **Challenges:** Ensuring data consistency between real and synthetic data (e.g., maintaining the same noise floor or distribution characteristics) is highly challenging. The Domain Gap Assessment must use rigorous statistical distance measures to quantify model risk. The process must mitigate the risk of the model learning "Biased results" from flawed synthetic data.
- **Design Questions:** What are the acceptable statistical distance thresholds (e.g., Maximum Mean Discrepancy, MMD) for the Domain Gap Assessment? How should the production data cleansing HMI handle high-volume synthetic data efficiently without requiring manual review of every point?

Analysis of Technical Requirements for Infrastructure

- **Open Source Stack:**
Python with Scikit-learn and specialized libraries (e.g., Scipy) for calculating domain distance metrics and statistical analysis.
- **Proprietary:**
MindSphere/Mendix (HMI interface and workflow for data verification),
AVL Model Factory (for managing simulation data versions and ensuring traceability of synthetic data parameters).

Explanation of Deployment Steps

1. **Synthetic Data Processing:** Deploy the feature calculation scripts using the same containerized framework as Step 1 to ensure environmental consistency.
2. **Merging and Cleansing Workflow:** Orchestrate a job that automatically merges data and triggers the Domain Gap Assessment script.
3. **Threshold Enforcement:** Define and enforce automatic thresholds at the Domain Gap Assessment checkpoint, routing output that fails the validation to data scientists for re-parameterization of the ROM.

AI Usage and Explanation

- **AI Module:** Domain Gap Assessment (statistical/ML anomaly detection).
- **Explanation:** The **Domain Gap Assessment** often employs **unsupervised learning** or statistical process control (SPC) algorithms to detect subtle shifts in data distribution. This AI checkpoint ensures that the synthetic data remains statistically aligned with the real world, preventing the hybrid model from being trained on physically unrealistic data.

1.2.6 Synthetic Process Data Generation and Model Validation

Step 4

To define the Design of Experiments (DoE) plan based on statistical needs, generate synthetic data to cover knowledge gaps, physically produce test parts based on the DoE, and execute EoL Testing and Verification to close the Sim2Real gap.

The Input is the statistical plan, and the envisaged target is **validated physical ground truth data** and a **verified simulation model**.

Functional Sub-Architecture

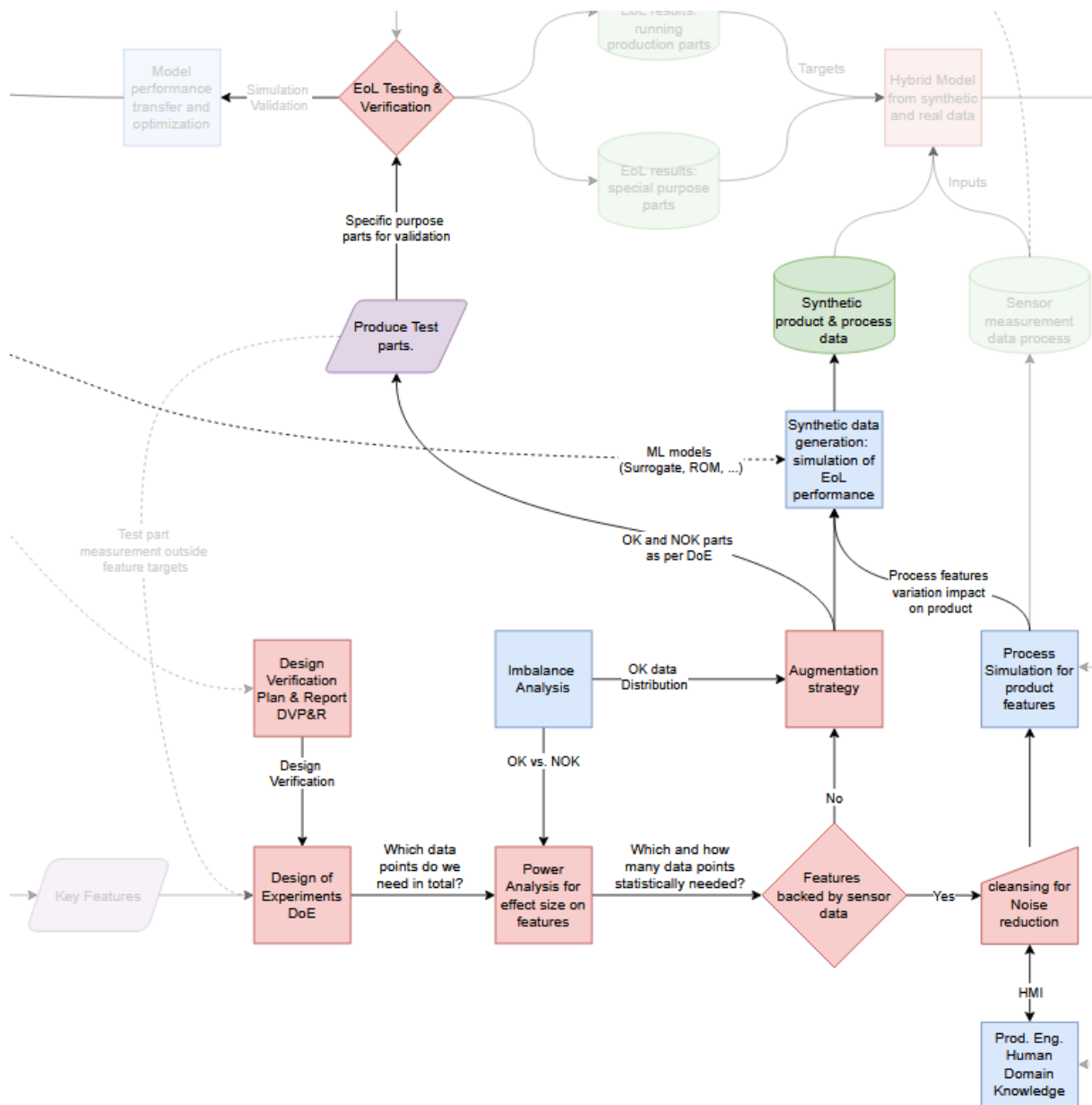


Figure 7 Sub-Architecture for Synthetic Data Generation and Validation

Detailed Sub-Workflow Description

1. Statistical Requirement Definition:

The sub-workflow begins with the analysis of the Design Verification Plan & Report (DVP&R), which formalizes the statistical needs identified by the preceding Imbalance Analysis and Power Analysis for effect size on features (from Step 3). This DVP&R, derived from the Product Development Process (PDP), outlines the validation strategy for product performance, including all product configurations to be evaluated (critical configurations and worst-case scenarios). This is often based on tolerance studies conducted during the PDP, which define allowed variations in key features. This DVP&R process is aimed at confirming the tolerance definitions of the key

features and making sure that the product over all feature variations will be still meeting the product requirements.

2. **DoE Planning and Execution:**

The Design of Experiments (DoE) task takes the DVP&R and uses the statistical results from Power Analysis for effect size to calculate the minimal, optimized evaluation configurations required ("which configurations do we need to check?"). The DoE defines not only all worst-case configurations for parameters, which mark the boundary conditions for NOK and OK parts, but also important variations of key features that may be dependent on other features. This step delivers the test matrix for all feature variations and hence the required data points, regardless of whether gathered by measurement or provided by synthetic provision.

3. **Data Imbalance Analysis** (module: Data imbalance analysis):

The main purpose of this step is to provide a clear and quantitative understanding of the extent of data imbalance, while also characterizing the distribution of the OK parts. This task includes basic calculations to identify the class ratio, defined as the relation between the size of the minority class (NOK parts) and the majority class (OK parts). Beyond this, this step also evaluates whether the OK data sufficiently covers the sampling space or if specific operational conditions are underrepresented or missing. For this purpose, statistical methods can be employed to identify sparsely populated regions, clusters, and data gaps. The resulting class ratio and distribution analysis support subsequent tasks, specifically the Power Analysis and the Augmentation Strategy task, particularly when focusing on augmenting OK data to ensure full coverage of the sampling space and improve the robustness of downstream ML and AI models.

4. **Power Analysis:**

This result, combined with the Data Imbalance Analysis (from Step 3), allows the Power Analysis to calculate the exact number of required datapoints at all relevant configurations to be statistically significant, providing the precise volume of synthetic datapoints to be provided to deal with data imbalance. This calculated plan is handed to the Produce Test parts task.

5. **Features backed by sensor data:**

This step compares the result from the Power Analysis which delivers all required configuration including the required data volume on each data point and compares it with the available datasets gathered from real measurement. The gap from this analysis is determining the configurations which need to be provided by synthetic data generation.

6. **Augmentation Strategy:**

In the subsequent Augmentation Strategy, these complementary synthetic configurations are defined for practical conditions. Due to the situations, for these configurations no direct validation from real measurements is available, and for the credibility requirements at least some datapoints which may be produced synthetically, the underlying models must be validated (and parameterized) with specifically produced test parts to confirm their accuracy. For practical reasons and as not all product features can be produced exactly to these conditions (cost), it is required to adapt this validation plan and define the parts which can be produced for the purpose of model parameterization and validation.

7. **Physical Experimentation:**

The Produce Test parts task manufactures these specific parts (either OK and NOK

parts as per DoE and augmentation plan). This is a costly, but necessary, physical step to generate ground truth data in the critical NOK parameter space where HFS and subsequently ROM validity is weakest.

(NB: if enough NOK data were available, especially around the acceptable tolerance boundaries, the data imbalance would not exist, and the models could be trained with traditional methods. Furthermore, it must be emphasized that the target is NOT to produce enough parts to be measured and train the model directly, which probably is the much more costly way, but to validate and parameterize the HFS and ROM's to these conditions, which is not always required, depending on the situational validity of these models).

8. Process Simulation for product features

The synthetic data generation is initiated by batch processing the variations of the process features identified in the prior steps. The ROMs/Surrogates are executed to produce synthetic data of product features across the defined parameter space variations of the process features. This output is a critical input for the following sub-module.

9. Synthetic Data Generation: Simulation of EoL performance

Using the synthetic product features calculated in the previous step and resulting from the DoE, this module proceeds to simulate the End-of-Line (EoL) testbed performance of these components (predicting their functionality or failure mode if they were real). This entire synthetic data process is performed by batch processing of the feature variations defined in the previous steps. For eventual validation, some of these synthetically described components will be produced on purpose, linking back to the physical process.

10. EoL Measurement:

The physically produced parts undergo EoL Testing & Verification. This produces EoL results: special purpose parts (validation data) and EoL results: running production parts (real operational data).

11. Sim2Real Validation:

The results from EoL outputs: special purpose parts are fed back to the Domain Gap Assessment to be compared directly against the synthetic data created by the ROM. This comparison represents the crucial Sim2Real validation. If the test part measurements fall outside the expected synthetic feature targets, it signals an error in the simulation model itself, triggering the feedback loop (Improve HFS in Step 2).

Challenges and Design Questions

- **Challenges:**

The primary risk is the cost and time associated with manufacturing the test parts exactly to the specifications, which is not always even possible due to their manufacturing processes. Smart definitions of such conditions with the aim of validating the model, rather than producing actual functional parts is crucial. However, if those variations in parameters may happen during production and their impact on product performance is relevant, either the models need to be validated, or the influence must be confirmed by testing for stable production. Furthermore, parameterization of the simulation models (HFS and ROM's) to all varying parameter conditions covering the features might be challenging, depending on the validity of the models.

- **Design Questions:**

What is the most efficient DoE topology (e.g., factorial, response surface) to cover the critical NOK parameter space with the minimum number of physical test parts?
How can the validation data acquisition tools (EoL Testing) be standardized to match the data structure of the synthetic data?

Analysis of Technical Requirements for Infrastructure

- **Proprietary**

Simcenter HEEDS (accelerates DoE planning);
Tecnomatix (process validation and planning for Produce Test parts)
AVL CAMEO (manages the test sequence and data acquisition for EoL Testing);
AVL Model Factory (for managing test data and parameters)

Explanation of Deployment Steps

1. **DoE Tool Integration:** Integrate the DoE tool (e.g., Simcenter HEEDS) directly with the HFS/ROM environment to automatically generate parameter sets for testing.
2. **Testbed Automation:** Automate the physical manufacturing and testing process (Produce Test parts) as much as possible to minimize human error and time per test cycle (e.g., by open EoL approaches)
3. **Data Feed:** Ensure the EoL Testing environment automatically pushes EoL results data directly into the central data lake for DGA analysis (Step 3).

AI Usage and Explanation

- **AI Module:** DoE Planning.
- **Explanation:** The DoE is increasingly governed by Active/Sequential Learning algorithms which act as an AI optimizer for experimentation. These algorithms use previous test results (real and synthetic) to intelligently propose the next most informative test point, dramatically reducing the quantity of expensive physical tests required.

1.2.7 Hybrid Model Development and Training

Step 5

To train the **Hybrid Model** using the merged real and synthetic dataset and perform **Model performance transfer and optimization** to achieve a **Credible AI model**.

Functional Sub-Architecture

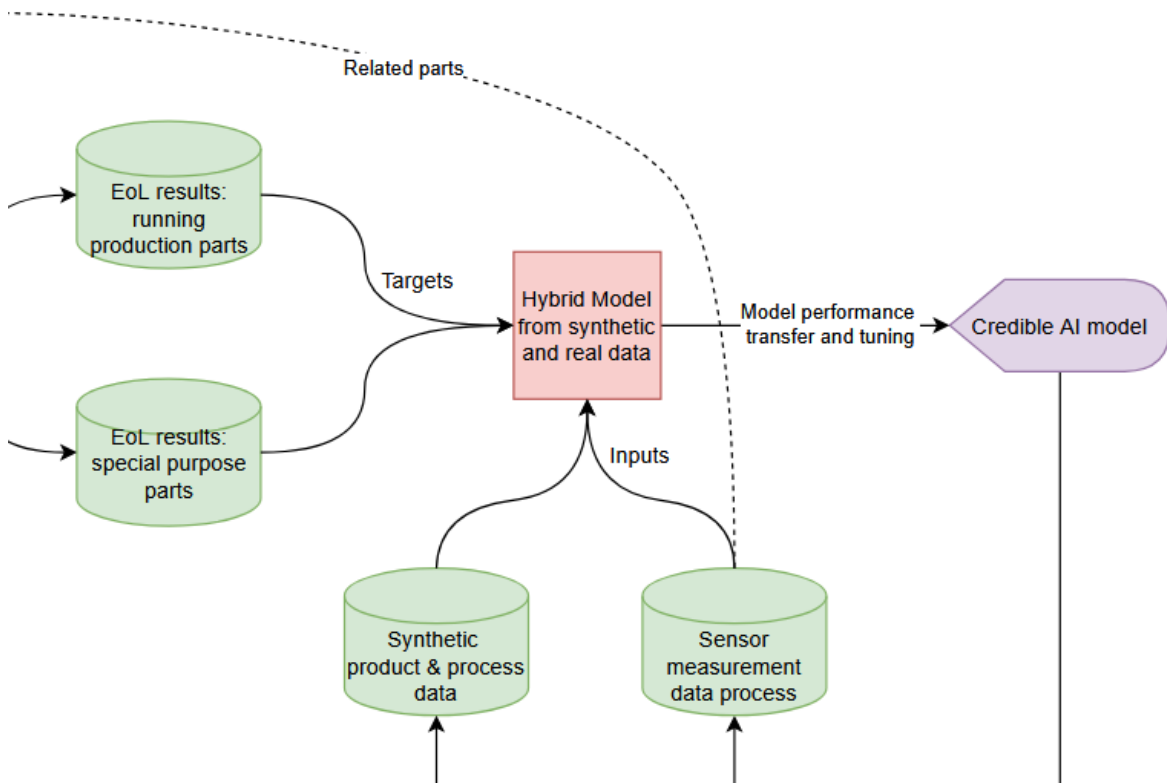


Figure 8 Sub-Architecture for Hybrid Model Development and Training

Detailed Sub-Workflow Description

1. **Hybrid Model from synthetic and real data** (Module: Hybrid model):
The validated Synthetic product & process data and the Sensor measurement data process (real data) are combined to form the input for the Hybrid Model, as the targets for the hybrid model is provided by the Eol results, both from the real measurements as from the synthetic performance simulation. Train the ML model directly on this combined dataset.
Key Consideration: You may need to assign a **weighting factor** to the samples in the loss function to control the influence of the potentially noisier synthetic data.

Alternative approach:

Pre-training on Synthetic Data (Transfer Learning) is the most common technique; however, in some cases, it may be a less accurate approach, especially when a large

amount of synthetic data is available, but it is an imperfect approximation of the real data. Train a large ML model (e.g., a deep neural network) exclusively on the **synthetic input and synthetic target** data. This allows the model to learn the core structure, features, and relationships very quickly without overfitting on limited real-world samples.

Replace the model's final output layer (or simply adjust its weights) and then retrain it using the real input and target data (Fine-Tuning). A small learning rate is used to adjust the previously learned weights slightly, adapting the model to the nuances of the real distribution. This is the most common approach, as it requires significantly less real data to achieve high performance.

2. **Credible AI model**

The outcome is the Credible AI model, verified against reliability parameters and ready for deployment.

Challenges and Design Questions

Challenges

- Achieving model sophistication to capture complex, non-linear relationships. Weighting factors may need to be included in the prediction model to control the influence of the potentially noisier synthetic data.
- Ensuring data consistency between real and synthetic sources (same features, scaling, format).
- Ensuring the model is interpretable (XAI).
- Managing the recursive Parameter fitting & optimization feedback loop between the ML model and the costly HFS.

Design Questions:

- How can XAI be integrated to verify that the model is learning physics-consistent relationships from the synthetic data?
- What metrics define the "Credible" status for low-risk applications?

Analysis of Technical Requirements for Infrastructure

- **Open Source Stack:**
ML frameworks (PyTorch, Keras, TensorFlow) for training. Python (Scikit-learn) for classic ML algorithms.
- **Proprietary:**
Simcenter HEEDS (for Model performance transfer and optimization); Simcenter ROM Builder (ROMs can be directly used as components in a hybrid architecture).
- **Compute:**
GPU resources are typically required for training Deep Learning-based hybrid models and optimization routines.

Explanation of Deployment Steps

1. **Data Unification:** Merge the Synthetic product & process data and Sensor measurement data process into a single, structured, and balanced training set.
2. **Credibility Check:** Final verification that the model's accuracy, reliability (confidence levels for parameters), and XAI transparency meet the defined risk thresholds.

AI Usage and Explanation

- **AI Module:**
Deep Learning and ML methods are the core of developing the model; neither approach, from pure measurement nor simulation alone, is able to provide the necessary functionality.

1.2.8 Model Performance Monitoring and Continuous Improvement

The architecture defines the continuous operational loop: Credible AI model Monitoring Process Simulation for product features Prod. Eng. Human Domain Knowledge. This establishes a live performance and tracking system that updates the central knowledge base and triggers potential refinements to the HFS or cleansing steps.

Step 6

To deploy the Credible AI model and implement continuous Monitoring and Process Simulation for product features to maintain model validity and adaptation to possibly changing conditions.

Functional Sub-Architecture

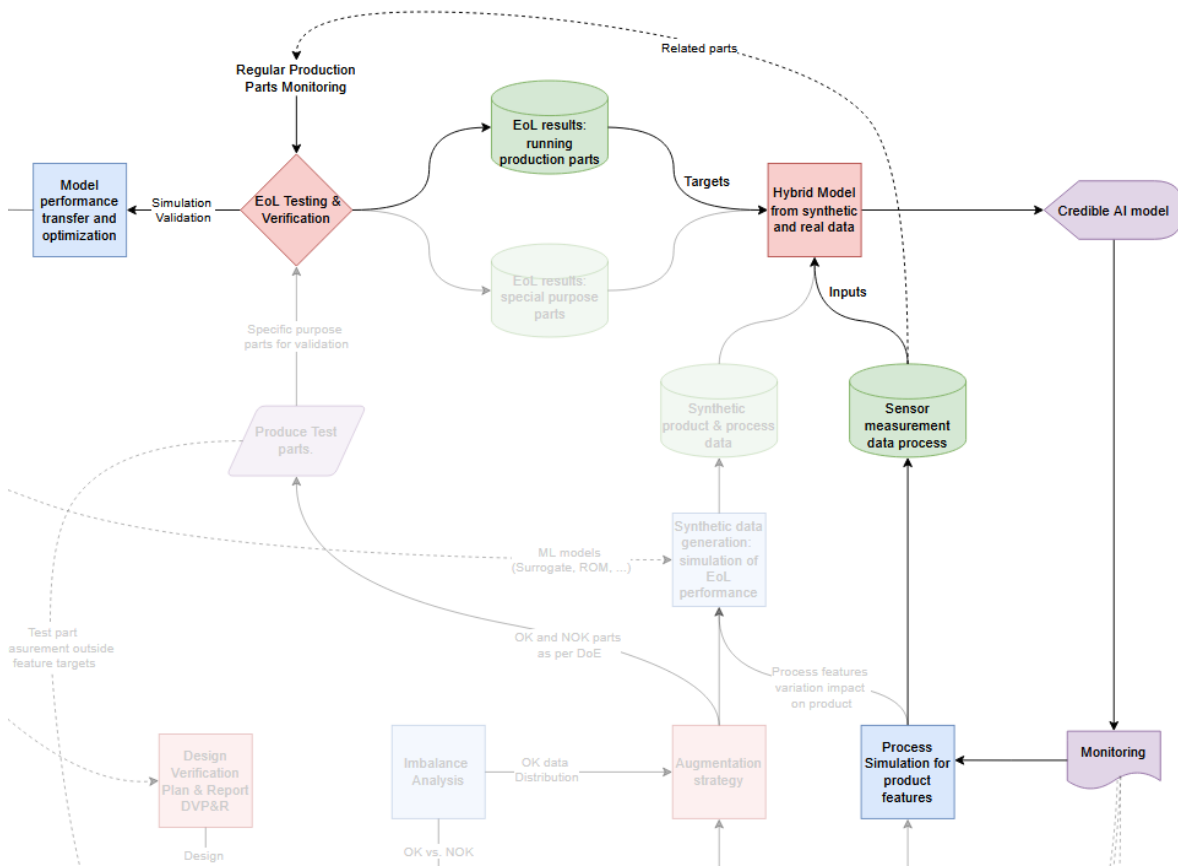


Figure 9 Sub-Architecture for Performance Evaluation and Continuous Improvement

Detailed Sub-Workflow Description

The Credible AI model is deployed into the production environment and immediately enters the Monitoring phase. The Monitoring system tracks inference metrics, confidence levels, and Process Trend Analysis based on live sensor data. The monitoring data feeds into the continuous Process Simulation for product features (a lightweight model often based on the ROM/Surrogate from Step 2), which helps diagnose drift by comparing observed performance against expected simulation bounds. The critical diagnostic information is then fed into the Prod. Eng. Human Domain Knowledge module (via HMI), updating the central knowledge base (dashed line). This module acts as the crucial human-in-the-loop for validating system drift and initiating costly reruns of the DoE or HFS refinement. The **Input** is live process data, and the **Envisaged Target** is sustained predictive accuracy and traceable decision-making.

Challenges and Design Questions

Challenges:

- Achieving response times for real-time inference at the edge.
- Detecting and isolating model drift over time.
- Ensuring audit logging and compliance for AI decisions.

- Managing the high-stakes loop for updating human domain knowledge.

Design Questions

- What latency is acceptable for real-time decision-making in the production line?
- How can XAI explanations be utilized in monitoring to clarify when and why issues occur?

Analysis of Technical Requirements for Infrastructure

- **Open Source Stack:**
Uptime Kuma/Nagios Core (for technical infrastructure monitoring);
ELK Stack (Elasticsearch, Logstash, Kibana) for log and event management.
- **Proprietary:**
MindSphere/Mendix (centralized IoT platform and application development for monitoring dashboards and HMI).
AVL CAMEO (for continuous optimization and monitoring in test environments);
AVL Model Factory (for managing the deployed model and data).
ERP systems in general to monitor and control production equipment
- **Deployment Location:**
Often requires edge deployment (on-premise infrastructure) to achieve low latency.

Explanation of Deployment Steps

1. **Edge Deployment:**
Deploy the containerized model (Docker/K8s) to on-premise infrastructure near the sensor data source to meet latency requirements.
2. **Observability:**
Implement real-time logging of feature attribution scores (LIME/SHAP) alongside technical and business metrics.
3. **Human Feedback Loop:**
The Monitoring system flags high-risk or low-confidence predictions to the Prod. Eng. Human Domain Knowledge module via HMI, allowing experts to validate the decision and manually enrich the knowledge base (dashed lines in the flow chart).

AI Usage and Explanation

- **AI Module:** Monitoring (using XAI).
- **Explanation:** Explainable AI (XAI) may be used to provide transparency. It could generate feature attributions for every prediction, allowing operators to understand why a part was flagged as NOK.
This is essential for:
 - 1) Trust and Accountability and,
 - 2) Root Cause Analysis by linking the prediction directly back to the process features (e.g., "Feature X was 70% responsible for this defect").

1.2.9 Available Datasets and Code

For the Data and AI theme, two datasets are planned to be made fully available by the end of the project. So far, one dataset is already published, and the other is partially available. The SoliDAIR project will also, in the future, publish open-source code for the use of the AI community.

- SoliDAIR journal bearings dataset: in tabular form, it contains data on the performance of journal bearings under various operating conditions. The data is mostly representative of mixed lubrication, as it is the precursor of the most common modes of bearing failure. The dataset consists of two parts; the first, already published in Zenodo ([Link](#)), contains the experimental data from various tests using a bearing test rig. The second part, which will be published before the end of the project, will contain synthetic data to augment and enrich the experimental dataset. This journal bearings dataset will serve the main purpose of demonstrating the development and implementation of the hybrid model approach.
- SoliDAIR quality prediction dataset UC BOS: This dataset, already published in Zenodo ([Link](#)), comprises an anonymized subset in tabular form, containing sensor measurement data from a high-volume, high-rate automotive production line for fuel injectors. This highly imbalanced dataset, containing a very small number of rejected (NOK) parts, serves the main purpose of training and evaluating the performance of classification and regression algorithms under severe data imbalance.

The code for the hybrid model module, which is the main component of the Data and AI theme, will be openly shared with the community via a GitHub repository and may also be published and referenced on other relevant platforms, such as the SoliDAIR Zenodo community. This repository will include a complete and functional pipeline for implementing the hybrid model approach, involving two data sources: the experimental/production dataset and the simulation dataset, the integration of the data imbalance analysis module, and an overview of prediction performance using basic ML models with different ratios of synthetic simulation data added. To make the code more accessible, it will be developed as an interactive web application using Streamlit.

1.2.10 Expertise needed

To implement the hybrid AI workflow described, a multi-disciplinary team is required, bridging the gap between the factory floor and advanced data science.

Data Acquisition and Preprocessing

Focus: Bridging the OT/IT gap and ensuring "Ground Truth" reliability in brown-field environments.

- **OT/IT Integration:** Expertise in connecting legacy PLC/SCADA systems to modern data lakes using protocols like MQTT, OPC UA, or tools like Apache Kafka/NiFi.
- **Industrial Data Engineering:** Skill in aligning high-frequency time-series sensor data with discrete part serial numbers (traceability).
- **Domain-Driven Feature Engineering:** Ability to translate raw physical signals (vibration, thermal, pressure) into meaningful process features based on machine physics.

- **Statistical Analysis:** Proficiency in Python (Pandas/SciPy) to perform Sensor Gap Analysis and correlation studies against quality parameters.
- **HMI/Dashboard Development:** Skill in building low-code interfaces (e.g., Mendix/Streamlit) for "Human-in-the-loop" data cleansing and validation.

HFS and Reduced Order Modeling (ROM)

Focus: Converting heavy physics simulations into "Fast Twins" that can generate synthetic data at scale.

- **Multiphysics Simulation:** Expert knowledge in CAE tools (Simcenter, Ansys, or OpenModelica) to build High-Fidelity Simulations (HFS) for product performance.
- **Model Order Reduction (MOR):** Skills in using POD, PCA, or Neural Networks to compress complex simulations into lightweight ROMs or Surrogate Models.
- **Sensitivity Analysis:** Expertise in Design of Experiments (DoE) to identify which process variables most significantly impact product quality.
- **Software Interoperability:** Technical ability to export models into tool-agnostic formats like **FMU/FMU** or **ONNX** for cross-platform integration.
- **Uncertainty Quantification (UQ):** Ability to calculate confidence intervals and error bounds (e.g., Bayesian methods) to ensure the ROM is a "credible" proxy for reality.

Feature Engineering and Identification

Focus: Harmonizing real and synthetic datasets while ensuring statistical alignment.

- **Synthetic Data Processing:** Expertise in running ROMs in batch mode to generate "labeled" NOK (defect) scenarios that are missing in real production.
- **Advanced Statistics (Domain Gap):** Ability to use metrics like **Maximum Mean Discrepancy (MMD)** or **Kolmogorov-Smirnov** tests to quantify the "Sim2Real" gap.
- **Data Architecture:** Skill in orchestrating automated pipelines that merge fragmented real-world measurement files with structured simulation outputs.
- **Anomaly Detection:** Proficiency in unsupervised learning (Isolation Forests, Clustering) to detect if synthetic data points are physically unrealistic.
- **Root Cause Reasoning:** Collaboration between data scientists and production engineers to re-categorize "noise" into "influential features" based on DGA feedback.

Synthetic Data Generation and Validation

Focus: Statistically-backed data augmentation and physical "Sim2Real" verification.

- **Statistical Power Analysis:** Ability to calculate the exact volume of data points required to overcome class imbalance and achieve statistical significance.
- **Advanced DoE Planning:** Expertise in sequential/active learning to minimize the number of expensive physical test parts needed for validation.
- **Manufacturing Execution:** Skill in coordinating the production of specific "out-of-spec" parts to provide ground-truth data for the most uncertain model regions.
- **EoL Testing Expertise:** Proficiency in configuring End-of-Line (EoL) testbed hardware and software (e.g., AVL CAMEO) to capture validation data.
- **Feedback Loop Management:** Ability to diagnose simulation inaccuracies by comparing EoL physical results with ROM predictions (Sim2Real validation).

Hybrid Model Development and Training

Focus: Fusing physics-based knowledge with data-driven ML.

- **Deep Learning & Transfer Learning:** Mastery of frameworks like PyTorch or TensorFlow, specifically in pre-training models on synthetic data before fine-tuning on real data.
- **Loss Function Engineering:** Skill in implementing custom weighting factors to balance the influence of "clean" simulation data vs. "noisy" real sensor data.
- **Hyperparameter Optimization:** Expertise in using tools like Simcenter HEEDS to automate the search for the most accurate model architecture.
- **Explainable AI (XAI):** Ability to implement LIME or SHAP to ensure the model's logic is "physics-consistent" and understandable to non-AI experts.
- **Compute Resource Management:** Experience in managing GPU-accelerated environments for training complex hybrid architectures.

Performance Monitoring and Improvement

Focus: Sustaining model credibility through real-time observability and human oversight.

- **MLOps & Edge Deployment:** Skills in containerization (Docker/Kubernetes) to deploy models directly onto the factory floor for low-latency inference.
- **Real-time Monitoring:** Expertise in tracking model drift, data distribution shifts, and system latency using tools like the ELK stack or Nagios.
- **Diagnostic Engineering:** Ability to use lightweight ROMs in parallel with live AI to diagnose the root cause of quality deviations.
- **Collaborative Decision Making:** Skill in translating XAI "feature attribution" scores into actionable insights for production managers and process experts.
- **Continuous Learning Integration:** Ability to design the "human-in-the-loop" workflow where expert feedback is fed back into the central knowledge base for model retraining.

Understanding the Interdependency

The success of this workflow relies on "translation" points between these roles. For example, the **Simulation Engineer** provides the physics, but the **Data Scientist** must ensure that physics is compressed into a format the production line can run in real-time.

The table below provides an overview of these interdependencies for each of the steps.

Workflow Step	Production / Process Engineer	Product / Simulation Engineer	Data Scientist / AI Specialist
Data Acquisition	Identifying raw sensor sources in "brown-field" lines; part traceability.	Contextualizing measurements against product specs.	OT/IT bridge setup; Time-series alignment; HMI dashboarding.
HFS & ROM	Defining operational constraints for process twins.	Building High-Fidelity Simulations (HFS); ROM compression; Sensitivity Analysis.	Implementing ROM training algorithms; Exporting to ONNX/FMI formats.
Feature Engineering	Validating "noise" vs. "influence"	Domain Gap Assessment (DGA)	Statistical distance metrics (MMD/KS);

	based on floor experience.	feedback; Parameter recalibration.	Synthetic feature calculation.
Generation & Validation	Managing the production of physical test parts (OK/NOK).	Sim2Real validation; Adjusting DVP&R based on power analysis.	Statistical Power Analysis; DoE topology optimization; Active learning.
Hybrid Training	Establishing "Credibility" thresholds for the factory floor.	Ensuring XAI results align with known physical laws.	Transfer Learning; Loss function weighting; Hyperparameter tuning.
Monitoring & CI	Root-cause analysis via XAI; Human-in-the-loop validation.	Updating simulations based on observed process drift.	MLOps; Edge deployment; Monitoring model drift & latency.

1.3 Theme 3: Robotics and AI

1.3.1 Motivation

Industries stand to gain significant advantages by integrating robotic systems into quality control (QC) applications, primarily due to enhanced **precision, repeatability, and data integration**. Unlike human inspectors, robotic systems, particularly when coupled with advanced machine vision, 3D laser scanners, or contact probes, can execute inspection tasks with unparalleled consistency, 24/7, eliminating subjective judgment and operator fatigue. This allows for the detection of small defects and verification of dimensional tolerances far beyond human capability.

Furthermore, automated QC systems generate a continuous, high-volume stream of precise data, which is essential for robust **Statistical Process Control**. This data enables real-time process monitoring, root cause analysis of defects, and predictive quality trends, allowing manufacturers to move from simple defect detection to proactive defect prevention and optimize overall production quality. This integration also increases **throughput** by performing in-line inspections at high speeds, removing QC as a production bottleneck.

1.3.2 Scope

This guide presents a comprehensive, use-case agnostic methodology for the complete lifecycle of an industrial automation project, from concept to commissioning, guided by the goals of the SoliDAIR project. The framework is not limited to a single application but provides a general process applicable to any robotic task. The scope begins with a systematic methodology for robotic platform selection, covering the analysis of generic task requirements (e.g., payload, reach, precision), data collection and AI model training, the critical trade-offs between industrial and collaborative systems, and the imperative for protocol compatibility with existing factory hardware (e.g., PLCs).

To highlight the SoliDAIR project's impact, this guide also proposes a depth-aware sizing methodology. While this methodology is inherently use-case agnostic, it is highly applicable to tasks such as defect detection. Rather than relying on established practices that use a static pixel-to-millimeter ratio, this approach couples **AI-driven** bounding box detection with robotic path planning. By mapping localized depth-level calibrations at repeatable inspection poses, the system dynamically applies the correct spatial ratio to estimate physical defect sizes in millimeters.

It then details the essential virtual commissioning phase, focusing on the development of high-fidelity Digital Twins (DTs). This section addresses the universal challenges of validating cell logic, optimizing cycle times, and bridging the "sim-to-real" gap, regardless of the specific application.

Finally, the guide provides a technical breakdown of the physical hardware integration and control architecture, covering critical steps from End-of-Arm Tooling design and Tool Center Point calibration to path planning and the implementation of high-speed positional triggers for precise process synchronization.

1.3.3 Requirements Identification and Robotics Platform Selection

Step1: Parts and Process Invariants

The foundation of any robotic workcell begins with a rigorous characterization of the physical

"invariants", the elements of the process that are fixed constraints. This requires a granular attribute analysis of the part that needs to be inspected, specifically the maximum and minimum dimensions, weight, and material properties like surface reflectivity or fragility, which directly dictate handling strategies and sensor selection.

Precisely define the inspection task: this includes identifying the type of inspection (e.g., 2D/3D machine vision, non-destructive testing), the nature and size of the defects to be detected. Simultaneously, the **process requirements** must be quantified; this includes the **takt time** (the maximum allowable time per cycle to meet production demand), the required coverage percentage (e.g., 100% surface inspection vs. random sampling), and the specific **tolerance limits** (e.g., +/- 0.05mm positioning accuracy). This analysis directly informs two critical specifications for robots: payload and reach. The payload must be calculated not just for the part being inspected (if handled), but as the sum of the End-of-Arm Tooling, which includes the camera, laser scanner, or sensor, plus any required cabling and fixtures. The robot's reach must be sufficient to access all inspection points on the part, considering the cell layout and any potential obstructions. Defining these non-negotiables upfront prevents the selection of hardware that is physically incapable of meeting the core production metrics.

Step 2: Robot Mechanics

Once the payload and reach requirements are derived from the part data, the robotic manipulator must be selected to withstand the dynamic forces of the application. Beyond basic **reach** (working envelope) and **payload** (which must account for the combined mass of the part, the End-of-Arm Tooling, and cable dress packs), engineers must prioritize **structural stiffness** and **repeatability**. For precision tasks, a robot with low backlash and high rigidity is essential to minimize settling time and jitter. Finally, the **IP rating** must be matched to the environment; a standard IP40 robot will fail in a machining cell requiring IP67 protection against cutting fluids and dust.

Following this, determine the required **precision**. Quality control tasks often demand high-precision robots, so you must verify that the robot's repeatability (e.g., ± 0.02 mm) is significantly finer than the smallest feature or defect being measured. This is closely tied to the **degrees of freedom (axes)**; a 6-axis articulated robot offers high flexibility for inspecting complex geometries, whereas a 4-axis SCARA robot might be faster and sufficient for simpler, planar QC tasks.

The choice between a **collaborative robot (cobot)** and a traditional **industrial robot** is a critical next step. Cobots are suitable for applications requiring human collaboration, flexibility, and easy programming, but they are inherently slower and have lower payloads. Industrial robots are the standard for high-speed, high-payload applications where the inspection cell can be fully guarded for safety. The selection of a traditional industrial robot, as opposed to a collaborative system, necessitates the implementation of a **guarded workcell** and a significantly larger **spatial footprint**. This approach requires the installation of **perimeter hard guarding** (i.e., safety fencing) to completely enclose the robot's **maximum work envelope**, the full volume of space the robot's end-effector can reach. This guarding creates a mandatory **exclusion zone**, which must remain inaccessible to personnel during operation to mitigate collision hazards. Consequently, the total cell layout must account for this safety clearance; for example, an industrial robot with a 1,800 mm reach (1.8m) could easily require a 3m x 3m fenced perimeter to be compliant and safe.

Step 3: Sensing

The sensing architecture serves as the system's feedback loop and must be tailored to the specific optical and physical properties of the target. For visual data, the choice involves

selecting between **2D area cameras** for feature detection, **line-scan cameras** for continuous high-resolution inspection, or **3D systems** (structured light, laser triangulation, or Time-of-Flight) for volumetric analysis. However, the sensor is only as good as the **optical path**: lighting must be engineered (e.g., dome, darkfield, or coaxial) to maximize contrast, and appropriate lenses must be used to provide the correct **Field of View (FOV)** and **Working Distance (WD)**. Crucially, hardware triggers must be defined to synchronize image acquisition with robot motion to eliminate motion blur.

Step 4: Data Acquisition

Building a production-grade AI system requires a dataset that rigorously represents the manufacturing environment rather than a simple collection of images. Because industrial lines suffer from **class imbalance** (predominantly "OK" Samples), the acquisition process must prioritize the systematic collection of "Not OK" (NOK) variations. Using the selected sensing equipment (cameras, lasers etc.), you should capture defects from multiple angles and poses, aiming for a baseline of at least **100 samples per class**. However, prioritize **intra-class variance** over sheer volume; avoid redundant duplicates and instead focus on capturing the complete morphological range of irregularities (e.g., varying scratch depths, surface textures). If physical samples are scarce, augment the dataset with **synthetic data** generated via the Digital Twin (see section 1.3.4). Next, annotate the data with precision ensuring, for example, that object detection bounding boxes **tightly enclose** defects to minimize background noise. Finally, partition the dataset into **Training (60-80%)**, **Validation (10-20%)**, and **Testing (10-20%)** subsets.

Crucially, you must maintain strict **data independence** to avoid "data leakage," a common pitfall where the model effectively cheats. If you capture multiple images of a *single physical part* with a specific defect (e.g., five different angles of "Scratch A" on "Part #101"), all those images must reside in the same data bucket (e.g., all in Training). **Never** split images of the exact same physical object between your Training and Testing sets. If you do, the AI model will merely memorize the unique visual signature of that specific part rather than learning the abstract features of a "scratch." By keeping them separate, you guarantee that your test results genuinely measure the model's ability to **generalize** to completely new, unseen parts on the production line.

Tools: Data collection can be executed using a flexible range of software platforms depending on the hardware architecture. For integrated industrial solutions, proprietary vendor suites (e.g. **Hikrobot MVS**) provide robust, out-of-the-box acquisition control, whereas **Python** coupled with **OpenCV** serves as a powerful open-source alternative for custom development. Following acquisition, the data can be labeled using any standard **annotation tool** that supports the required output formats.

Reference datasets suitable for model training and testing are readily available through the **SoliDAIR Zenodo community** repository, providing data scientists with verified examples of acquired data (see section 1.3.6)

Step 5: AI Model Selection and Training

This step is about choosing the right "brain" for your robot and teaching it how to make decisions. The goal is to pick a model that is smart enough to find the defects but fast enough to keep up with the production line.

Choosing the Right AI Architecture

Think of the AI model as a tool. You need to select the right tool based on what you need to know about the part:

- **For Simple Checks (Classification):** If you only need a "Yes/No" answer—like "Is the label present?" or "Is the cap screwed on?", an option is **Image Classification** models (like ResNet). These are lightweight, very fast, and look at the whole image to put it in a bucket: "Good" or "Bad."
- **For Detailed Inspection (Detection & Segmentation):** If you need to know *where* a defect is (e.g., "Where is the scratch?") or *how big* it is (e.g., "Is this dent larger than 2mm?"); **Object Detection** (like YOLO) or **Instance Segmentation** are more **appropriate**. These models are more complex because they draw a box around the defect or trace its exact outline. They require more computer power but provide detailed quality data.

Transfer Learning

Teaching an AI from scratch requires millions of images and weeks of time. To speed this up, you can use **Transfer Learning**. Imagine hiring a mechanic who already knows how engines work in general; you only need to train them on your specific car model. Similarly, we take an AI that has already "seen" millions of generic objects (learning what edges, shapes, and textures look like) and then fine-tune it using your specific dataset. This allows the AI to learn your defects much faster and with far fewer images.

Balancing the Risks: FAR vs. FRR In a factory, "Accuracy" (e.g., "I got 95% of questions right") is often a misleading score. Instead, we have to tune the AI based on what kind of mistake is more expensive for your business. You must balance two specific risks:

- **False Acceptance Rate (FAR):** This is the risk of the AI saying a bad part is "Good." This is dangerous because a defective product reaches the customer.
- **False Rejection Rate (FRR):** This is the risk of the AI saying a good part is "Bad." This causes waste because you end up scrapping or re-inspecting perfectly good products.

The Trade-off: You typically cannot have zero of both. A decision must be made: *Is it worse to accidentally ship a defect (High Risk), or is it worse to throw away good materials (High Waste)?* You need to tune the model's sensitivity to match your business priority.

Tools: For the model training phase, the **Open-Source** ecosystem offers powerful, industry-standard solutions. **Python** and **PyTorch** are used for a huge variety of AI training tasks. The **Ultralytics YOLO** Python framework is highly recommended for training and deploying state-of-the-art object detection networks due to its ease of use and performance.

Step 6: Interfaces

A robust industrial system relies on a segregated communication architecture that balances control speed with data volume. The **fieldbus** (e.g., **EtherCAT**, **EtherNet/IP**, **PROFINET**) acts as the deterministic "control plane," handling real-time motion commands and I/O states with millisecond-level cycle times. Separately, the "data plane" handles high-bandwidth sensor information via protocols like **GigE Vision** or **USB3 Vision**, ensuring image transfer doesn't clog the motion controller. For higher-level coordination, protocols like **OPC UA** and **MQTT** are specified to bridge the Operational Technology (OT) with IT systems (MES/Cloud).

This means that choosing a robot that is compatible with your facility's existing industrial equipment, particularly the **Programmable Logic Controllers (PLCs)**, is critical for

seamless integration and operational efficiency. When a new robot "speaks the same language" as your current systems, using common industrial protocols like EtherNet/IP or PROFINET, it can be integrated directly into the established control architecture. This allows the PLC, which acts as the "brain" of the production line, to directly manage the robot's tasks, coordinate its movements with other machinery, and enforce unified **safety logic** (like E-stops). Conversely, selecting an incompatible robot forces costly and time-consuming custom integration, requiring gateways and complex "handshaking" logic that is difficult to program, debug, and maintain, ultimately creating an isolated data silo rather than a cohesive automated system.

Step 7: Safety & Compliance

Safety is not an add-on but a fundamental design constraint that dictates hardware selection and cell footprint. A comprehensive **Risk Assessment** (per **ISO 12100**) determines the necessary Performance Level (PL) and Category (typically **PLd/Cat.3** per **ISO 13849-1**). This decision drives the choice between a collaborative approach (**ISO/TS 15066**) which utilizes force-limiting cobots allowing for fenceless operation but at reduced speeds, or a traditional industrial approach (**ISO 10218**) utilizing hard guarding, safety scanners, and light curtains to isolate high-speed hazards. This step defines the "safety hardware," including safety PLCs, interlocks, and Safe Torque Off (STO) requirements.

1.3.4 Simulation and Digital Twin

Step 1: Initiate Digital Twin Development

Before integrating a newly acquired robot onto the production line, leveraging simulation software to create a Digital Twin (DT) is an essential step for virtual commissioning. By using either the robot manufacturer's proprietary software (like **FANUC RoboGuide** or **KUKA.Sim**) or a specialized third-party platform (such as **Siemens Simcenter Amesim** or **NVIDIA Isaac Sim**), you can build a high-fidelity virtual model of the exact robot and its intended workcell. This DT allows your team to conduct extensive offline programming and experimentation in a safe, virtual environment. You can thoroughly validate robot reach, test logic, identify and correct potential collision points with fixtures or other machinery, and accurately optimize cycle times, all without interrupting the existing production flow or risking damage to the physical robot, ensuring the seamless integration.

Step 2: Robot + End-Effector Models

The construction of the digital twin begins with the definition of the physical assets using standard description formats like **URDF** (Unified Robot Description Format), **SDF**, or **USD**. This step goes beyond visual geometry; it requires the precise specification of kinematic chains and dynamic properties. You must incorporate accurate **inertia tensors**, **mass**, and **center of mass** for every link and the specific End-of-Arm Tooling (EOAT), as these values govern the physics engine's ability to simulate torque and gravity compensation correctly. Additionally, a rigorous **self-collision matrix** must be defined to prevent the virtual robot from passing through itself, setting the baseline for valid path planning.

Step 3: Calibration Package

To align the virtual sensor data with physical reality, a two-tiered calibration architecture is implemented. First, **Intrinsics** are characterized for every optical sensor; this involves calculating the lens distortion coefficients (radial and tangential) and focal lengths to rectify raw

images or laser profiles into a linear model. Second, **Extrinsics** are defined to establish the spatial relationships between rigid bodies. This results in a transform tree (**TF tree**) that mathematically links the camera to the tool (hand-eye calibration), the tool to the robot flange (TCP), and the robot base to the world frame, ensuring that a pixel detected by the camera can be accurately mapped to a coordinate in the robot's workspace.

In advanced quality control use cases, detecting the existence of a defect is sometimes insufficient. Systems must accurately measure the defect's exact size to enforce strict engineering tolerances. However, when inspecting objects with complex 3D geometries, defects can occur at various depth levels relative to the camera, which inherently distorts standard 2D pixel measurements. To solve this, beyond standard intrinsic and extrinsic calibration, the system requires localized depth scale calibration.

Once the optimal path planning is calculated to ensure 100% optical coverage of the part, the robot moves the camera to each programmed inspection pose. At each specific stop, the user of the system identifies the necessary depth levels where defects could potentially occur. The user then pinpoints and measures the physical dimensions of static elements (such as specific cavities or machined edges) located precisely at those targeted depth levels. By correlating these known physical measurements with their pixel dimensions in the image, a localized pixel-to-millimeter ratio is established for each specific depth level within that single FoV. This creates a spatial lookup table tied to that exact robot pose, preparing the system for accurate real-time defect size measurement.

Tools: For the critical task of sensor characterization, **OpenCV** provides a comprehensive suite of algorithms and standard procedures designed to ensure accurate intrinsic and extrinsic calibration of visual sensors.

Step 4: Time Alignment

High-fidelity modelling requires not just spatial accuracy but temporal synchronization. This step involves modelling the latencies and offsets between disparate data streams. The simulation must account for the time delta between a hardware **trigger**, the sensor **exposure** interval, the timestamp of the robot's state (joint encoders), and the encoder values of external axes like conveyor belts. Accurate time alignment ensures that the simulation correctly predicts the physical position of a moving part at the exact moment of data acquisition.

Step 5: Realistic Environment

To ensure a seamless simulation-to-reality commissioning process, the fidelity of the robotic simulation is paramount; the more accurately the digital twin mirrors the physical cell, the more transferable the programmed logic becomes. This necessitates that all components within the workcell are meticulously modelled. All objects, such as inspection parts, **fixtures**, **pallets**, **conveyors**, **exclusion zones** (virtual walls that match the physical safety guarding), and all End-of-Arm Tooling, including grippers and sensors, must be imported not only with their precise kinematic models (true dimensions from CAD data) but also with their exact dynamic properties, including mass, center of gravity, and moments of inertia.

It is crucially for sensor simulation, that surface properties must be approximated, assigning **material reflectance** and specularity values to objects allows ray-tracing renderers (in simulators like **NVIDIA Isaac Sim**) to realistically mimic how lidar beams or structured light patterns interact with the parts and environment.

Step 6: Collect and Maintain Expected Outputs

Version-Controlled Robot Description

The primary output of the modelling phase is a consolidated, version-controlled package (typically a git repository). This package contains the "single source of truth": the mesh files (visual and collision), the kinematic description (URDF/XACRO), and the full **TF tree** structure. It must also include a generated **calibration report** that documents the residuals and uncertainty values of the intrinsic and extrinsic calculations, providing traceability for the model's accuracy.

Simulatable Cell

The static descriptions are instantiated into a dynamic, executable environment: a **Simulatable Cell**. This is a configured scene within a physics-based simulator (e.g., **Gazebo**, **NVIDIA Isaac Sim**) that loads the robot, environment, and sensor plugins with the exact transforms defined in the previous steps. This allows for "headless" (no GUI) regression testing and interactive experimentation, serving as the digital sandbox where code is validated before touching physical hardware.

Step 7: Validation

Reachability, Line-of-Sight and Scene Fidelity

Before procurement, the virtual cell undergoes rigorous kinematic validation. **Reachability checks** confirm that the robot can reach all necessary waypoints (pick, inspect, place) with the specific End-of-Arm Tooling attached, without hitting joint limits or singularities. Simultaneously, **Line-of-Sight (LOS) studies** utilize ray-casting in the simulation environment to ensure that the vision sensors have a clear, unobstructed view of the part features at every inspection pose, ensuring that the robot arm itself or the gripper does not occlude the camera or cast shadows on the inspection area.

The final validation acts as a "reality check" for the digital twin. This involves probing known fiducials (fixed reference markers) in both the real world and the simulation to verify that the virtual distances match the physical measurements. This step confirms that the placement of fixtures, conveyors, and the robot base in the simulation corresponds to their actual installation, ensuring that a path planned in the simulator will not result in a collision or a missed target in the real cell.

Challenges

While developing this high-fidelity simulation environment, teams will likely encounter several significant challenges. The primary hurdle is often model fidelity and the "sim-to-real" gap; while acquiring accurate kinematic CAD models is a baseline, determining the precise dynamic properties (mass, center of gravity, and especially moments of inertia) for custom-built grippers or sensor mounts is far more complex and crucial for accurate physics. For quality inspection, the most formidable challenge is the realistic simulation of the sensor itself; accurately modeling the optical properties of a camera, the behavior of a laser scanner on various material surfaces (e.g. reflectivity), and the subtle appearance of defects is exceptionally difficult.

Furthermore, implementing the proposed depth-aware measurement methodology introduces some challenges. The first is the "as-designed" versus "as-built" geometric gap. In simulation, the CAD model of the inspected part is mathematically perfect, but physical parts possess manufacturing tolerances or casting variations. As a result, the precise Z-depths of

the static features mapped in the Digital Twin may shift slightly on the real factory floor, requiring the localized pixel-to-millimeter lookup tables to be fine-tuned during physical commissioning to prevent measurement errors. Finally, this calibration relies heavily on the presence of measurable reference features. A major challenge arises when inspecting smooth, featureless, or continuously curving 3D surfaces (such as polished automotive panels). On these surfaces, distinct cavities or edges may simply not exist at the target depth level to serve as a reliable baseline for the pixel-to-millimeter calibration.

1.3.5 Physical Deployment

Step 1: Hardware Integration

End-of-Arm Tooling Integration

The integration of a robotic platform for automated quality inspection begins with a rigid physical foundation. The **End-of-Arm Tooling (EOAT)** requires exceptional mounting rigidity; the bracket holding the inspection sensor, whether a machine vision camera, 3D laser scanner, or contact probe, must be completely stable, as even small vibrations amplified by the robot's motion can render high-resolution measurements useless.

Cable Management

Equally critical is cable management, which should utilize a full, robot-specific "dress pack." Since inspection sensor cables often carry high-speed data sensitive to electromagnetic interference (EMI) and physical damage (such as GigE, USB3, or CoaXPress), the dress pack must be designed to allow for full 6-axis articulation without snagging, kinking, or exceeding the cable's minimum bend radius.

Communication Protocol Definition

On the control side, the cell architecture typically designates a **PLC (Programmable Logic Controller)** as the "Cell Master." The PLC coordinates the robot, part fixtures and reject mechanisms, while the robot controller functions as a "slave" device that executes its own motion program only upon receiving commands like "Part Ready" or "Start Inspection." This relationship is established through a primary **fieldbus connection** (e.g., EtherNet/IP, PROFINET, or EtherCAT) that handles the essential program calls and status handshaking between the robot and the PLC.

Step 2: Tool Center Point (TCP) and Frame Calibration

The robot's path is entirely dependent on its understanding of its tool and its environment. Once the hardware is integrated, the robot's ability to navigate its environment relies entirely on precise calibration. The most critical step for inspection accuracy is the definition of the **Tool Center Point (TCP)**. The TCP must be defined exactly as the functional point of the sensor, such as the focal point of a camera, the tip of a probe, or the center of a laser scanning plane. For vision systems, an incorrectly defined TCP, for example, a Z-axis error of just 1 mm, will cause the robot to position the camera out of focus, compromising the entire inspection. Beyond the tool, the robot must also understand the location of the work piece. This requires defining a **User Frame** that aligns the robot's coordinate system with the specific part or fixture. By programming all inspection paths relative to this User Frame rather than the robot base, the system ensures consistent alignment with the part's geometry.

Step 3: Path Planning and Motion Control (Module: Robotic control simulation module)

For quality inspection, the *quality* of the motion is as important as the destination.

Offline Programming

For any complex part, inspection paths should be generated **offline** using simulation software (e.g., Siemens Process Simulate, RoboDK, NVIDIA Isaac Sim or the robot's proprietary software), and planning libraries (e.g. NVIDIA cuRobo, MoveIt **Motion Planning**) by importing the part's **CAD model**. Manually teaching hundreds of inspection points with a pendant is impractical and inaccurate.

Motion Type Selection

Travel Moves (PTP): Use **Point-to-Point (PTP)** or **Joint** moves for high-speed travel *between* inspection zones (e.g., moving from "Home" to "Scan_Start_Position"). These are the fastest moves and do not follow a straight line.

Inspection Moves (Linear): Use **Linear (LIN)** moves for the actual inspection path (e.g., moving a camera over a surface or a probe along a seam). This ensures the sensor moves at a **constant velocity** and in a predictable straight line.

Motion Smoothing (Corner Rounding)

Avoid "stop-and-go" motion. A robot that stops at every inspection point, triggers and then moves again is slow and introduces vibration. Instead, use "fly-by" or "smoothing" parameters to have the robot flow through waypoints without a full stop. Crucially, this must be paired with **position-based triggering**, so the camera/sensor is fired at the *exact* moment it passes over the target, even if it doesn't stop.

Singularity Avoidance

When planning paths, especially complex 6-axis moves, use the provided software to check for and route around **singularities** (e.g., wrist-lock). A robot hitting a singularity will pause and re-orient, which will ruin any in-process scan.

Step 4: System Logic

The final element is the system logic that governs how the robot, sensor, and PLC communicate during high-speed operation.

High-Speed Positional Triggering

Crucially, one must not rely on the PLC for **high-speed positional triggering** due to latency and scan-time variability. Instead, the robot's internal, real-time I/O must be configured (using functions like "Position Compare Output") to fire a digital output precisely when the TCP reaches a specific coordinate. This output is wired directly to the sensor's trigger input, ensuring the image is captured at the exact intended location with microsecond-level repeatability, even while the robot is in continuous motion.

Real-Time Depth-Aware AI Measurement

During operation, the system transforms standard 2D pixel output into accurate metric data. The system logic executes a two-step measurement process synchronized with the robot's motion:

1. **Detection:** The AI model performs inference to detect the defect and generate bounding box coordinates (X, Y in pixels).
2. **Dynamic Depth-Measurement:** The system correlates the defect's bounding box with the pre-calibrated depth-level map for that specific robot pose. It dynamically

retrieves and applies the correct pixel-to-millimeter ratio to calculate the physical defect dimensions.

Inspection Procedure Flow Example:

1. **PLC:** "Robot, part is in position. Start inspection."
2. **Robot:** Moves along its programmed linear path, firing the camera trigger at specific coordinates.
3. **Camera/AI Edge Device:** Acquires images and executes the AI detection and dynamic depth measurement pipeline to calculate physical defect dimensions. It then evaluates these dimensions against a predefined measurement threshold and sends a corresponding "Pass/Fail" signal directly to the PLC.
4. **Robot:** Finishes its path and signals the PLC, "Inspection path complete."
5. **PLC:** Combines "Inspection complete" from the robot and "Pass" from the camera, then commands the line to move the part, or commands a reject-gate to activate if the result was "Fail."

Step 5: Validation

The final gate is the validation of the system's temporal performance against the customer's production rate. This check compares the calculated total cycle time against the required takt time. It sums up the **dynamic motion time** (acceleration/deceleration curves), the **static dwell time** (image capture exposure, gripper actuation), the AI computational latency (the milliseconds required for the neural network to process the image and execute the depth-measurement inference at the edge), and the **communication overhead** (handshakes). The system passes only if the total calculated time is less than the takt time, preferably with a safety margin to account for real-world variability.

Challenges

During implementation, significant challenges often arise from the interplay of mechanics and timing. For instance, small vibrations from the robot's motion, amplified by an insufficiently rigid bracket, can introduce noise or blur into high-resolution sensor data, compromising inspection accuracy. The most critical hurdle is often **synchronization and latency**; a minor mismatch or "jitter" between the robot controller's high-speed positional trigger and the sensor's actual acquisition time will cause the inspection to occur at the wrong physical location, leading to inaccurate measurements. Finally, an imprecise **TCP calibration** can systematically skew all results by placing the sensor out of its optimal focal plane or position.

1.3.6 Available Datasets and Code

To support the wider AI and research community, the SoliDAIR project actively publishes open-source resources, including code, pre-trained models, and curated datasets. Currently, two specialized, bounding box annotated datasets focused on **aluminum automotive** components, **collected with vision equipment mounted on robotic manipulators**, are available for download via **Zenodo**:

- **Surface Defects Dataset:** A comprehensive collection of images capturing surface irregularities, such as scratches and dents.
- **Thread Defects Dataset:** A focused dataset documenting structural flaws within threaded features, including deformations and porosity.

By the end of the project the source code of a simulation environment, built with NVIDIA Isaac Sim, containing a robotic manipulator performing inspection tasks on 3D parts (see chapter 1.3.4) will be open sourced on **GitHub** and potentially linked to other relevant platforms such as Zenodo or AIO.

1.3.7 Expertise needed

Successfully deploying and operating an AI-driven robotic inspection cell requires a specialized blend of engineering and operational skills. The implementation of this framework requires distinct skill sets across the virtual development, physical integration, and operational phases.

Development and Virtual Commissioning Phase

Currently, a system integrator or robotics engineer is responsible for designing the cell and establishing the Digital Twin prior to physical build and an AI engineer for developing and training the AI models. The required skills for this stage include:

- **Simulation and Kinematics:** The ability to build high-fidelity Digital Twins, requiring a strong understanding of 3D CAD modeling, robot kinematics, payloads, and dynamic properties (e.g., inertia tensors).
- **AI Data and Lifecycle Management:** The ability to evaluate the quality of the "Not OK" (NOK) image dataset, ensure strict data independence to avoid leakage, and balance the False Acceptance Rate (FAR) versus the False Rejection Rate (FRR) based on business priorities. While employees experienced with the nature of the defects can handle data collection and quality evaluation, the actual AI model training and deployment must be executed by dedicated AI engineers. Deep algorithm-level AI coding is not mandatory for the personnel; however, a basic knowledge of the ML architecture is required by the employees, along with a fundamental understanding of how visual constraints (e.g., camera Field of View, reflections, lighting) directly impact the AI engineers' model training.
- **Defect Measurement Calibration:** To implement the SoliDAIR depth-aware measurement methodology, engineers must deeply understand coordinate frames. They must be capable of identifying distinct geometric depth levels on a part and accurately measuring static features to build the localized pixel-to-millimeter lookup tables.

Physical Deployment Phase

Transitioning from the virtual environment to the physical factory floor shifts the required expertise toward industrial automation and safety:

- **Automation and PLC Programming:** Automation Engineers are indispensable for managing fieldbus communications (e.g., EtherNet/IP, PROFINET). They must write the deterministic logic that handles hardware handshakes and translates the AI's quantitative Pass/Fail signal into physical sorting actions.
- **Mechatronics and I/O Configuration:** Expertise is required to configure high-speed positional triggers using the robot's real-time I/O, ensuring image acquisition is perfectly synchronized with robot motion to prevent blur and latency.
- **Safety Compliance:** A thorough understanding of ISO safety standards is mandatory to conduct risk assessments and integrate safety PLCs and light curtains.

Operational Phase

During daily operation, the robotic inspection cell is handled by key users and line operators. These individuals do not need to be robotics programmers, but they must be able to:

- **Evaluate the AI Performance:** Experience with defect characteristics, enabling critical evaluation of model predictions, identification of incorrect behavior, and initiation of internal model improvements when necessary.
- **Interpret System Diagnostics:** Understand PLC and robot fault codes and correctly execute recovery sequences following an emergency stop or a failed inspection cycle.
- **Monitor Calibration Drift:** Operators must be trained to recognize physical drift. If the robot's End-of-Arm Tooling is bumped, the physical depth levels will no longer match the digital lookup table. Operators must know when to trigger a TCP recalibration or re-measure the pixel-to-millimeter ratios to maintain measurement accuracy.

2 Acknowledgements and disclaimer

The author(s) would like to thank the partners in the project for their valuable comments on previous drafts and for performing the review.

#	Partner	Partner full name
1	FHG	FRAUNHOFER GESELLSCHAFT ZUR FORDERUNG DER ANGEWANDTEN FORSCHUNG EV
2	BRO-B	BROSE FAHRZEUGTEILE SE & CO. KOMMANDITGESELLSCHAFT
3	CIE	FUNDACION CIE I+D+I
4	BOS	BOSCH SANAYI VE TICARET AS
5	AUT	AUTFORCE AUTOMATIONS-GMBH
6	SISW	SIEMENS INDUSTRY SOFTWARE NV
7	UGS	UG SYSTEMS GMBH & CO. KG
8	THL	TWI ELLAS ASTIKI MI KERDOSKOPIKI ETAIREIA
9	VIF	VIRTUAL VEHICLE RESEARCH GMBH
10	I2M	I2M UNTERNEHMENSENTWICKLUNG GMBH

LEGAL DISCLAIMER

Copyright ©, all rights reserved. No part of this report may be used, reproduced and or/disclosed, in any form or by any means without the prior written permission of SoliDAIR and the SoliDAIR Consortium. Persons wishing to use the contents of this study (in whole or in part) for purposes other than their personal use are invited to submit a written request to the project coordinator.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document shall be liable or responsible, in negligence or otherwise, for any loss, damage or expense whatever sustained by any person as a result of the use, in any manner or form, of any knowledge, information or data contained in this document, or due to any inaccuracy, omission or error therein contained.

During the preparation of this work the authors used generative AI tools to improve the readability and language in parts of the manuscript by enhancing stylistic clarity including language editing and grammatical correctness. After using these tools/services, the authors have reviewed and edited the content as needed and take full responsibility for the content of the published article.



Funded by
the European Union