Deliverable 3.2

First planning algorithms, XAI and user modelling, and human-AI conversation loop for use-cases after lab evaluation report

Version	1.0
Lead partner	TU/e
Dissemination level	Public
Type of deliverable	Demonstrator
Contractual date	31/03/2026
Actual delivery date	31/03/2026

D3.2. First planning algorithms, XAI and user modelling, and human-AI conversation loop for use-cases after lab evaluation report

Deliverable description as per DoA: First planning algorithms, XAI and user modelling, and human-AI conversation loop for use-cases after lab evaluation report

(Report + Prototype)

Authors	**Organisations**
Mehrdad Asadi	VUB AI
Diederik M. Roijers	VUB AI
Ann Nowé	VUB AI
Szymon Bobek	JU
Grzegorz J. Nalepa	JU
Adriana Basbous Moukarzel	EUT
Silvia Izquierdo Badiola	EUT
Fábio Neves-Moreira	INESC TEC
Miguel Lunet	INESC TEC
Hendrik Baier	TUE

Contributors	
Chengpeng Hu	TUE
Paolo Speziali	VUB AI

Revised by	
Grzegorz J. Nalepa	JU
Szymon Bobek	JU

Disclaimer

This document reflects the authors' views only and neither the Agency nor the Commission are responsible for any use that may be made of the information contained

Table of Content

Table of Figures 6

List of Abbreviations 7

1. Executive summary 8
1.1 Objectives of Work Package 3 8
1.2 Deliverable Structure 9

2. Research Progress 10
2.1 Planning and Search Algorithms for Human-AI Collaboration (Task 3.1) 10
2.1.1 Semi-Supervised Representation Learning 10
2.1.2 Planning and Learning in Multi-Objective Environments 11
2.1.3 Preference Guided Route Planning in Multi-objective Optimization 12
2.2 Bidirectional Information Flow Between Users and AI (Task 3.2) 14
2.2.1 Information Flow from AI to User (explainable AI) 14
2.2.1.1 LLM-enhanced Explainability, A three level framework 14
2.2.1.2 Structuring LLM-mediated optimization 16
2.2.1.3 Efficient Multi-Modal Model as Explainer of Visual Anomalies 17
2.2.1.4 Towards Explainable Sequential Decision Making 19
2.2.1.5 Explaining Plans of Routing Agents 21
2.2.1.6 Explaining Actions of RL Agents 22
2.2.1.7 Explaining Policies of RL agents 23
2.2.2 Information Flow from User to AI (user modelling and feedback) 25
2.2.2.1 Translating Unstructured User Preferences and Feedback into Structured Decision Parameters 25
2.2.2.2 Active Representation Learning in Anomaly Detection Task 26
2.3 User - AI Conversation Management (Task 3.3) 28
2.3.1 Explaining Actions of Search Algorithms 28
2.3.2 Multimodal Models as Anomaly Detector with Metadata-Augmented Prompts 29

3. Realization of PEER Companion Agents in Use Cases 31
3.1 Sonae. A Personalized In-Store Navigation Assistant for Customers (SigaApp) 31
3.1.1 Environment Model 32
3.1.2 User Model 33
3.1.3 Problem Abstraction 35
3.1.4 Problem Solver 38
3.1.5 Solution Realization and Monitoring Module 39
3.1.6 Feedback and Learning Module 40
3.1.7 Conversation Management Module 40
3.1.8 Next steps 40
3.2 City of Amsterdam. A Personalized Navigation App for Wheelchair Users 41
3.2.1 Environment Model 41
3.2.2 User Model 41
3.2.3 Problem Abstraction Module 42
3.2.4 Problem Solver 42

3.2.5 Solution Realization and Monitoring Module 46
3.2.6 Feedback and Learning Module 46
3.2.7 Conversation Management Module 49
3.2.8 Next steps 50
3.3 Proditec. Human-in-the-loop and Explainable Anomaly Detection Trainer 52
3.3.1 Environmental Model 52
3.3.2 User Model 53
3.3.3 Problem Abstraction Module 53
3.3.4 Problem Solver 54
3.3.5 Solution Realization and Monitoring Module 54
3.3.6 Feedback and Learning Module 56
3.3.7 Conversation Management Module 56
3.3.8 Next steps 57
3.4 Datacation. AI-assisted warehouse configuration optimization 57
3.4.1 Environment Model 57
3.4.2 User Model 59
3.4.3 Abstraction Module 60
3.4.4 Problem Solver 62
3.4.4.1 Scientific Solver 62
3.4.4.2 Increasing scalability 65
3.4.5 Solution Realization and Monitoring Module 66
3.4.6 Conversation Management Module 66
3.4.7 Next Steps 67
Lessons Learned 70
Deviations occurred and corrective actions taken 72
Conclusions 73
References 74

Table of Figures

Figure 1 WP3 Gantt Chart 9
Figure 2 Workflow of the proposed methods, illustrating the integration of unsupervised representation learning, and proposed double thresholding with human-assisted labeling for anomaly detection. 11
Figure 3 Understandability and dialogue level. Taken from original article [7] 15
Figure 4 Visualization of the human-in-the-loop refinement process for anomaly detection 18
Figure 5 Examples of misleading textual hints in anomaly detection, highlighting cases where descriptions fail to accurately convey the nature of defects. 18
Figure 6 Different research fields approach XSDM from different angles 19
Figure 7 A taxonomy for XSDM shows potential for transferring insights 20
Figure 8 High-potential XSDM techniques for transfer across research fields 20
Figure 9 Example for a counterfactual route explanation 21
Figure 10 LogiEx explains planning by combining domain knowledge, the search process, and logical reasoning 29
Figure 11 Different modalities produced by the PatchCore model—heatmaps, segmentation maps, and patches—using an example of a squeeze anomaly from the capsule dataset 30
Figure 12 Architecture of the PEER Companion Agent. 31
Figure 13 Main analytical challenges being tackled in Sonae use case. 31
Figure 14 Current retail store layout. 32
Figure 15 Simulation environment to test the graph generation approach. 32
Figure 16 The proposed light-weight next basket recommendation pipeline 34
Figure 17 Static product recommendation. 36
Figure 18 Dynamic product recommendation examples. 37
Figure 19 Triggers for the dynamic product suggestion. 37
Figure 20 Routes in Pareto front for Osdorp-Midden route planning problem (far start/origin – two objective instance (distance, crossings) 46
Figure 21 First solution (iteration) – optimized for Length objective 48
Figure 22 Second solution (iteration) – optimized for Length objective 48
Figure 23 Third solution (iteration) – optimized for number of crossings 49
Figure 24 Designed user interface that PG-IPRO assumes available. 50
Figure 25 Example of overlapping distributions of normal and anomalous samples in the MVTec AD Leather category 55
Figure 26 Example of heatmaps generated for user during conversation. 57
Figure 27 Example conversation flow including information retrieval and solver configuration 61
Figure 28 Example solver parameter configuration suggestions provided by AI agent based on conversation 62
Figure 29 Rolling-horizon solver approach. 63
Figure 30 Solution example. 65
Figure 31 Connecting learning algorithms with classical optimization. 66

List of Abbreviations

Abbreviation	Definition
AD	Anomaly Detection
API	Application Programming Interface
DSL	Domain Specific Language
DT	Digital Twin
EMM	Exceptional Model Mining
ESR	Expected Scalarized Return
GP	Gaussian Process
HITL	Human-in-the-loop
IPRO	Preference-Guided Iterative Pareto Referent Optimization
JSSP	Job Shop Scheduling Problem
LLM	Large Language Model
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MF	Matrix Factorization
MIP	Mixed Integer Programming
MODFS	Multi Objective Depth Frist Search
MORL	Multi-objective Reinforcement Learning
MOSSP	Multi Objective Shortest Path Problem
MVP	Minimum Viable Product
NBR	Next Basket Recommendation
NIR	Next Item Recommendation
RL	Reinforcement Learning
SDM	Sequential Decision Making
XSDM	Explainable Sequential Decision Making
TD	Temporal Learning
UI	User Interface
UX	User Experience
WMS	Warehouse Management System
XAI	Explainable AI

1. Executive summary

Work Package 3 (WP3) has further advanced towards its objectives, and this D3.2 report presents a comprehensive view of our ongoing research as well as its application across our four use cases. The report points out our progress on AI explainability, user modelling, and AI-user interaction in general, as well as on research topics specifically motivated by our use cases, such as anomaly detection, multi-objective routing, or LLM-mediated optimization.

In addition, we report on the further development of the proposed PEER agent architecture in practice. Following on the Minimum Viable Product (MVP) of the previous release as part of D3.1[1], which was delivered in the form of Figma designs for Wizard of Oz style AI simulations, this second release (MVP2) constitutes the first code implementation of the PEER agents. It will allow users to directly interact with the systems in the next round of the use case evaluations. We describe the algorithms employed in these agents, including the first Problem Solver modules based on novel (multi-objective pathfinding) as well as tried-and-tested approaches (mixed integer programming), and the first XAI and user feedback prototypes for the different domains. Furthermore, we highlight the planned next steps, which will bring the systems closer to real-world deployment and enable richer human-in-the-loop interaction, for example by fleshing out the Solution Realization and Monitoring modules, and by using abstraction to scale up the employed solvers.

1.1 Objectives of Work Package 3

WP3 is named "*Build: Human-AI collaboration on sequential decision-making tasks*". In this work package, we aim to develop the next generation AI planning systems that will be used at the core of the PEER collaborative AI assistant. They will support improved feedback loops and two-way communication flow between AI and human user, allowing for better collaborative human-AI teamwork in complex tasks that extend over time.
The timeline of WP3 tasks is outlined in the Gantt chart below (Figure 1). Task 3.1 lays the foundation by developing flexible AI planning methods. Task 3.2 enhances AI-user interaction by refining explanation and feedback mechanisms. Finally, Task 3.3 integrates these advancements into a comprehensive conversational framework, ensuring a cohesive and adaptive AI experience. This document, as part of Deliverable 3.2, provides a report on the progress of these three tasks so far, and details our work towards the "first planning

algorithms, XAI and user modelling, and human-AI conversation loop for use-cases after lab evaluation report".

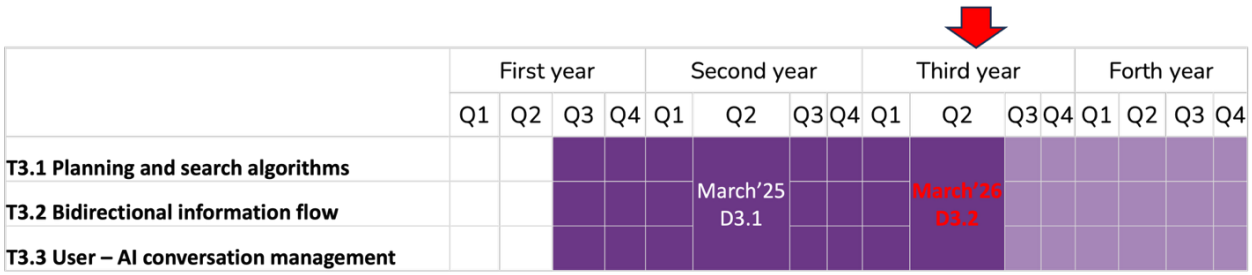


Figure 1 WP3 Gantt Chart

1.2 Deliverable Structure

This report is structured to provide a complete overview of work performed in WP3, its research, application in use cases, and integration with other work packages. Section 2 gives an overview of our research contributions in some of the directions identified as promising in the previous deliverable D3.1, as well as some work on newly identified topics - both fundamental topics as well as topics directly motivated by the results of the previous evaluation round. This is organized according to the three tasks 3.1, 3.2, and 3.3. Section 3 presents our progress on the envisioned PEER systems in all use cases, organized according to the modules of the generic architecture we proposed in D3.1 [1]. Finally, Section 4 concludes with lessons learned.

2. Research Progress

2.1 Planning and Search Algorithms for Human-AI Collaboration (Task 3.1)

According to DoA: Within this task, we will extend the planning and search algorithms at the heart of the AI systems used in the different use cases. The main idea is to allow for the flexibility that improved human-AI communication and collaboration will require, such as proposing partial instead of complete solutions, proposing a variety of solutions instead of single solutions, updating the solutions instead of computing them from scratch, proposing simple to explain solutions instead of complex ones, and adapting the problem-solving algorithms to process a richer set of input data: not only from the problem definition itself, but also from the human-AI conversation around it. We also work on automated construction/updating of planning models (action models and control structures).

2.1.1 Semi-Supervised Representation Learning

In practical anomaly detection scenarios, datasets are typically large, anomalies are present but extremely rare, and labeled data is scarce. Full classification approaches are infeasible because they require labeling a substantial portion of the dataset, even when using imbalance-aware learning techniques. Conversely, fully unsupervised anomaly detection methods are insufficient in safety-critical applications, as model validation and performance assessment still require labeled instances. Assuming an average anomaly prevalence of 1%, one would need to label approximately 1,000 instances to expect at least 10 anomalous samples for reliable evaluation. In industrial workflows such as quality control, anomaly detection is closely tied to high-consequences decisions, such as rejecting production batches, triggering root-cause investigations, or modifying manufacturing parameters in real-time. Under these conditions, two factors become equally critical: the ability to process large volumes of unlabeled data through unsupervised detection and the need for reliable quality assurance supported by labeled instances. This necessity drives the development of approaches that minimize labeling effort while maintaining detection accuracy. Our research responds to this need for balance by proposing a method that is both accurate and label efficient.

We mainly address the practical challenge of quality assurance in anomaly detection, where reliable labels are essential to ensure algorithmic performance. To mitigate the abovementioned limitation, we propose a double-thresholding strategy for anomalydetection based on PatchCore[2], evaluated on the MVTec AD benchmark[3]. The approach enhances the separation between normal and anomalous sample, requiring minimum annotated data. Experiments demonstrate clear improvements in anomaly detection compared to the standard thresholding mechanism. As shown in the Figure 2 the pipeline is summarized as follows:

- Initial thresholding based on inter-patch distances in the latent space, prioritizing regions with the highest deviation to isolate potentially abnormal areas.
- Score smoothing combined with global distribution analysis, ensuring that anomaly assessment considers the entire image context rather than isolated patches.
- Final thresholding guided by the refined score distribution, leveraging statistical insights from the smoothed anomaly scores to establish a robust decision boundary.
- Run-time and Human-in-the-loop (HITL) validation, where uncertain cases near the decision boundary are reviewed by experts to refine the threshold and improve data quality (See Section 2.2.2.2).

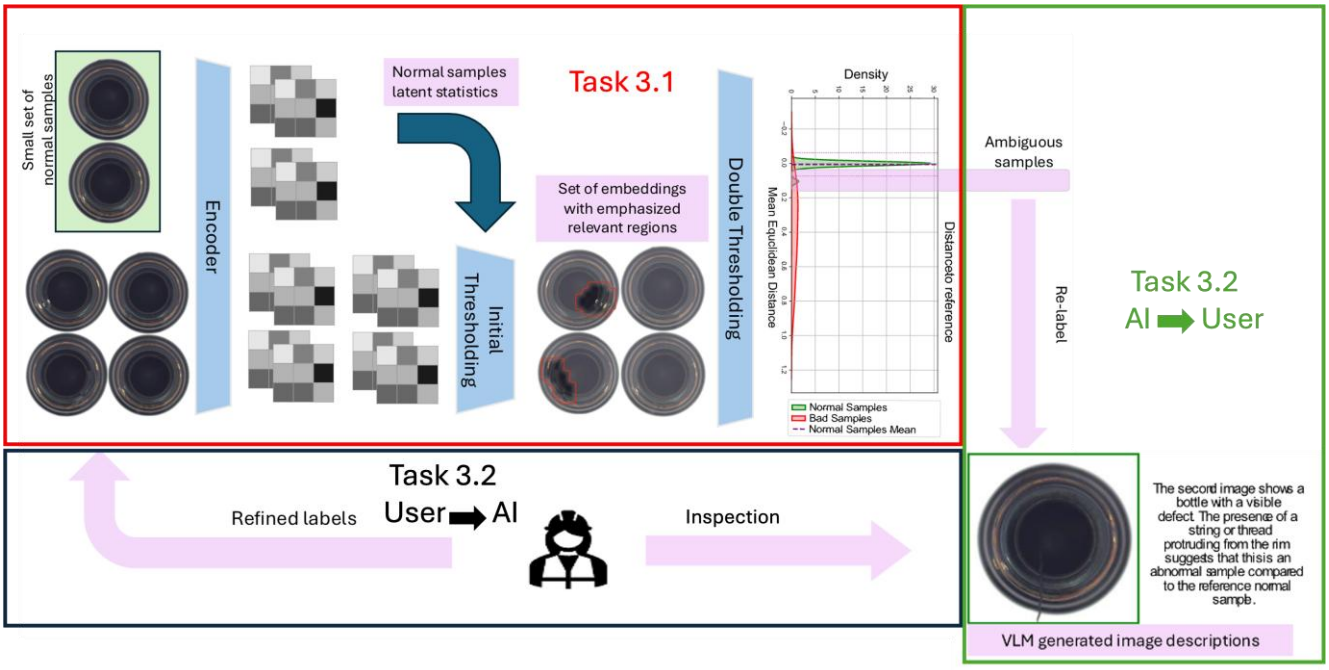


Figure 2 Workflow of the proposed methods, illustrating the integration of unsupervised representation learning, and proposed double thresholding with human-assisted labeling for anomaly detection.

The evaluation of anomaly scoring methods across the MVTec AD dataset categories reveals that the Silhouette-based method consistently outperforms others, achieving the highest average F1 scores in 10 out of 15 categories. Notably, it leads in categories such as *Bottle*, *Cable*, *Capsule*, *Carpet*, *Screw*, *Tile*, and *Zipper*, indicating good generalization across both texture and object types. The Cluster-based method shows competitive performance, ranking highest in Grid, Hazelnut, Metal Nut, Toothbrush, and Wood. Meanwhile, the Wasserstein method achieves the best score only in the Pill category, and the Mean Distance method leads in Leather and Transistor. The pure PatchCore baseline, while consistent, yields significantly lower F1 scores across all categories, highlighting the advantage of incorporating our double-thresholding approach.

2.1.2 Planning and Learning in Multi-Objective Environments

In **Multi-Objective Reinforcement Learning (MORL)**, an agent must optimize for several goals simultaneously—such as a delivery robot balancing speed, battery life, and safety.

Examples from PEER are the multi-objective routing in the Amsterdam use case, and the multi-objective warehouse planning in the Datacation use case. These objectives are typically combined into a single value using a **utility function**. While linear functions (simple weighted sums) are common, many real-world preferences are *nonlinear*. For instance, in a toy "FishWood" scenario where you need both fish and wood for dinner, the utility is determined by whichever resource is in shortest supply—a classic "min" function where having 100 fish but 0 wood results in zero utility.

The motivation for our learning approach **MOUAC** [4] is the failure of standard RL techniques when faced with such nonlinear preferences under the **Expected Scalarized Return (ESR)** criterion. In ESR, the agent seeks to maximize the utility of the outcome of each individual episode, rather than the utility of the average outcome of many episodes. The crucial challenge is that the classic Bellman equation—the mathematical foundation of RL that allows agents to learn step-by-step—no longer holds in its standard form for ESR. In addition, there is a "delayed feedback" problem: because utility is only calculated at the very end of a trajectory based on the final total of all rewards, the agent receives no useful signal during its journey.

MOUAC achieves a solution by projecting the problem into an **extended Markov Decision Process (MDP)**. By including the currently "accrued" rewards as part of the agent's state, the system restores the validity of the Bellman equation. This allows the use of **Temporal Difference (TD) learning**, where the agent updates its beliefs based on one-step changes rather than waiting for the end of a long episode.

The outcome is an **Actor-Critic** approach that uses a **Utility Critic** to estimate future utility more efficiently. Through experiments in test environments like "Deep Sea Treasure" and the higher-dimensional "Fruit Tree", MOUAC demonstrated faster convergence and higher stability than previous state-of-the-art methods. It effectively balances the trade-off between bias and variance, providing a more robust learning tool for AI assistants that must manage several, potentially conflicting human preferences.

2.1.3 Preference Guided Route Planning in Multi-objective Optimization

As explained in our first deliverable (D3.1), multi-objective decision-making problems are prevalent in the daily life of a human. AI can help with this problem by aiding in proposing potential solutions with optimal trade-offs. However, finding these solutions can become a time problem as some methods require all the Pareto-optimal solutions to be known beforehand to perform proper preference optimization and to satisfy the needs of the user. Interactive methods are required to solve this time constraint problem and that is why we propose a new method, called Preference-Guided IPRO [5], to perform multi-objective preference optimization by adding human-in-the-loop feedback to guide search in the objective space of Pareto-optimal solution vectors. The goal for a user is to find a solution with optimal trade-offs and simultaneously increasing satisfaction levels as high as possible in the lowest amount of time.

The progression of the PG-IPRO method runs as follows:

- **Initialization phase of (GP-)IPRO**: Instantiate PG-IPRO that performs the original IPRO initialization phase (more details in[6]).
- **Propose an initial Pareto-optimal solution**: After initializing PG-IPRO, suggest one of the extrema Pareto-optimal solutions or suggest the first solution returned by the Oracle that is called with the Nadir as referent.
- Start the **preference elicitation loop** (to iteratively ask queries and update the objective search space):
    - **Retrieve solution** for the following query:
        - **Query single-objective preference**: The user gives feedback on what objective it wants to see optimized with regard to the currently proposed solution. This current solution is always the last one generated by the Oracle (in IPROs internal loop) or initially the first Pareto-optimal solution proposed after initialization.
        - **Call internal IPRO loop**: Generate a new Pareto-optimal solution by calling IPROs internal logic with the objective to be optimized. This guides the search on current user preferences by selecting a referent from the improving. This generated solution is the newly proposed solution to the user.
    - **Make query**: Ask the user to pairwise compare the newly proposed solution (with the values of the objectives) against the solution[1] most preferred from previous queries until that timestep. Higher noise results in more uncertainty by the user during pairwise comparing two solutions for which one is more in line with its preferences.
    - **Most preferred data point update**: The new pairwise-preferred solution is maintained as most preferred.

The goal is not to approximate the entire Pareto front, but to guide IPRO in selecting specific referents. PG-IPRO uses the information collected from the user by asking preference-specific questions to attain optimal performance. Queried preferences guide PG-IPRO by selecting the next most promising referent. Consequently, increasing utility iteratively by querying the Oracle, which returns a new route solution to be guaranteed with improved user satisfaction. Our experiments show promising results for this newly introduced interactive method compared to a Bayesian optimization method as baseline. Investigations are also conducted to verify the benefits of the anytime characteristic in (PG-)IPRO, resulting in collecting Pareto-optimal solutions faster on demand, without first having to wait for the approximated Pareto front to be generated. This shows a higher performance in the time to find an average intermediate route compared to the time to collect the approximated Pareto front of Pareto-optimal routes.

[1] This current most preferred solution might not necessarily be the one with highest underlying utility due to high noise levels when the user compares solutions. This leads to incorrect previous comparisons made by the user with regard to the real utility function that represents their preferences.

2.2 Bidirectional Information Flow Between Users and AI (Task 3.2)

According to DoA: In this task, we will extend the information flow between users and AI. This involves a) the generation of more adaptable, personalised, and therefore more relevant explanations in order to inform the user, as a catalyst for transparency and trustworthiness; and b) the smarter processing of user feedback in order to inform the AI, both about the task at hand, as well as about the current user's needs and preferences. An important aspect of this task is incorporating two types of user feedback: direct feedback offered by the user, and indirect feedback from observed user behaviour.

2.2.1 Information Flow from AI to User (explainable AI)

2.2.1.1 LLM-enhanced Explainability, A three level framework

As AI systems permeate high-stakes and socially sensitive domains, there is an urgent need to move beyond one-size-fits-all explanations. Generic or overly technical explanations risk alienating non-expert users, undermining trust, and failing to meet regulatory or societal expectations. What remains lacking is ***a structured methodology for tailoring explanations to distinct audiences*,** ensuring that they are not only technically faithful, but also contextually meaningful, ethically grounded, and socially relevant. In our recent work, we address this gap by proposing a multilevel audience-aware framework for explainability that aligns explanation strategies with the varying goals, knowledge levels, and trust requirements of stakeholders. Figure 3 shows how these levels are integrated and interact. This multilevel approach reflects how AI systems are adopted in real-world contexts, where explanations evolve alongside users' familiarity, system sophistication, and broader social demands.

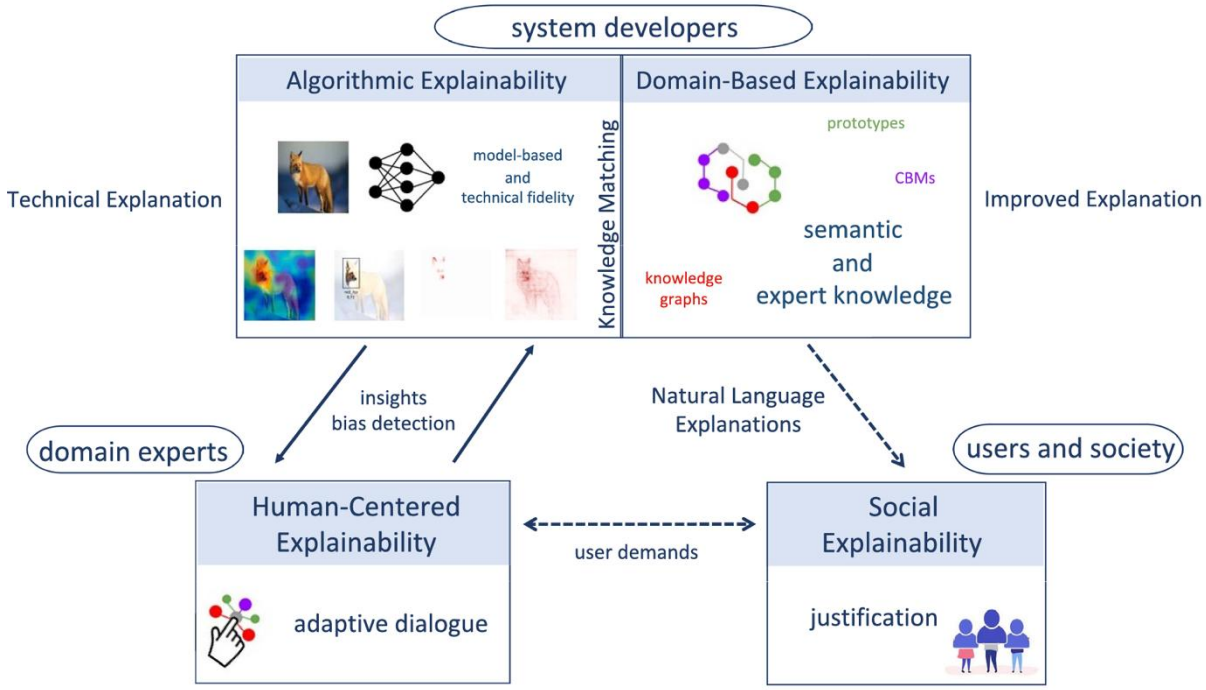


Figure 3 Understandability and dialogue level. Taken from original article [7]

We introduce a multilevel framework for XAI that addresses the needs of domain experts and broader societal audiences. The first level focuses on model-based explanations, combining inherently interpretable models and post hoc analyses to reveal how input drives outputs and how domain knowledge is incorporated. The second level supports human-centered explanations, allowing stakeholders to interact with, query, and adapt the explanation content through dialogue. The third level delivers socially grounded narratives that foster public trust, regulatory compliance, and collective accountability. By structuring explanations in this way, our framework ensures that each successive level builds on the fidelity and technical rigor of the previous level while extending accessibility to non-technical users.

Case Study: Applying the Multi-level Framework with Large Language Models

To illustrate the application of the multilevel explainability framework in practice, we now present a set of concrete case studies. Each example is drawn from a common scenario— loan approval—to show how explanations can be constructed across different levels and enriched using Large Language Models (LLMs). These cases show how LLMs can support the generation of explanations by translating technical output into user-friendly narratives, facilitating interactive dialogue, and contextualizing decisions for greater societal understanding, we refer the reader to our recent publication for more details on this.

While the proposed multilevel framework offers a structured approach to audience-centered explainability, it remains a conceptual foundation that requires further empirical validation. The illustrative case studies provide plausibility but are not exhaustive across domains or user types. Additionally, the use of LLMs introduces challenges related to factual consistency, bias amplification, and explainability of the LLMs themselves.

Looking ahead, several research directions remain open. First, there is a critical need for standardized qualitative and quantitative metrics to evaluate the effectiveness, usability, and ethical adequacy of explanations at all three levels. Second, adaptive systems could

dynamically tailor explanation types to individual user profiles, learning contexts, or decision criticality. Third, operationalizing social explanations on a scale will require integrating participatory audit systems, regulatory dashboards, and community feedback loops. Finally, since LLMs play an increasingly central role in these explanation pipelines, ensuring their fairness, robustness, and alignment with societal values is essential. Addressing these challenges will help transform the proposed three-level framework from a conceptual roadmap into a practical, trustworthy solution for real-world AI systems. Future research should explore how multilevel explanations can be integrated into operational AI pipelines in PEER use case domain such as navigation service, logistics, and other public services, where transparency and accountability are paramount. This includes designing user studies across different stakeholder groups to assess how explanations influence understanding, trust, and behavior. In addition, interdisciplinary collaboration will be essential, with legal scholars, social scientists, and domain experts, to ensure that the explanations are not only comprehensible, but also normatively sound.

2.2.1.2 Structuring LLM-mediated optimization

In the warehouse optimization use case (DAT), human-AI co-deliberation is operationalized through a mediation layer between human operators and a Mixed-Integer Programming Model (MIP). Rather than using LLMs to generate end-to-end optimization models directly as done in the LLMs-for-operations research paradigm[8], [9], the formal optimization formulation is preserved and a locally deployed LLM is positioned as an intermediary agent responsible for translating qualitative operator input into quantitatively bounded solver parameters. This architectural choice ensures that the mathematical guarantees of the solver remain intact, while the LLM supports interpretation, contextualization, and configuration.

Solver parameters (e.g., penalty weights, constraint limits, and lookahead planning horizon) are inferred from user input and returned together with structured rationales. These rationales are generated based on the operator's instructions, the recorded conversation history, validated warehouse state information retrieved from the Digital Twin (DT), and predefined feasibility constraints. Importantly, they are not presented as mechanistic explanations of the LLM's internal reasoning process. As discussed in the literature [10], reasoning traces produced by LLMs cannot be assumed to faithfully reflect internal computational mechanisms. Accordingly, no claims are made regarding introspective interpretability. Instead, the generated traces are treated as structured interpretive artifacts intended to support deliberation. By explicitly highlighting which operational factors (e.g., congestion, labor constraints, retrieval prioritization) are likely reflected in the suggested parameter values, the system enables operators to evaluate, question, refine, or extend the proposed configuration. In this manner, rationales function as a catalyst for informed human oversight and iterative co-configuration, rather than as evidence of internal model transparency.

To ensure structure and auditability, all system responses are constrained by predefined output schemas, embedding validation within a formally defined interface rather than

allowing unconstrained narrative generation. Furthermore, grounding of outputs is achieved through controlled tool invocation mechanisms. Queries regarding warehouse capacity, location compatibility, or configuration changes are resolved exclusively through API calls to the DT.

2.2.1.3 Efficient Multi-Modal Model as Explainer of Visual Anomalies

Visual anomaly detection has advanced significantly with the integration of explainability techniques such as attention mechanisms, saliency maps, and post hoc reasoning frameworks [11]. These methods improve transparency by highlighting image regions or modalities responsible for anomaly detection, which is critical in industrial and medical applications.

Within this task, we evaluated how Phi3.5 multi-modal model may efficiently reason and explain the visual anomalies. As shown in Figure 2, this explainer could play a vital role in enabling human-in-the-loop for data quality assurance and validation. We take advantage of such multi-modal model to incorporate a human-in-the-loop mechanism to validate and resolve ambiguities near the decision boundary, leveraging expert input for critical cases where automated methods struggle. This enables flexible use of our approach as either a standalone detection system or a data labeling tool compatible with any anomaly detection algorithm, enabling practical deployment of anomaly detection systems under limited labeling budgets. Figure 4 illustrates how such explainer is used for data quality assurance and human-guided representation learning.

Our study focuses on the MVTec Anomaly Detection dataset[12] comprising 15 distinct categories containing 3629 images for training and validation, and 1725 images for testing. The dataset is organised into two primary groups: objects and textures. A comprehensive overview of the dataset is presented in Table 4. For each category in the dataset, we trained a PatchCore anomaly detection model using the implementation provided by the anomalib library[13]. We employed *ResNet-18* as the feature extraction backbone and set the *batch size* to 16 during training. The PatchCore model produces informative outputs, including anomaly heatmaps and segmentation masks, which were utilized in our subsequent experiments. Its strong visual reasoning and language generation capabilities make it well-suited for interpreting image anomalies and providing human-readable explanations.

Our findings indicate that multimodal foundation models hold significant potential for explainable visual anomaly detection our next step is to evaluate and implement the module in real use case scenario.

We also take advantage of such explanation to convey AI reasoning over samples that lies in regions of uncertainty of our double-thresholding mechanism (See Section 3.3.5 - Figure 25). Figure 4 illustrates the most ambiguous instances selected for manual review. The descriptions generated with the Phi3 vision instruct model above the images enables human-

AI communication in the anomaly detection process. It can be seen that the last image although classified as normal, contains a dent that was not detected by the algorithm, but was correctly identified by the Phi3 model and highlighted for human review.

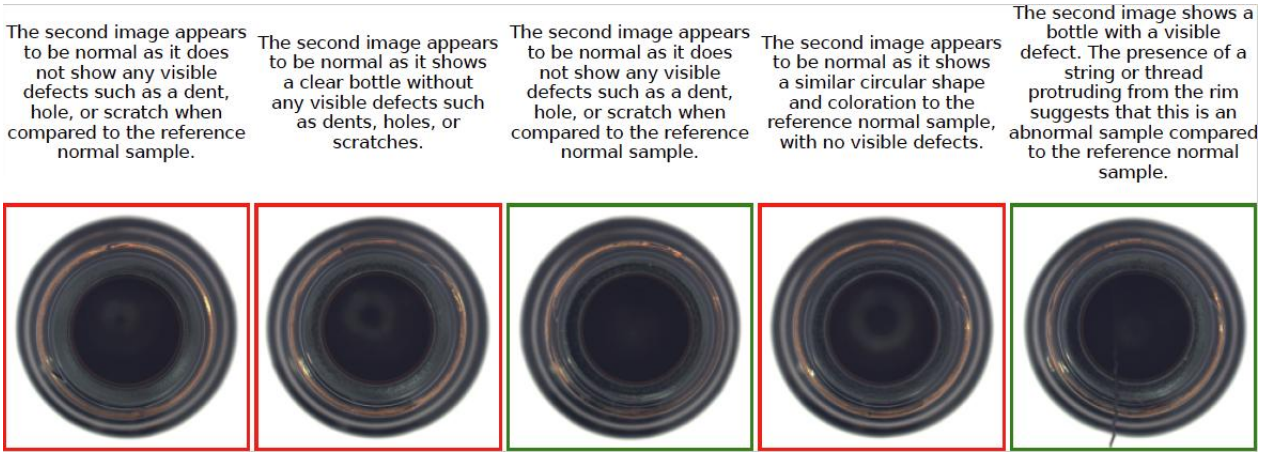


Figure 4 Visualization of the human-in-the-loop refinement process for anomaly detection, highlighting the most ambiguous instances selected for manual review. Above the images there are descriptions generated with Phi3 model that adds a layer of explainability to the anomaly detection process. The red rectangles indicate that the image was predicted as anomalous, while the green rectangles indicate normal predictions.

In certain cases, they may even introduce confusion, making the absence of a description preferable. The examples of such misleading textual hints were presented in Figure 5 below.

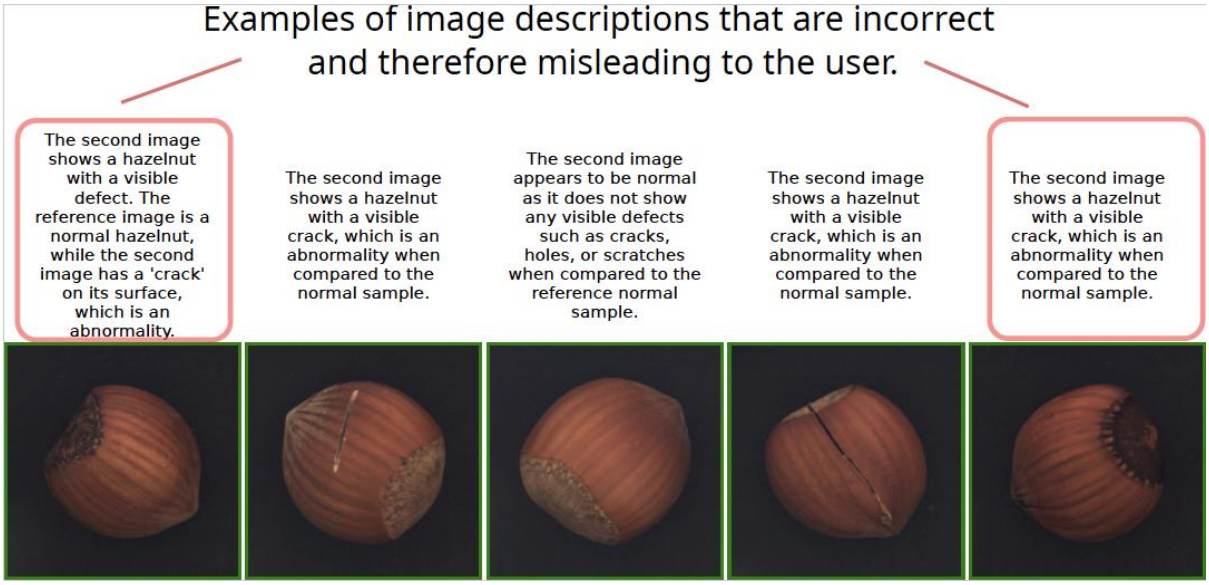


Figure 5 Examples of misleading textual hints in anomaly detection, highlighting cases where descriptions fail to accurately convey the nature of defects.

Further study is needed to understand this effect. To address this issue, we have initiated work on visual prompting techniques (See Section 2.3.2), such as heatmaps and highlighted regions, to enhance explanation capabilities. However, the effectiveness of the method is strongly dependent on the quality of the backbone model.

2.2.1.4 Towards Explainable Sequential Decision Making

As discussed in the previous deliverable report D3.1, the field of Explainable AI (XAI) has seen significant advancement, but its traditional focus on single-shot tasks — like identifying an object in an image — leaves a large gap when applied to Sequential Decision-Making (SDM). In SDM, an agent performs a series of interdependent actions over time to reach a goal, meaning a single decision cannot be understood without considering its history and its impact on the future. To date, explainability approaches for SDM problems have become fragmented across sub-fields like explainable planning, reinforcement learning (RL), and robotics.

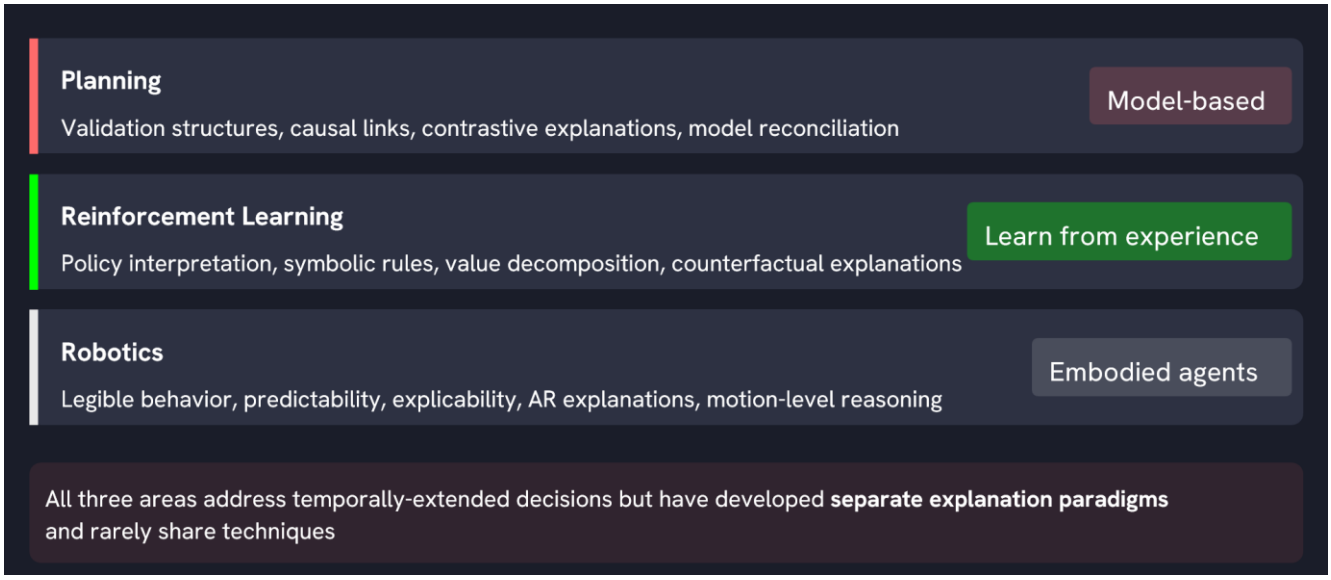


Figure 6 Different research fields approach XSDM from different angles

In ongoing work building on the Dagstuhl seminar organized in 2024, we continue to advocate for a holistic "**XSDM**" framework to unify these efforts. By moving beyond narrow domains, researchers can transfer insights — such as applying “model reconciliation" (aligning a user’s mental model with the AI's actual model) from the planning community to the RL community. A core contribution of our recently published vision paper[14] is the identification of three primary "objects" that require explanation in any sequential task: the **decision sequence**, the underlying **algorithm**, and the **environment**; as well as comparing how different research areas have emphasized different explanation paradigms in these shared categories.

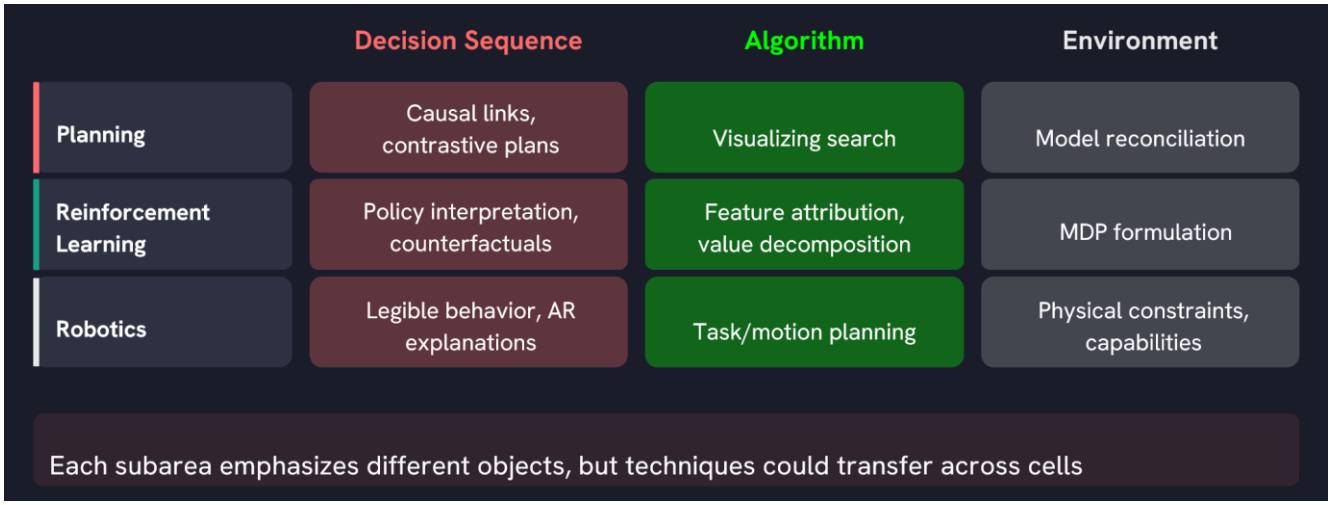


Figure 7 A taxonomy for XSDM shows potential for transferring insights

Explaining the **decision sequence** involves clarifying why specific actions were chosen, how they lead to long-term goals, and what trajectories were explored. This is more complex than standard feature attribution because it must account for temporal intent and potential "key events" that change the course of a plan. Explaining the **algorithm** focuses on the "how"—illuminating the search or learning process, such as how an RL agent assigns credit for rewards. Finally, explaining the **environment** addresses the agent's world model, revealing how it perceives physical constraints or legal rules, which is especially critical in robotics.

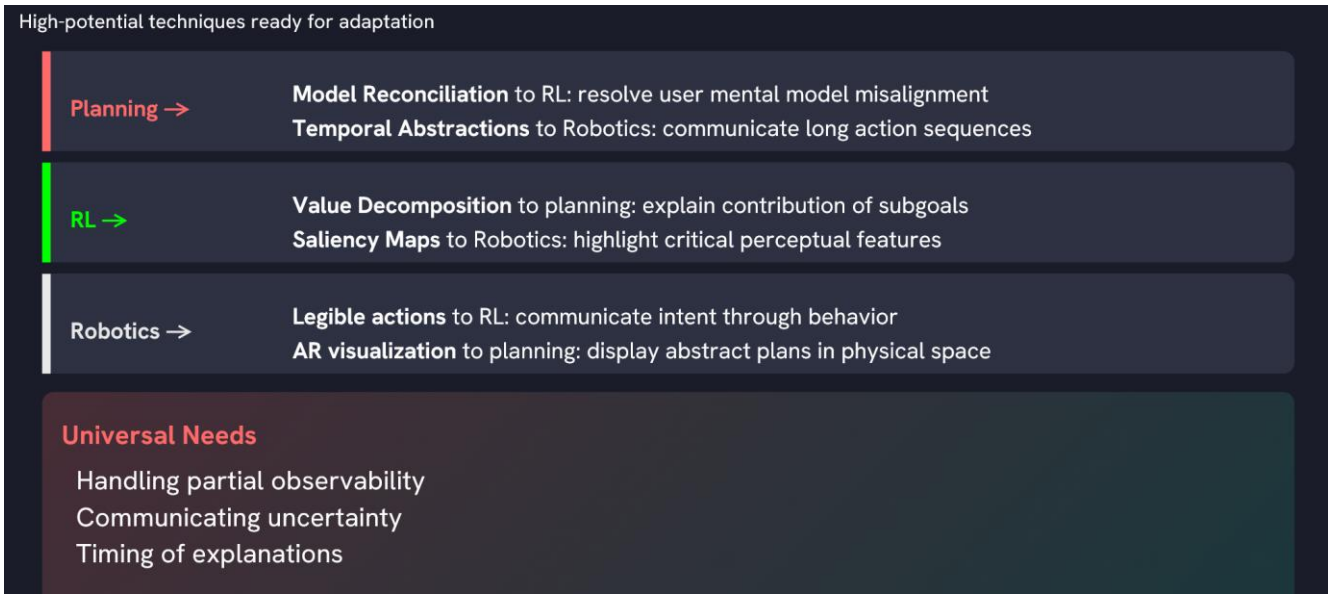


Figure 8 High-potential XSDM techniques for transfer across research fields

The work also highlights the unique psychological aspect of SDM: users are more likely to assign "agency" and "intent" to systems that act over time (in fact, even to simple animations that unfold over time [31]. This requires explanations that move beyond mathematical outputs toward communication of beliefs and capabilities. Ultimately, the XSDM framework could serve as a roadmap for the PEER project, ensuring that as we build more complex assistants, we maintain a unified language for transparency across all use cases.

2.2.1.5 Explaining Plans of Routing Agents

One of the most popular ways to explain an AI's decision is through a **counterfactual explanation**: telling the user what would have to change in the world for the AI to have made a *different* choice, often suggested by the user. This was the topic of a scientific competition we organized at IJCAI 2025, the first **Counterfactual Routing Competition (CRC 2025)**[2], centered on the PEER use case of **accessible routing in Amsterdam**.

As reported in previous deliverables, this use case focuses on the needs of wheelchair users, whose navigation requirements are highly individual. A standard route planner might suggest a path that is the shortest; but the user might prefer a different path or not understand why the planner's recommended “fact” path is indeed better under the current circumstances than their expected “foil” path. If the AI doesn't pick the user's expected route, the user may want to know *why*. A counterfactual explanation provides this: *"I didn't pick your preferred shortcut because it's too narrow for your wheelchair, but if that sidewalk were 10cm wider, it would be the optimal route"*.



Figure 9 Example for a counterfactual route explanation

To facilitate this, we used data provided by the city of Amsterdam, representing parts of the city as a large graph where nodes represent locations and edges represent the navigable streets or sidewalks between them. Each edge was enriched with **attributes** critical to accessibility, such as path width, curb height, and whether it is a designated bike or foot path. This structure allows the routing problem to be framed as a Sequential Decision-Making (SDM) task on a map, where an agent—our AI assistant—must find the best sequence of street segments to connect a start point to a destination based on a specific user’s model of preferences.

2 https://sites.google.com/view/crc25-ijcai/home

The primary challenge for competition participants was to develop algorithms that could identify the **minimal "counterfactual" changes** to this city map that would make a user's preferred "foil" route appear optimal to the route planner. Participants were provided with a variety of graph operators to simulate these changes, ranging from **structural operators** (like unblocking a street currently under maintenance) to **variable operators** (like lowering a curb or widening a narrow sidewalk). This turned the explanation process into an optimization problem: finding the "closest" version of reality where the user's suggestion becomes the best possible choice.

The competition successfully brought together several international teams, evaluating their solutions on a hidden dataset of real Amsterdam neighborhoods. The results demonstrated that complex search strategies, including evolutionary algorithms and reinforcement learning, could effectively generate these personalized explanations in real-time. These outcomes provide the PEER project with potential methodologies for making urban mobility more inclusive, ensuring that our AI assistant can not only suggest safe paths but also explain the physical constraints of the city in a way that respects the individual autonomy and safety of its users.

2.2.1.6 Explaining Actions of RL Agents

Some of our ongoing explorative work is currently investigating a paradigm shift in how agents learn: moving away from opaque "black-box" models toward **structured policy learning** coupled with language, rules, or programs. This research direction seeks to treat human-understandable abstractions not merely as an output for the user, but as a core functional component of the agent's internal reasoning process.

A primary commonality across these approaches is the concept of a **"Bottleneck."**[15], [16] In traditional reinforcement learning, the internal logic of an agent is often a high-dimensional neural network that is impossible for humans to audit. Bottleneck models—whether they use **concepts, rules, or natural language**—force the agent to pass its sensory information through an intermediate layer of human-interpretable granules before choosing an action. For example, instead of mapping raw pixels directly to a "move left" command, a "Language Bottleneck" agent might first generate a text description of the scene (e.g., "There is an obstacle ahead") and then use that description to decide its next move.

Interestingly, some research suggests that these human-centric structures serve as effective **regularizers**. By constraining the agent's logic to rules or programs that "make sense" to humans, we prevent it from learning brittle, overfitted strategies that rely on statistical noise rather than true task dynamics. This process effectively instills **human inductive biases** into the machine. In tasks like industrial scheduling or complex strategy games, this has been shown to improve sample efficiency, allowing agents to reach high performance levels much faster than those starting from a clean slate.

From a collaborative standpoint, these frameworks aim to achieve **verifiable and editable behavior**. When an agent's actions can be expressed through a chain of self-explanations, a human partner can not only understand *why* a decision was made but can also directly intervene to correct a faulty rule or update a preference. This bidirectional flow of information is central to the ambitions of PEER.

2.2.1.7 Explaining Policies of RL agents

Several ongoing research projects are dealing with the challenge, motivated for example by the Datacation use case, of explaining not only individual decisions, but the entire decision-making approach of a reinforcement learning agent.

InnateCoder: Accelerating RL with LLM-Derived Skills [17]. In traditional reinforcement learning, agents typically begin their training process "tabula rasa," meaning they start from a clean slate with no prior knowledge of the task. Consequently, they must spend a vast amount of time learning even the most basic movements through thousands or millions of trial-and-error interactions. This high sample complexity makes training inefficient for complex, real-world problems. InnateCoder changes this paradigm by using foundation models (like GPT-4 or Llama 3.1) to endow agents with "innate skills" before they ever interact with their environment.

The motivation for InnateCoder stems from the realization that while Large Language Models (LLMs) possess vast general knowledge about how to solve problems, they lack the low-level precision to control an agent in real-time. For example, an LLM knows that to win a strategy game, a player must first collect resources and then build a base, but it cannot issue the thousands of individual motor or unit commands required to execute that plan. InnateCoder bridges this gap by distilling high-level human reasoning into programmatic options—small, reusable snippets of code written in a Domain-Specific Language (DSL). In this context, an "option" is a temporally extended action that tells the agent how to behave for several steps until a specific goal is reached.

The process begins by providing the LLM with a natural-language description of the problem and the grammar of the DSL. Although the LLM is unlikely to write a perfect program to solve the entire task immediately, its output contains many useful sub-programs. InnateCoder breaks these programs down to extract a library of unique skills. Crucially, it filters these skills based on their "action signatures"—their actual behavior in the environment—to identify thousands of semantically distinct options.

This library is used to induce a semantic space for the agent's behavior. In standard programmatic RL, agents often search in a "syntax space," where they change their code one character or rule at a time. This is often inefficient because many different-looking programs actually do the exact same thing. By searching in the semantic space induced by LLM-derived options, InnateCoder ensures that its search explores programs that are functionally different, making the learning process far more sample-efficient.

The achieved outcome is a system that sets a new state-of-the-art in benchmarks like MicroRTS (a strategy game) and Karel the Robot (a suite of single-agent tasks). In competitive testing, InnateCoder agents outperformed those that had to learn from scratch and even beat human-written policies and deep RL agents that used significantly more computing power. Because the resulting policies are written as code, they remain interpretable and verifiable, potentially providing a powerful and transparent foundation for the collaborative AI assistants developed in the PEER project.

ProRL: Lightweight and Interpretable Policies. Job Shop Scheduling (JSSP)—deciding which machines should process which parts and when—is a vital but mathematically complex industrial SDM task, similar to for example the warehousing challenges faced in the Datacation use case. While deep reinforcement learning can often solve these problems relatively well, the resulting neural networks tend to be too large to run on the cheap "edge" devices used on factory floors, and their logic is too opaque for factory operators to trust.

We have developed a framework called ProRL to produce interpretable programmatic **policies**, applied to scheduling in our first implementation. Instead of a neural network, ProRL solves tasks by generating simple programs using a Domain-Specific Language. These programs use "if-else" logic to choose between well-known Priority Dispatching Rules (PDRs)—human-readable heuristics like "Shortest Processing Time" or "First In, First Out". The decision-making is steered by five abstract concepts that humans naturally understand, such as "Machine Load Balance" and "Job Remaining Time".

To find the best program, ProRL uses a bilevel optimization process. At one level, it searches for the best "architecture" of the program (the sequence of if-else statements); at the other level, it uses Bayesian Optimization to tune the parameters (the exact thresholds that trigger a change in rules). Crucially, this process does not require a "pre-trained" neural network to act as a teacher.

The achieved outcome is a policy that is verifiable and lightweight. In tests on standard scheduling benchmarks, ProRL not only outperformed traditional manual rules but also beat complex neural networks, particularly on large-scale tasks where rewards are sparse (meaning you only find out if you did a good job at the very end). Furthermore, because the policies are programmatic, they can be translated by an LLM into plain text explanations like: *"When resources are balanced, focus on jobs with the most remaining work; otherwise, prioritize the shortest available operation"*. We believe that similar programmatic policies in appropriate domain-specific languages will be able to offer interpretability and verifiability in many domains—as well as the option for human users to inspect and edit the learned behaviors, which represents a form of **collaborative problem-solving**.

DiPRL: Bridging Continuous Learning and Discrete Code. A fundamental challenge in programmatic RL is the "discretization gap". To use powerful mathematical tools like policy gradient algorithms, researchers often "relax" a discrete program into a continuous version—essentially a fuzzy distribution where the AI considers many possible program branches at

once. However, at the end of training, this must be "snapped" into a single, clean piece of code (discretization). This sudden jump often causes a catastrophic drop in performance because the program "forgets" important nuances it learned in the fuzzy state.

DiPRL is a novel methodology we designed to avoid this "performance collapse". Instead of discretizing the program at the end, DiPRL uses a gradual commitment strategy during the learning process itself. It achieves this through Program Architecture Entropy Regularization.

In this context, "entropy" measures how uncertain the AI is about which program branch to use. High entropy means the AI is still exploring many "fuzzy" possibilities; zero entropy means it has committed to one clear, discrete program. DiPRL includes a penalty for high entropy in its reward function, forcing the AI to slowly "prune" away alternative branches as it learns. By the time training is finished, the program has naturally converged to a single, high-performing discrete program without any sudden jumps.

The results show that DiPRL maintains stable performance throughout training, avoiding the "entropy cliff" where previous methods fail. In tasks like "DoorKey" or "HalfCheetah", DiPRL achieved performance competitive with black-box neural networks while resulting in code that is fully interpretable. This could make it a vital tool for the PEER project's goal of creating AI that is both highly capable and completely transparent. We expect, for example, a tool that supports logistics engineers to be more transparent and trustworthy if it can spell out the heuristics it uses in human-readable form.

2.2.2 Information Flow from User to AI (user modelling and feedback)

2.2.2.1 Translating Unstructured User Preferences and Feedback into Structured Decision Parameters

We study the problem of information flow from user to AI in human-in-the-loop decision-support systems, where qualitative and potentially ambiguous user input must be translated into formally constrained decision parameters. Rather than focusing on the collection of feedback itself, the work investigates how heterogeneous and possibly ambiguous user expressions can be transformed into formally constrained decision parameters that are compatible with planning and optimization algorithms [8], [9].

Direct feedback is captured when operators explicitly modify solver penalties, adjust constraint limits, or refine operational priorities (e.g., prioritizing retrieval efficiency during peak demand) by means of an interactive graphical user interface. User input does not always directly correspond to a solver variable (e.g. objective weights, penalty parameters). It may take the form of short textual comments, conversational instructions, or high-level statements such as “reduce congestion in this area.” To address this gap, we investigate an the use of an intermediate interpretive layer based on generative AI agents. Recent work has shown that LLMs can support structured reasoning and tool use in decision-support settings [18], [19],

[20]. Building on this, the translation process consists of two tightly connected stages, which are illustrated below through their application to the Warehouse configuration management (Datacation) use case:

1) Semantic Interpretation and Contextualization: User input is interpreted in relation to the current warehouse configuration, operational constraints, and previous decisions and the user's intent is recognized. The agent makes use of tools to query the current models and internal rule sets where necessary to disambiguate instructions and ensure that interpretations are context-aware.
2) Constrained Parameter Inference and Structured Validation: Interpreted preferences are converted into quantitatively bounded solver parameters. This process is guided by predefined schemas and admissible parameter ranges. Prompting strategies require the agent to generate outputs in typed formats (e.g., structured JSON objects), and outputs are automatically validated for both content and structure before being exposed to the solver. All inferred parameters remain subject to predefined feasibility bounds and schema validation, ensuring that user-driven adjustments are safely integrated into the optimization workflow. The mathematical guarantees of the optimization model remain intact, while users retain meaningful influence over objectives and constraints.

In addition to explicit feedback, the system may consider indirect signals such as repeated corrections, frequent rejection of certain solution characteristics, or consistent emphasis on specific trade-offs. These can be interpreted as underlying user preferences, treated as contextual cues rather than fixed user profiles. They may influence subsequent parameter suggestions, for example by slightly adjusting default weights or highlighting alternative configurations. This form of adaptive interpretation does not require intrusive profiling or explicit preference elicitation at every step. In future iterations, these interaction logs can serve as a basis for fine-tuning or adapting the translation model itself.

Overall, this line of work investigates how natural language interactions and non-predefined information about both the user and the environment can be systematically integrated into optimization workflows, while maintaining robustness and auditability.

2.2.2.2 Active Representation Learning in Anomaly Detection Task

To complement our pipeline and the methods described in Sections 2.1.1, 2.2.1.3, we propose a human-in-the-loop mechanism to validate and resolve ambiguities near the decision boundary, leveraging expert input for critical cases where automated methods struggle. This enables flexible use of our approach as either a standalone detection system or a data labeling tool compatible with any anomaly detection algorithm, enabling practical deployment of anomaly detection systems under limited labeling budgets. We employ human-assisted refinement of the threshold estimation on the ambiguous areas, allowing the method to

achieve a clearer separation between normal and abnormal regions, improving the quality of anomaly detection.

To integrate human expertise into the semi-supervised learning process, we compute an *ambiguity score* (more details in Section 3.3.5 and our recent publication) for each instance representing its proximity to the decision threshold. These ambiguous instances are then prioritized for manual inspection and possible re-labeling, enabling iterative refinement.

This human-in-the-loop step effectively reduces false positives and false negatives by leveraging human judgment in regions of uncertainty, ensuring higher robustness and interpretability of the anomaly detection pipeline in industrial contexts.

As shown in Table 1, the evaluation after human-based refinement highlights a shift in method effectiveness, with Cluster and Mean Distance emerging as the most robust and generalizable approaches across diverse anomaly types.

*Table 1 Average F1 score and standard deviations across methods **after** human-based refinement*

Dataset	Cluster	Mean Distance	Pure Patchcore	Silhouette	Wasserstein
Bottle	1.000 ± 0.000	1.000 ± 0.000	0.184 ± 0.000	0.942 ± 0.013	0.977 ± 0.026
Cable	0.898 ± 0.074	0.868 ± 0.040	0.115 ± 0.001	0.765 ± 0.105	0.836 ± 0.167
Capsule	1.000 ± 0.000	0.638 ± 0.080	0.650 ± 0.436	0.394 ± 0.035	0.776 ± 0.107
Carpet	1.000 ± 0.000	0.899 ± 0.076	0.222 ± 0.255	0.784 ± 0.034	0.816 ± 0.086
Grid	0.385 ± 0.143	0.884 ± 0.058	0.085 ± 0.000	0.339 ± 0.118	0.831 ± 0.090
Hazelnut	0.980 ± 0.030	0.934 ± 0.029	1.000 ± 0.000	0.821 ± 0.134	0.934 ± 0.029
Leather	0.970 ± 0.018	0.988 ± 0.014	0.142 ± 0.000	0.942 ± 0.031	0.966 ± 0.032
Metal Nut	0.903 ± 0.165	0.930 ± 0.043	1.000 ± 0.000	0.830 ± 0.075	0.926 ± 0.036
Pill	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.427 ± 0.140	1.000 ± 0.000
Screw	0.452 ± 0.378	0.665 ± 0.219	0.144 ± 0.000	0.298 ± 0.098	0.745 ± 0.165
Tile	0.959 ± 0.062	0.921 ± 0.048	0.142 ± 0.000	0.771 ± 0.123	0.911 ± 0.026
Toothbrush	0.837 ± 0.027	1.000 ± 0.000	1.000 ± 0.000	0.791 ± 0.111	1.000 ± 0.000
Transistor	0.972 ± 0.054	0.738 ± 0.130	0.087 ± 0.000	0.596 ± 0.229	0.805 ± 0.184
Wood	0.808 ± 0.148	0.860 ± 0.053	0.156 ± 0.040	0.517 ± 0.085	0.854 ± 0.048
Zipper	1.000 ± 0.000	0.843 ± 0.092	0.266 ± 0.324	0.594 ± 0.103	0.854 ± 0.086

These findings suggest that distance-based and clustering-enhanced methods offer stronger baseline performance, especially when refined with appropriate hyperparameter tuning.

However, while relabelling based on clustering and distance metrics can significantly boost anomaly detection performance, the size of the ambiguity set—i.e., the number of samples requiring manual inspection and relabelling—can sometimes be prohibitively large.

This introduces a substantial overhead in human effort, which may diminish the practical gains of relabelling. For instance, in categories like Grid and Capsule, the ambiguity set spans a wide range, making it difficult to justify the manual cost despite performance improvements.

Therefore, a sweet spot must be identified—a trade-off between maximizing post-relabelling performance and minimizing the size of the ambiguity set.

2.3 User - AI Conversation Management (Task 3.3)

According to DoA: The main goal of this task is to integrate the improved problem-solving techniques developed in Task 3.1 and the extended information flow developed in Task 3.2 into a consistent, ongoing conversational interaction loop between users and AI. In this section, we study methods that will allow a) discuss problems and solutions on the desired human-specified level of abstraction (e.g., instead of using raw feature values, use granules of information which carries more semantic information), and b) allow to store, integrate and re-use the knowledge that was generated within previous interactions with the system to close the loop of human-AI collaboration. This will form a logical backend for the use-case specific interface (such as visualizations, textual conversations, etc.).

2.3.1 Explaining Actions of Search Algorithms

As part of our ongoing work on explaining the actions of search algorithms, we are further studying Monte Carlo Tree Search (MCTS) as a cornerstone of modern planning, used to simulate thousands of possible futures to find an optimal path. The size and complexity of the resulting search trees effectively turn them into "black boxes" to the human operators who must trust the recommended actions. Motivated by applications of routing and pathfinding algorithms in the PEER project, we have studied this with domain experts such as transit dispatchers, who need to know not just *what* routing decisions an AI suggests, but also *why* they are safe or efficient.

LogiEx [21], [22] addresses this by integrating LLMs with symbolic reasoning. While standard LLMs can be imprecise or "hallucinate" technical details, LogiEx uses the LLM as a front-end to translate natural language questions into formal logic expressions (specifically Computation Tree Logic, or CTL). These logic statements are then used to mathematically check the MCTS search tree for factual evidence.

The framework categorizes expert queries into three types to ensure comprehensive support. Category 1 (Post-hoc) answers factual questions about the existing plan, such as "Why was vehicle A chosen over B?" by aggregating reward data from the search tree. Category 2 (Human-Guided Search, or HuGS) allows users to explore hypothetical "what-if" scenarios, such as "What if vehicle 1 breaks down?". In these cases, LogiEx triggers *new* MCTS simulations to find the answer. Finally, Category 3 provides background knowledge about the algorithm's constraints or general logic using Retrieval-Augmented Generation (RAG). All answers are provided in natural language.

The achieved outcome is a system that significantly outperforms basic LLM approaches in factual consistency—achieving a 7.9x improvement over Llama 3.1 in one tested metric. In user studies with transit experts, LogiEx was rated as more understandable, satisfying, and reliable than traditional tree visualizations. This for the first time takes a step towards

interactive "back-and-forth" conversations for the dispatchers, allowing them to refine their understanding of the complex planning results of MCTS.

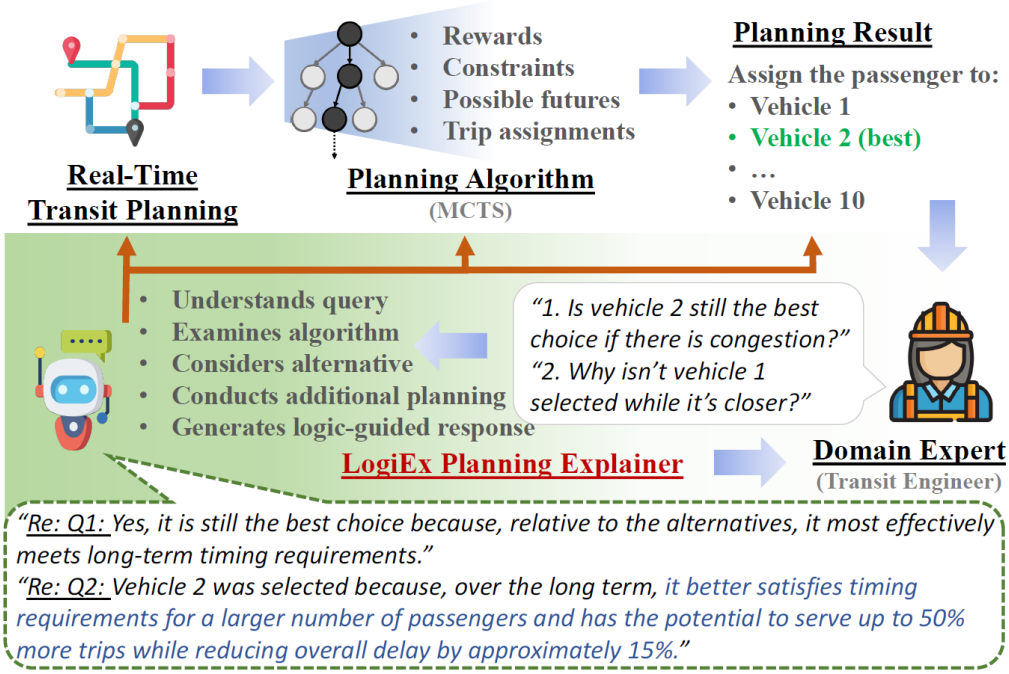


Figure 10 LogiEx explains planning by combining domain knowledge, the search process, and logical reasoning

2.3.2 Multimodal Models as Anomaly Detector with Metadata-Augmented Prompts

We study the task of in context learning and human–AI interaction in multi-modal visual anomaly detection [23], where an AI assistant must support users with, context-aware image anomaly classification. This work is considered as a baseline and provide the user with text-based explanation and will be evaluated against PatchCore anomaly detection algorithm.

We argue that with the rapid advancement of multimodal foundation models, post hoc fusion is a viable alternative. Unlike early fusion, post hoc fusion does not require retraining or modifying large models. Instead, it leverages pre-trained unimodal or multimodal models to generate decisions independently and fuses these outputs afterwards using lightweight models or rule-based systems. This allows for greater flexibility in selecting anomaly detection models, a model-agnostic approach to explainability and customization of explanations to fit stakeholder needs through LLM capabilities such as rephrasing and summarization.
Here the central objective is to explore how these advanced models can be leveraged detect anomalies in visual and prompt engineered dialogue. This research focuses on analyzing the following topics:

- Investigate how textual prompts, particularly those incorporating domain-specific expert knowledge, can be designed to guide the model's inference process, with the goal of improving accuracy and contextual relevance.

- We incorporate diverse input modalities—including metadata such as heatmaps, segmentation masks, and image patches—derived from models like PatchCore. These additional data types are used within prompts to help the model attend to salient regions and provide richer contextual information during inference.
- Rather than relying solely on displays or post hoc visualizations, our approach highlights the potential of integrating natural language interaction to support a more dynamic and user-driven inspection process. This perspective encourages deeper engagement, improved interpretability, and lays the groundwork for more interactive forms of human-AI collaboration.

We evaluated the multimodal model's ability to classify images as normal or anomalous. Each test image is paired with a random normal image, and the model must decide if the test image is good or anomalous. Classification performance is measured by the F1 score and compared to the PatchCore model. To investigate this, we designed two versions of the experiment to investigate whether integrating knowledge from PatchCore enhances anomaly classification performance. (1) In the first setup, the multimodal model receives a normal image along with a textual description of possible anomaly types based on expert knowledge. The model is tasked with classifying the anomaly type present in the test image. (2) In the second setup, the model is provided with a normal image, an anomalous image, the corresponding anomaly heatmap/segmentation/extracted patches and the same expert-provided descriptions of potential anomalies. The extracted patches correspond to the most anomalous regions, with non-anomalous areas masked—rendered black (see Figure 11). The objective remains to classify the type of anomaly, with the additional visual and semantic context.

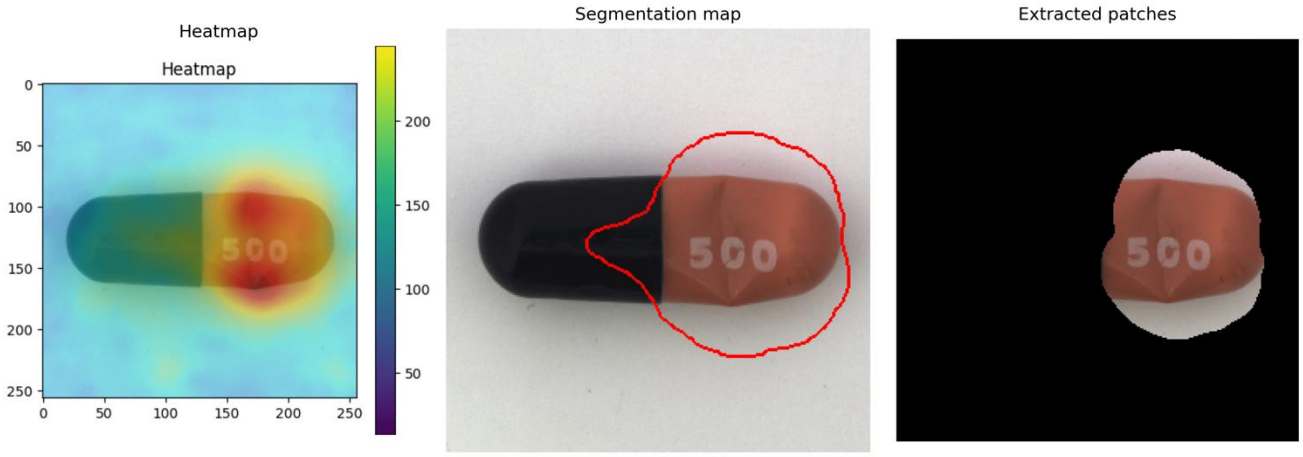


Figure 11 Different modalities produced by the PatchCore model—heatmaps, segmentation maps, and patches—using an example of a squeeze anomaly from the capsule dataset

As observed in our investigations, only the Wood dataset achieved more than 50% accuracy in correctly classifying anomaly types. In general, classification performance on texture datasets outperformed that on object datasets. Furthermore, incorporating additional metadata extracted from PatchCore—such as heatmaps, segmentation maps, or patch-based representations—did not lead to a significant improvement in classification accuracy. In the Toothbrush dataset, only a single anomaly type is present; therefore, the reported *100%* accuracy should not reflect the model's general classification capability.

3. Realization of PEER Companion Agents in Use Cases

This section presents the developments in mapping and instantiation of PEER modular architecture [24] within four use cases. Figure 12 depicts the generic PEER companion (We refer the reader to Deliverable 3.1 for more details).

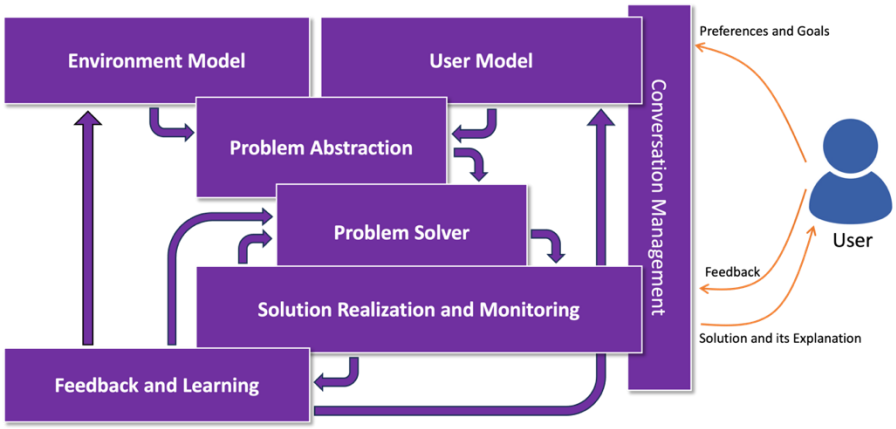


Figure 12 Architecture of the PEER Companion Agent revisited from D3.1.

3.1 Sonae. A Personalized In-Store Navigation Assistant for Customers (SigaApp)

The main analytical challenges are being tackled as extension to MVP#1 are shown in Figure below. This includes a descriptive problem to digitize the retail store, a predictive problem to make product recommendations, a static prescriptive problem where a picking route is obtained before starting the shopping journey, and a dynamic prescriptive problem where route and product suggestions are integrated Figure 13.

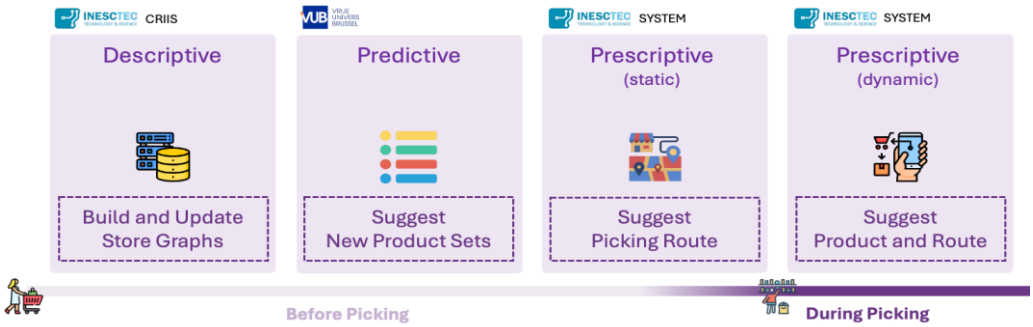


Figure 13 Main analytical challenges being tackled in Sonae use case.

The developments of the referred analytical challenges are being guided by Sonae, making sure that all the data and the solution adhere to the real-world problem they face. In the

following subsections, we describe the developments achieved in each building block of the collaborative and adaptive decision system being developed in the retail use case.

3.1.1 Environment Model

During this review period, the Environment Model (e.g., store model) was significantly expanded to reflect a realistic large-scale retail environment. The current implementation represents one of the largest retail stores in Portugal and is discretized into approximately 4,500 nodes (see. Figure 14).

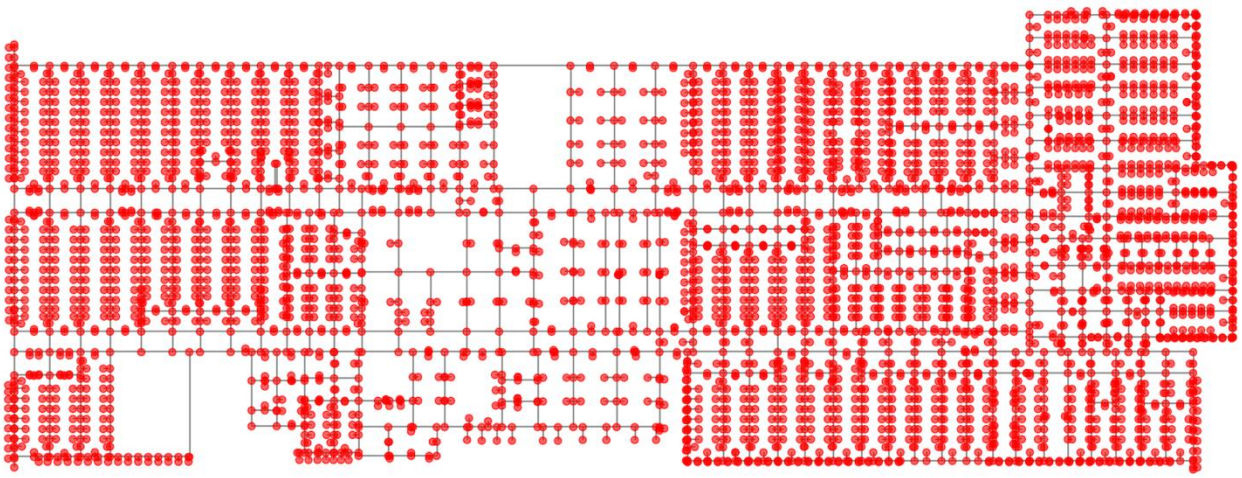


Figure 14 Current retail store layout.

Each node corresponds to a physical location (e.g., shelf segment, corridor position, or checkout area), and edges encode walking distances between adjacent positions. This graph-based representation enables modeling realistic walking paths and aisle constraints, associating products to shelf-level coordinates, and ensures compatibility with real-world deployment conditions.

A major development introduced during this period concerns the advances in the automation of store digitization. INESC TEC developed an approach based on autonomous robot navigation combined with computer vision techniques. The robot captures shelf images and associates detected products with spatial coordinates, enabling automated construction of the store graph and product-to-node mappings (see Figure 15).

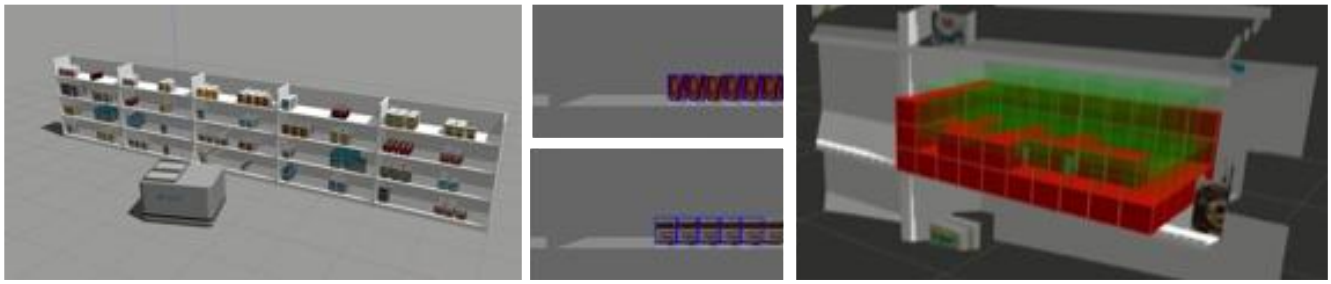


Figure 15 Simulation environment to test the graph generation approach.

At the current stage, (1) the graph generation approach is validated within a simulated environment, (2) full graph of a large-scale store has been successfully constructed, and (3) the resulting graph is fully compatible with the routing and optimization modules under development, as well as the frontend of the mobile app. This development reduces manual configuration effort and increases scalability across different retail environments.

3.1.2 User Model

The User Model captures customer purchasing preferences and supports personalized product recommendations prior to the start of the shopping journey.

As the product recommendation module works directly with user-item interaction data, this functionality is implemented and experimented with Sonae's real and fully anonymized dataset. The company dataset consists of the past 3 years of data from 320k customers and 13 million orders. The dataset includes anonymized customer id, purchase date and time, order ID, product ID, product category and subcategory. This dataset is close to open-sourced Instacart dataset in terms of fields, context, and sparsity.

Table 2 Statistical summary of the MC Sonae dataset (January 1, 2024 up to May 30, 2025)

Property	Value
Number of users	193, 204
Total orders	1, 582, 864
Distinct products	115, 104
Transactions	24, 465, 237

Physical retail customers typically make purchasing decisions without providing direct feedback on products, which limits the applicability of traditional collaborative filtering and many modern neural recommendation approaches that depend on explicit user–item interactions. At the same time, companies like Sonae generates rich volumes of implicit customers' transactional data Table 2, including items purchased together and regular purchasing patterns. Despite the abundance of these signals, implicit feedback presents two main challenges. First, user preferences have to be inferred from item-purchase counts instead of (partial) direct ratings. Second, there is a large amount of missing interaction data to deal with. In sparse datasets, which are common in retail, the lack of interaction cannot simply be treated as negative feedback because it might just be an unobserved event. Additionally, implicit data tends to be noisy, which adds to the model complexity and makes models more difficult to interpret.

While methods such as Matrix Factorization (MF) effectively capture user–item interaction patterns and can leverage implicit feedback such as purchase counts, they ignore temporal structure and therefore cannot predict *when* an item is likely to be needed again. While neural nets[25] and Transformer-based models have been successfully applied to Next Item Recommendation (NIR) due to their ability to model sequential user behavior, their computational complexity poses significant challenges in NBR due to highly repetitive/sparse

interactions[26]. Long basket sequences, large item catalogs, and the need to predict sets of items exacerbate the training and inference costs of such architectures, motivating the development of more efficient and scalable methods for NBR.

To fill this gap, we propose a hybrid model (Figure 16) that combines MF-based preference modeling with a lightweight, interpretable temporal component derived from cyclic analysis of user–item purchase behavior, enabling personalized and time-aware suggestions for next basket recommendation [27].

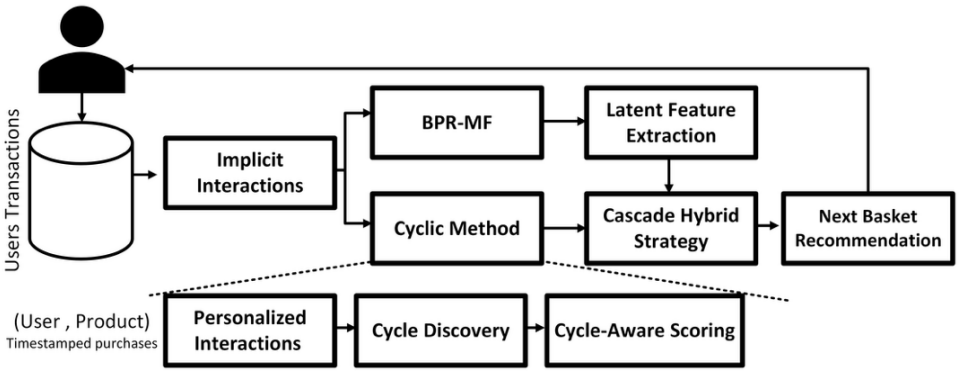


Figure 16 The proposed light-weight next basket recommendation pipeline

Our method jointly models basket-level interactions and cyclical temporal shopping patterns to more accurately reflect how customers engage with in-store shopping. In doing so, it supports recommendation decisions based on personal purchasing habits rather than short-term profit-maximization signals, aligning with real consumer behavior. The approach is transparent, computationally efficient, and compatible with state-of-the-art implicit-feedback and sequence-modeling techniques, while remaining simple enough for deployment in real-world retail systems.

As shown in Figure above, our proposed recommendation model combines the cyclic-aware information with classical MF-based techniques to model both habits and exploration.

Here provide some quantitative results based on analyzing the performance of the proposed method on the our use case using real dataset.

Table 3 Quantified metrics for product recommendation (Sonae dataset)

K	Precision@K	Recall@K	F1@K	NDCG@K
5	0.3217 ± 0.0091	0.0387 ± 0.0034	0.0581 ± 0.0041	$\mathbf{0.3346} \pm 0.0092$
10	0.2871 ± 0.0090	0.0623 ± 0.0050	0.0847 ± 0.0053	0.3094 ± 0.0093
20	0.2500 ± 0.0075	0.0964 ± 0.0070	0.1148 ± 0.0061	0.2835 ± 0.0088

Table 3 reports the performance of the proposed method across different cutoff values. As expected, precision and Normalized Discounted Cumulative Gain (NDCG) decrease with increasing K, reflecting the growing difficulty of maintaining highly relevant rankings as more

items are recommended. Conversely, recall and F1 increase steadily, indicating improved coverage of relevant items at larger cutoffs. Notably, the model achieves its highest ranking quality at *K=5 (NDCG@5 = 0.3346)*, highlighting its effectiveness in prioritizing relevant items at the top of the recommendation list. At *K=10*, the results exhibit a favorable balance between precision and recall, while *K=20* emphasizes recall-oriented performance. These results demonstrate the robustness of the proposed method across different recommendation scenarios.

3.1.3 Problem Abstraction

The Problem Abstraction Module formalizes the integrated routing and product suggestion task at two complementary levels. A static optimization formulation for route computation (from the previous review period), and a sequential decision formulation for dynamic recommendation control. Together, these define the decision space over which decision policies operate.

Static Optimization Formulation (MILP): Given the large scale of the store graph, solving the routing problem over the full environment at each decision step would be computationally inefficient. For this reason, a relevant subgraph is dynamically extracted prior to optimization. This subgraph includes the customer's current location, the set of remaining products in the shopping list (mandatory nodes), the set of candidates recommended products generated by the User Model (optional nodes), and the checkout location as the terminal node. On this subgraph, the routing problem is formulated as a Mixed-Integer Linear Programming (MILP) model. The objective is to determine a feasible picking route that visits all mandatory products exactly once while optionally incorporating one or more recommended products, minimizing total walking distance including any detours. The formulation includes binary decision variables representing edge traversal, flow conservation constraints to ensure route continuity, visit constraints guaranteeing that all mandatory products are collected, and boundary constraints defining the start node (current customer position) and end node (checkout). Optional product inclusion constraints allow the integration of a recommended product into the route when selected. This optimization model can be repeatedly invoked whenever the shopping list changes or when a recommended product is accepted. Within the overall architecture, the MILP acts as a route-optimization oracle.

Sequential Decision Formulation (MDP): While the MILP computes an optimal route for a given shopping list configuration, the overall shopping assistance problem is inherently sequential. At each product scan event, the system must decide whether to suggest a new product and, if so, potentially trigger a route recomputation. This process is formalized as a Markov Decision Process (MDP). The state captures the relevant decision context, including the customer's current position, the remaining shopping list, the set of candidates recommended products, the current route, and the customer profile or persona. At each decision step, the action consists of either suggesting one product from the candidate set or refraining from making a suggestion. If a product is suggested and accepted by the customer,

it is added to the shopping list and the MILP model is invoked to compute a new optimized route incorporating the additional item. In this way, the MILP solution becomes part of the action execution mechanism inside the MDP. State transitions occur whenever the customer scans a product, deviates from the suggested route (i.e., scan a product not in the shopping list), or updates the shopping list without scanning any product. The reward structure is to be conceptually defined to balance route efficiency, customer adherence, acceptance of suggested products, and avoidance of unnecessary detours.

To exemplify the importance of such formulation, let us analyse the case where we just use a recommendation system such as the one in the User model. The product recommendation system takes as input the customer profile/persona and the current shopping list and predicts the most probable related product purchases. The most straightforward approach would be to recommend related products after the customer identifies what they want to buy. However, this recommendation would be static and independent of the valuable information that emerges during the shopping journey. Therefore, we developed a dynamic product recommendation system.
The static version of product recommendation is illustrated in Figure 17. Once the shopping list was defined, a fixed set of related products was suggested. These corresponded to the recommended items (e.g., yellow nodes) associated with the selected shopping list products (e.g., green nodes).

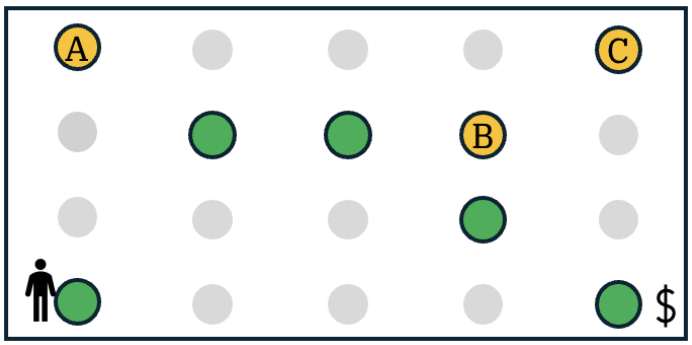


Figure 17 Static product recommendation.

What was missing to enable dynamic product recommendations was the ability to suggest products mid-route. This requires combining the recommendation system with route generation and stochastic customer behavior. Taking the example in Figure 17: given a shopping list (green dots), a list of recommendable products (yellow dots) is identified. Subsequently, a route is generated that starts at the customer's position, ends at the payment stations, and passes through all nodes containing shopping list products. Given the generated route and the identified recommended products, there are different ways to dynamically suggest a product at any given moment:

- Recommend **Product A**, resulting in minimal detour
- Suggest **Product B**, resulting in no detour at all

- Recommend **Product C**, which would require a significant detour but may be worthwhile if it's a product the customer really wants
- **Wait** and refrain from recommending any product for now, gathering more information to guide the decision

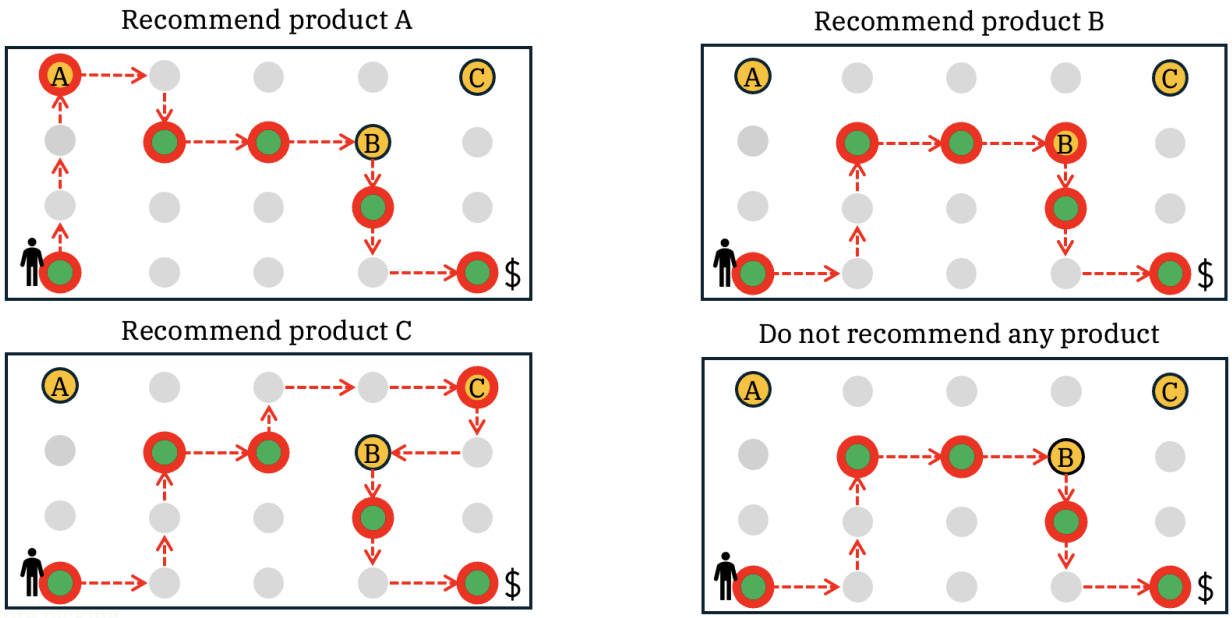


Figure 18 Dynamic product recommendation examples.

In this sequential decision-making setting, the product to be recommended can change due to two main triggers (see Figure 19):

- Customer behavior resulting in a detour (not following the proposed route)
- Customer updating the shopping list (adding or dropping products)

If the shopping list is updated, both the route and the list of recommended products are updated. If the customer detours, the route must be updated. Consequently, if either the route or the list of recommended products changes, the dynamic product suggestion may also change.

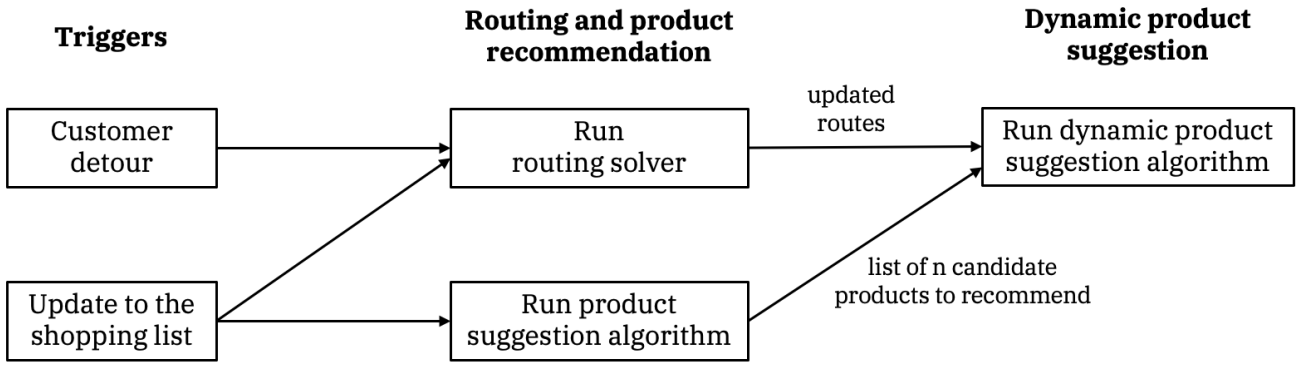


Figure 19 Triggers for the dynamic product suggestion.

Consider a customer navigating the store with a predefined shopping list and a recommendation system that suggests products according to the his/her profile. Some of

those products are close to the existing route, while others would require a major detour. Our algorithm proposes the best-fit product that still minimizes the detour. Later in the shopping journey, the customer adds eggs, milk, and chocolate to the shopping list, likely indicating an intention to bake a cake. By updating the shopping list, the list of candidate products to be recommended is updated, and flour may become a good suggestion.
In summary, both the route and the product recommendations must be analyzed in an integrated, dynamic way. The best recommendation changes throughout the shopping experience based on two key factors: (1) customer behavior, such as detouring to unexpected parts of the store, and (2) shopping list updates. The main insight is that routing information and product recommendations should be tightly coupled, as valuable information generated throughout the shopping journey directly influences which products should be suggested at each moment.

3.1.4 Problem Solver

The Problem Solver module implements the computational mechanisms used to solve the optimization and decision problems defined in the Problem Abstraction Module. It consists of two complementary components: the routing solver used to compute feasible picking routes and the decision policy governing dynamic product suggestions.

Routing Solver: The routing problem defined through the MILP formulation can be solved either using an exact optimization solver or through heuristic methods. While exact solvers guarantee optimality, their computational time increases significantly with the size of the problem instance, particularly in large-scale retail environments and when frequent route recomputations are required.
Given the real-time constraints of the mobile application and the increased store size introduced in this MVP, a heuristic-based routing solver is currently used as the default implementation. The heuristic provides high-quality solutions with substantially reduced computation times, enabling dynamic rerouting during the shopping journey without noticeable delays for the customer.
The exact MILP solver remains available as an alternative solution method. Its use becomes particularly relevant when additional operational constraints must be incorporated into the model, as exact formulations provide greater flexibility for constraint modeling and extensibility. Further refinement of the solver is needed to make it scalable to settings with a large number of products. This improvement does not necessarily relate directly to the solution method, but rather to preprocessing steps such as aggregating products from the same corridor into the same node and redefining the trigger for re-optimizing the route, as rerouting at every product scan may not be beneficial when changes are minimal or nonexistent. This dual-solver design ensures both scalability in the current MVP and robustness for future extensions.

Sequential Decision Policy: At the current MVP stage, the decision policy governing this MDP is rule-based and myopic. However, the formal MDP structure establishes the foundation for future learning-based policy optimization to be developed and validated in a simulator environment in the next project phase.

At each product scan event, the system evaluates the set of candidate recommended products provided by the User Model. From this set, a single product may be selected for suggestion based on a simple proximity criterion. Specifically, the current policy suggests the product whose inclusion would result in the smallest detour relative to the existing route. If no suitable product satisfies the predefined condition, no recommendation is made.

When a product suggestion is accepted by the customer, it is added to the shopping list and the routing solver is invoked to compute a new route incorporating the additional item. If the suggestion is rejected, the current route remains unchanged.

Note that the proximity criterion (recommend the closest product) can be changed, resulting in additional myopic policies such as recommend the farthest product, recommend product that minimizes detour (for efficiency-focused customer), or recommend product that maximizes detour (for customer who want to explore the store).

These myopic strategies do not account for long-term effects of decisions and does not learn from historical customer behavior. However, they provide stable and interpretable baseline policies that enable system integration and real-world testing. The formal MDP structure allows this policy to be replaced in future iterations by learning-based approaches that optimize decision timing and product selection based on observed customer responses.

3.1.5 Solution Realization and Monitoring Module

Solution realization and monitoring are critical in this use case, as they provide the basis for triggering the rerun of routing and recommendation algorithms. Some triggers are independent of solution realization, such as changes to the shopping list or potential time-based triggers. These events directly require algorithm updates regardless of how closely the customer follows the proposed route.

However, other triggers depend explicitly on comparing the planned solution with actual user behavior. A key example is customer detours – when a customer selects a product that is not the next node in the proposed route. In such cases, rerouting requires actively monitoring and comparing the recommended path with the path effectively followed in the store. This continuous comparison enables the system to detect meaningful deviations and decide when recalculation is necessary.

As future work, storing detailed information about the followed route and its deviations from the proposed one will be essential. This historical data can support the training of learning algorithms, allowing future routes and recommendations to become increasingly tailored to individual customer preferences and behavioral patterns.

3.1.6 Feedback and Learning Module

The current MVP implements basic behavioral monitoring mechanisms to support future learning-based extensions. The system records route adherence, detected deviations, and acceptance or rejection of dynamic product recommendations. These signals provide structured feedback regarding customer interaction with the suggested route and recommendations. At the current stage, this information is used for performance analysis and system evaluation rather than online learning.

The formal MDP defined in the Problem Abstraction Module establishes the foundation for future reinforcement learning approaches. In the next MVP phase, learning-based policies will be trained in a retail store simulation environment using simulated customer behaviors before potential deployment in real-store settings.

3.1.7 Conversation Management Module

The Conversation Management Module governs how the system communicates decisions to the user during the shopping journey. In the current implementation, interaction is structured and interface-driven. Dynamic product suggestions are presented through a pop-up mechanism that allows explicit acceptance or rejection. Route updates are communicated visually through the navigation interface. This interaction design ensures transparency of system decisions while minimizing cognitive load.

Future developments may extend this module to include richer explanatory mechanisms and adaptive communication strategies tailored to user profiles.

3.1.8 Next steps

The next phase of development will focus on advancing the system along four main dimensions: environment automation, predictive personalization, sequential policy learning, and controlled evaluation.

From the environment perspective, the automated store digitization process will be validated in additional retail settings. The objective is to assess the robustness and transferability of the robot-based graph construction pipeline and reduce manual configuration effort when onboarding new stores.

At the user modeling level, the predictive recommendation module will be extended to incorporate contextual signals from the current shopping cart. The current implementation generates recommendations independently of the active basket, which may not fully capture short-term shopping intent. Introducing cart-aware modeling will allow the system to distinguish between routine purchases and situational shopping moments, improving personalization accuracy.

Within the decision layer, the rule-based myopic policy will be replaced by a learning-based sequential policy. Using the previously defined MDP framework, reinforcement learning methods will be developed and trained in a simulator environment with simulated customer acceptance and rejection behavior. This will enable the system to learn not only which product to recommend, but also the optimal timing of recommendations and whether a detour is justified given the customer profile and route context. This also opens further opportunities for explanation techniques from the field of explainable reinforcement learning.

Finally, a more structured evaluation campaign will be conducted to measure the impact of the system on customer experience and efficiency. The planned experimental design will compare different conditions, including customers shopping without the application, with a static map, with route guidance only, and with route guidance combined with dynamic recommendations and explanations. This evaluation will provide quantitative evidence of the added value of the collaborative and adaptive decision framework.

3.2 City of Amsterdam. A Personalized Navigation App for Wheelchair Users

3.2.1 Environment Model

The data for Osdorp-Midden city (road) infrastructure is stored as a graph and builds this data structure as an adjacency matrix with a nested dictionary. Each key in the dictionary is a vertice (or node) and the value for that vertice is another dictionary representing its neighbors (with a directed edge) and the properties of the objectives stored by the edge. The graph representing Osdorp-Midden contains 30503 nodes and 71446 edges. The dataset will further be pre-processed and abstracted as explained in Problem Abstraction module.

3.2.2 User Model

Here we assume that users do not or cannot spend a lot of time comparing all optimal solution alternatives (Pareto front) at once. At such times, they maybe have time to express they opinion on the components of objective function and compare two/three routes at best and go for the best of those. In this scenario, it may not be useful to pre-compute the entire Pareto front. Instead, we just need to output the routes that will provide the best option in terms of expected utility and let them pick the alternative that will maximize their utility. We assume the user will select the one that satisfies his/her preferences and can compare the trade-offs of the two presented routes numerically.
Based on the use case's accessible route data, we are considering a user willing to optimize for the list of objectives and constraint below:

- path length
- crossing

- bike path length
- walk path length
- width of the path (constraint – mapped directly from wheelchair information)

3.2.3 Problem Abstraction Module

The Problem Abstraction Module has not been a priority for this deliverable D3.2. However, we have employed several kinds of abstraction in our ongoing research which could prove useful for the final deliverable D3.3. For example, we have developed a graph simplification technique that takes the original map data from the municipality of Amsterdam as input, and returns a simpler map that is computationally easier to handle by detecting redundant edges, and edges that can be combined with each other without any loss of routing options. Additionally, we have studied route-planning scenarios which were reduced to four objectives, simplifying not only the computation of Pareto fronts, but also the explanations of potential trade-offs we could produce for the user. It could make sense to operate in such lower-dimensional spaces in particular for users who do not have strong preferences in all objectives our datasets and algorithm can in principle handle.

3.2.4 Problem Solver

This subsection proposes a Multi-Objective Depth First Search (MO-DFS) method to solve Multi Objective Shortest Path Problem (MOSSP) by using Pareto dominance pruning and guiding Oracle search with scalarization methods by using distance measures to an upper point reference vector (strictly Pareto dominating the referent that the Oracle receives).

Our goal is to use a multi-objective path planning optimization solver as the Oracle to iteratively find new feasible and suitable Pareto-optimal route solutions. Each route has a start and goal location with multiple objectives to be preferably minimized. The solver plans a route where the solution path has a value vector to define each objective. Our Oracle uses a multi-objective Depth First Search algorithm to return a suitable route together with its cost vector, namely the objective values, guaranteeing strictly Pareto dominating a given referent.

Multiple types of general solution strategy exist for the oracle to produce a Pareto-optimal solution:

- **Scalarization**: in the case of original IPRO with achievement scalarization functions (scalarization of returned rewards).
- **Direct solution**
    - Multi-objective genetic algorithm (population-based approach): In different solution approaches.
    - Multi-Objective Depth-First Fearch (MO-DFS, path planning-based approach): In our case of PG-IPRO.

Since MO-DFS is a direct solution approach, like genetic algorithms, it can take a lot of time to find a solution due the combinatorial problem characteristic. This requires additional information/data to be able to constraint the search space. A reverse Dijkstra algorithm (see pseudo code Algorithm 1: Reverse Dijkstra) finds the shortest distance starting from the goal node to every other node in the graph. This introduction is an optimized version of value iteration to iteratively calculate lower bounds objective cost for each node to the target node in the graph. This is computed from the target/destination node (goal location) and these shortest distance are actually the lower bound (or minimum cumulative edge) cost value for each objective in an accumulated path to get from the current node to the target.

Require: An objective (number indicating which objective to optimize for)
  $node_values \leftarrow \emptyset$
  $pq \leftarrow$ priority queue initialized with $(0, target)$
  while pq is not empty **do**
    $(cost, current) \leftarrow pq.pop()$
    if $current \in node_values$ **then**
      continue
    end if
    $node_values[current] \leftarrow cost$
    for all $neighbor \in graph[current]$ **do**
      if $neighbor \notin node_values$ **then**
        $edge_costs \leftarrow graph[current][neighbor]$
        $new_cost \leftarrow cost + edge_costs[objective]$
        $pq.push((new_cost, neighbor))$
      end if
    end for
  end while
  return $node_values$

Algorithm 1: Reverse Dijkstra

This further allows for efficient search space pruning by taking into account these lower cost bounds as a distance measure for MO-DFS to guide and constraint the search for potential Pareto-optimal candidate path (route) solutions.

The following algorithm is used in the MO-DFS Oracle (used by PG-IPRO) to find a Pareto-optimal multi-objective path strictly Pareto dominating a referent vector (see pseudo code Algorithm 2):

- First, i**nitialize lower bounds cost per node** (for each objective) with Reverse Dijkstra (Algorithm 1: Reverse Dijkstra).
- **Start MO-DFS** taking the lower bounds, upper bound target region (ideal vector for the target region) and lower bound target region (referent):
    - **Initialize stack** (with zero path costs, the source and the path only containing the source) **and a visited dictionary** to remember for each node the non-dominated cost vectors.
    - **Start stack loop** by popping top element each iteration.

- **Goal check**: A new lower bound update (and new best path found) when target is reached and is strictly Pareto dominating the current lower bound (initially the referent of the target region).
- **Loop over neighbors**: and for each neighbor do the following:
    - Calculate the **new estimated cost** (current cost to reach the node and adding the lower bound cost to optimally get to the target node).
    - **Prune paths** that do not lead to a Pareto-optimal solution by checking if the new estimated cost is strictly worse in any dimension compared to the lower bound or if any of the objectives in the estimated cost are equal to the referent vector. This pruning enables smaller computational overhead and decreases the time in searching for a Pareto-optimal solution that strictly dominates the referent of the target region. Afterwards also prune paths that are Pareto dominated by another option to get to the node (from the visited dictionary).

- **Calculate estimated distance measure** to the upper bound target region (or ideal) with a scalarization method (normalized for the search space at hand).
- **Sort neighbor lists** on the estimate of an optimal distance heuristic measure and **place the elements on the stack** based on this estimated closeness to the upper bound of the target region (lowest distance element at the top of the stack).

Require: Source, target, graph, number of objectives, lower bounds cost per node, lower bound of target region (referent), upper bound of target region
$current_best_path \leftarrow []$
$current_best_cost \leftarrow upper_bound_of_target_region$
$initial_cost \leftarrow [0, ..., 0]$
$stack \leftarrow [(initial_cost, source, [source])]$
$visited \leftarrow \{source : [initial_cost]\}$
while *stack* is not empty **do**
  $(cost, current, path) \leftarrow stack.pop()$
  if $current = target$ **then**
    if $cost \leq current_best_cost$ **then**
      $current_best_cost \leftarrow cost$
      $current_best_path \leftarrow path$
    end if
    continue
  end if
  $neighbor_list \leftarrow []$
  for all $neighbor \in graph[current]$ **do**
    $edge_cost \leftarrow graph[current][neighbor]$
    $new_cost \leftarrow cost + edge_cost$
    $lower_est \leftarrow [lb[neighbor] \text{ for } lb \in lower_bnds]$
    $estimate \leftarrow new_cost + lower_est$
    if $\exists i : estimate_i > current_best_cost_i$ **or** $\exists i : estimate_i = upper_bound_of_target_region_i$ **then**
      continue
    end if
    $is_dominated \leftarrow$ False
    for all $prev \in visited.get(neighbor, [])$ **do**
      if $prev \leq new_cost$ **then**
        $is_dominated \leftarrow$ True; **break**
      end if
    end for
    if $is_dominated$ **then**
      continue
    end if
    $Update(visited[neighbor])$
    $distance \leftarrow scalarization_method(estimate, lower_bound_target_region, upper_bound_target_region)$
    $neighbor_list.append(new_cost, neighbor, distance, (path + neighbor))$
  end for
  Sort $neighbor_list$ by $distance$ descending
  for all $(ncost, n, d, npath) \in neighbor_list$ **do**
    $stack.push(ncost, n, npath)$
  end for
end while
return $(current_best_cost, current_best_path)$

Algorithm 2: Multi-Objective Depth-First Search with pruning and upper bound guidance

As shown in Figure 20, the full set of Pareto optimal solution for a single two-objective scenario of Osdorp-Midden contains 14 solutions which is cognitively hard to process at once. Therefore, using PG-IPRO requires sequential feedbacks and reduce this computational demand significantly.

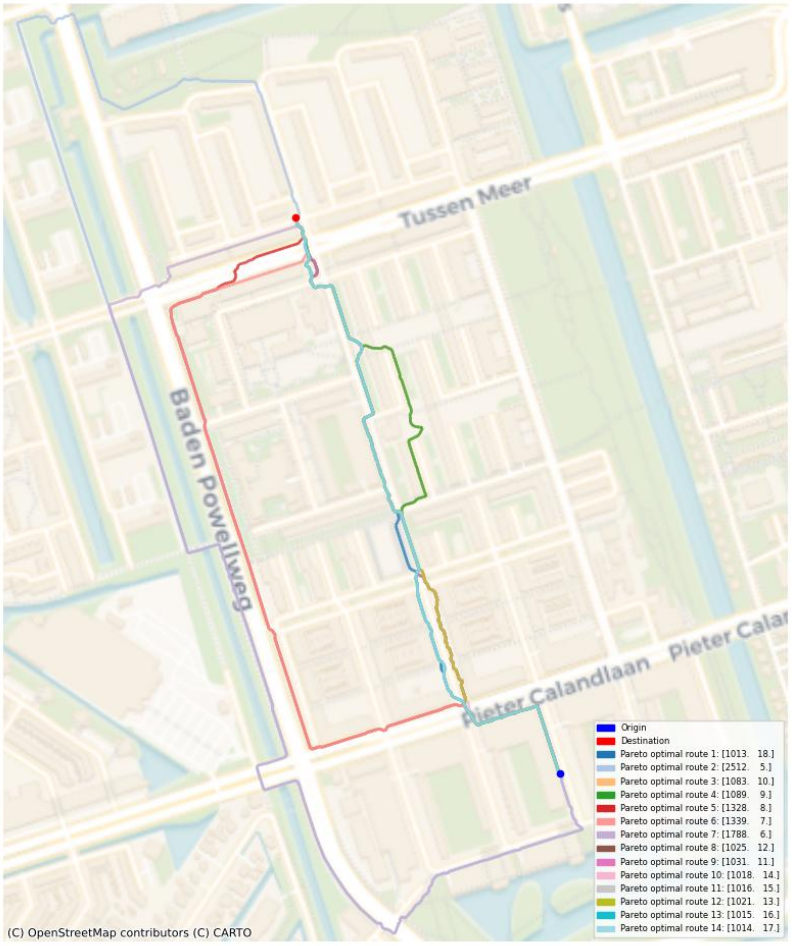


Figure 20 Routes in Pareto front for Osdorp-Midden route planning problem (far start/origin – two objective instance (distance, crossings)

3.2.5 Solution Realization and Monitoring Module

Since the next evaluation round with end users is planned to be conducted under lab conditions, the Solution Realization and Monitoring Module has not been a priority yet. In the future, this module is expected to monitor whether a user is following a recommended route or deviating from it, which could trigger additional user interactions – with the goal of either explaining to the user why this deviation will most likely lead to a suboptimal route, or understanding the reasons for the deviation (an obstacle, a more pleasant street etc.) and updating world and user models accordingly.

3.2.6 Feedback and Learning Module

As explained in Section 2.1.3 this module works by repeatedly asking the user for feedback and refining the search for better solutions based on those preferences. It starts with an initial solution (either the first Pareto-optimal solution or the most recently generated one) and asks the user which objective they would like to improve. Using this preference, the system runs its internal optimization process (IPRO) [6] to generate a new Pareto-optimal solution that better aligns with the selected objective. The user is then asked to compare this new solution

with the best solution they have preferred so far, choosing which one better matches their needs. Because user judgments may be uncertain, especially when solutions are similar, some variability or “noise” can affect these comparisons. After each comparison, the solution that the user prefers becomes the new reference point. This process continues iteratively, gradually learning the user’s preferences and guiding the search toward more desirable solutions. , , and show and explains this iterative process for a four-objective (path length, crossing, walk path length, and bike path length) optimization process. We refer the reader to our API Specification guidelines for more details on implementation[3].

[3] To be released on PEER Zenodo repository

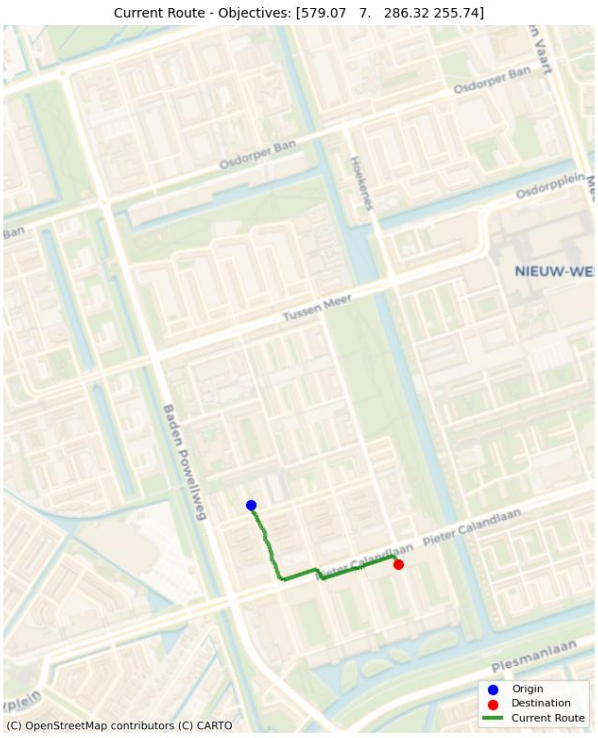


Figure 21 First solution (iteration) – optimized for Length objective

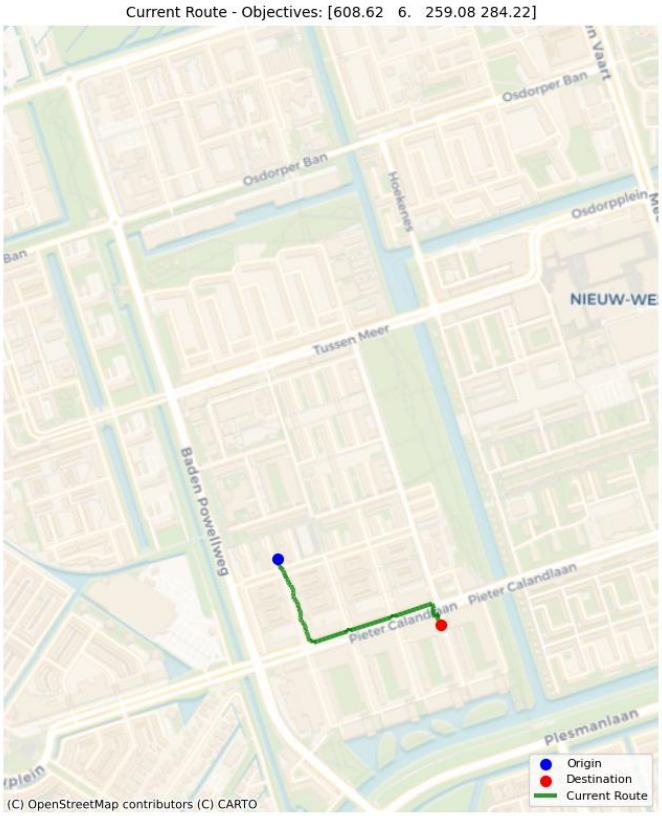


Figure 22 Second solution (iteration) – optimized for Length objective

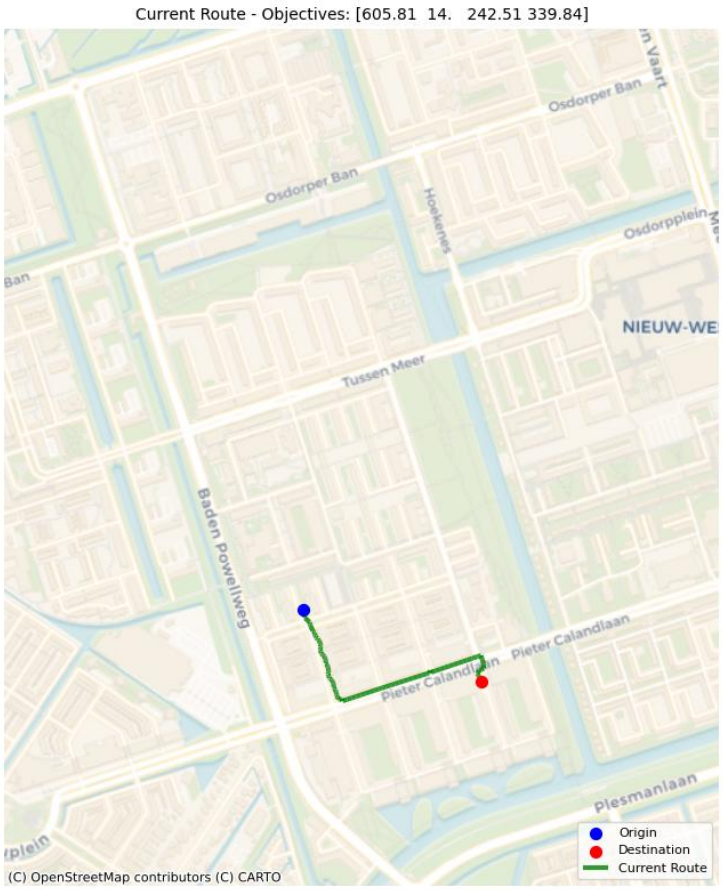


Figure 23 Third solution (iteration) – optimized for number of crossings

3.2.7 Conversation Management Module

The following user interface was delivered by AMS UI/UX designers in a Figma format. It describes the interaction design that PG-IPRO assumes for the current implementation of the route planner in the City of Amsterdam. With the collaboration of WP3 partners and use case discussions, one of the possible user interactions is presented in Figure 24. This visualization is based on the user providing a single-objective optimization preference for one of the objectives to be improved.

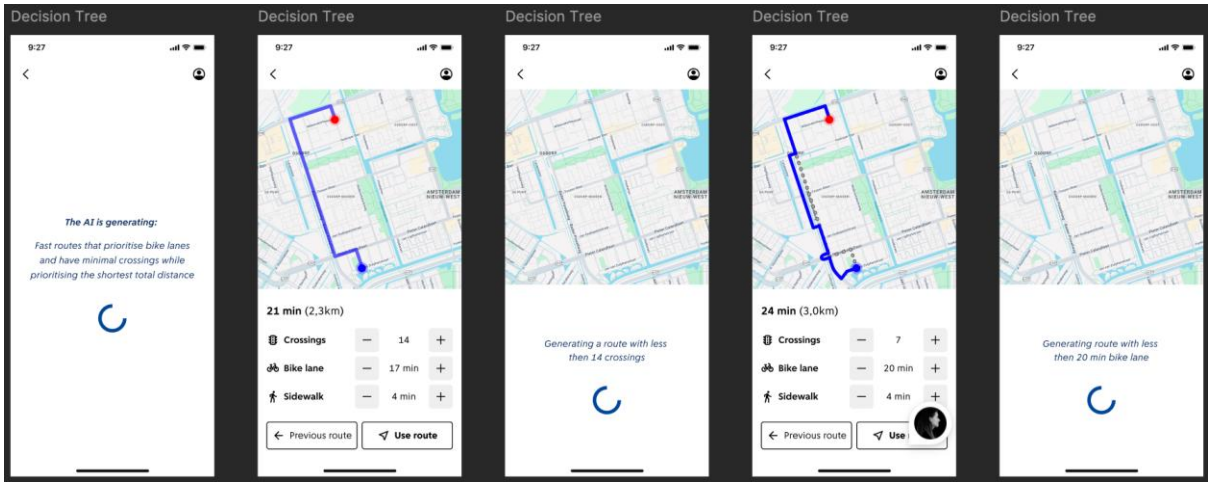


Figure 24 Designed user interface that PG-IPRO assumes available.

3.2.8 Next steps

In line with the project's plan for the third round of use case evaluations, D3.2 is delivered in the form of code implementations; we are making the step from Wizard of Oz style AI simulations to a running system that users – with or without support from the researchers conducting the evaluations - can interact with. However, not all PEER modules have been implemented yet, and also the functionality of the implemented modules in D3.2 leaves room for future work towards D3.3 in several aspects.

With regard to the routing itself, another line of research is the formulation of PG-IPRO with some prior assumptions about the distribution of possible utility functions. Incorporating such prior information could significantly improve PG-IPRO's sample efficiency and convergence speed, leading to scalability in higher-dimensional objective spaces.

In parallel, we are extending the routing work with an analysis of heterogeneous user preferences using Exceptional Model Mining (EMM) [28], [29]. Based on observed route-choice data and route attributes, EMM identifies interpretable subgroups testing whether feature effects differ significantly between contexts (e.g. short, medium or large distance on a route). The resulting context-specific effects can be integrated into the running system to enable more personalized and transparent route recommendations.

With regard to user-AI interactivity and route explanations, we will develop further ideas in collaboration with the municipality of Amsterdam and based on
the next round of evaluation.

As the intended user group is non-technical, explanations will have to be less in-depth than for example in the Datacation use case with its technically highly qualified users. There is further work to be done to understand and focus on the most relevant and most understandable routing aspects for the users of the envisioned app; we could, e.g., explore landmark-based vs. objective-based comparisons between routes, and will continue to hone in on the most intuitive interfaces.

As an example for an insight from our ongoing research: Section 2.2 describes the Counterfactual Routing Competition (CRC 2025) we ran at the IJCAI 2025 conference, which

asked participants to efficiently compute counterfactual explanations for the question: “Why is *this* [fact] route optimal, and not *that* [foil] route I expected?” However, while the competition generated great interest and an interesting variety of approaches, we found that not all produced explanations seemed immediately intuitive or satisfying from a human perspective. A user might ask, “Why not this [foil] route?” and expect an explanation related to problems with the foil route, such as obstacles they can perceive (narrow sidewalks or high curbs), or suboptimal trade-offs (such as a shorter total distance that is however not outweighing a longer stretch on a dispreferred bike path). They might intuitively expect an explanation saying, “Your foil route would be optimal, if it wasn't for this aspect you may have overlooked''. Instead, the participants’ returned counterfactual maps in some cases made the fact route impassable, forcing the router onto a route that happens to match the expected foil route. This type of explanation essentially says, “Your foil route would be optimal, if the fact route was artificially made worse in this specific way''. This could feel opaque or counterintuitive to a human observer. To provide counterfactual explanations that are more human-understandable and satisfactory, additional constraints capturing human expectations and interpretability will be required in the future.

3.3 Proditec. Human-in-the-loop and Explainable Anomaly Detection Trainer

3.3.1 Environmental Model

The environmental model as introduced in D3.1 consists of labelled datasets from the visual anomaly detection area. In this deliverable, we additionally extended this model to contain hyperparameters related to the model and dataset, which will further be optimized in Problem Solver component.

To evaluate the functionality and human-AI interaction pipeline, we evaluate our proposed methods on the MVTec Anomaly Detection (MVTec AD) dataset [12], a widely used benchmark for industrial anomaly detection tasks. The dataset comprises 15 sub-datasets representing diverse object and texture categories, each containing high-resolution images of both normal and anomalous samples. A comprehensive overview of the dataset is presented in Table 4. This dataset represents our environmental model for evaluating the algorithms proposed for this image anomaly detection. The proposed methods will be used on use case dataset, consisting images of Singapor coins during MVP#2 user studies.

Table 4 Description of MVTec Anomaly Detection dataset. Will be used during use case evaluation.

	Category	Train (normal)	Test normal	Test anomaly	Anomaly groups
Textures	Carpet	280	28	89	5
	Grid	264	21	57	5
	Leather	245	32	92	5
	Tile	230	33	84	5
	Wood	247	19	60	5
Objects	Bottle	209	20	63	3
	Cable	224	58	92	8
	Capsule	219	23	109	5
	Hazelnut	391	40	40	4
	Metal Nut	220	22	93	5
	Pill	267	26	141	7
	Screw	320	41	119	5
	Toothbrush	60	12	5	1
	Transistor	213	60	40	4
	Zipper	240	32	119	7
	Total	3629	467	1258	73

The set of hyperparameters and their constraints as a part of environmental model depends on the problem abstraction module selection. In most of the cases they are related to the hyperparameters of the machine learning model (such as number of epochs, backbone

architecture for image representation, etc.), and preprocessing steps (such as image size, normalization, cropping, labelling, etc.).

3.3.2 User Model

In contrast to MVP#1, where the evaluated user group appeared to have limited expertise with the task, our study assumes a more general user profile. Users are assumed to be capable of recognizing visual anomalies in generic coin defects. We also do not assume expertise in AI or machine learning but we assume that the goal of the user is to achieve high F1 score, balancing Precision and Recall. The primary focus is on human-AI interaction, including the insights gathered through user queries and the minimal supervision needed for data annotation.

Additionally, we formalize the notion of a user model that captures the characteristics and objectives of the interacting human. In our framework, the user model is represented as a set of parameters that describe the user's proficiency in anomaly detection algorithms. Currently we only support expert and non-expert profiles. Alongside these competency-related attributes, the model also includes several parameters reflecting the user's operational objectives, such as desired annotation quality, the amount of time the user intends to dedicate to data labeling, tolerance for labeling effort, and preferences regarding the trade-off between speed and accuracy. We refer to these user-specific parameters as soft parameters, in contrast to the model's hyperparameters, since they shape the human-in-the-loop optimisation process rather than the underlying machine learning model itself. These soft parameters enable the system to adapt its guidance, recommendations, and interaction patterns to align with the user's goals and constraints.

3.3.3 Problem Abstraction Module

This module encapsulates the technical processes involved in image anomaly detection, executing them entirely in the background. This includes model optimization, anomaly scoring, and data processing, all of which are automated and abstracted from the user as explained above. By handling these tasks internally, the module allows the study to concentrate on human-AI interaction, ensuring that the main objective—gathering user insights and supporting minimal supervision in data annotation—can be pursued without requiring knowledge of underlying anomaly detection algorithms.
Moreover, the abstraction module is responsible for formally defining the anomaly-detection task as a semi-supervised learning and hyperparameter-optimization problem. In practice, this means that the system operates on latent representations of input images and, under minimal user supervision, learns an appropriate decision threshold that separates normal from abnormal samples. From a mathematical perspective, the abstraction module encapsulates the optimization procedure that identifies this threshold while simultaneously tuning the necessary hyperparameters to ensure stable and effective anomaly scoring. By consolidating

these processes, the abstraction layer isolates the complexity of the learning formulation from the user, enabling intuitive interaction while maintaining rigorous ML-based anomaly detection underneath.

3.3.4 Problem Solver

In this deliverable, the Problem Solver is responsible for selecting the most suitable model and hyperparameter configuration for the anomaly-detection task. Originally, the module relied solely on a native set of hyperparameters tied to the technical structure of each model, such as backbone architecture, number of layers, or training epochs. We have now extended this set to include softer, higher-level parameters that reflect human decisions and data-preprocessing choices, including labelled-data size, presence of missing values, or execution-time constraints. At present, these parameters are evaluated within a simulated environment to study their impact on the learning process. As future work, they will be integrated into the conversational user interface, enabling users to interact with them directly and to take a more active role in the training procedure - even without specialized machine-learning expertise.

3.3.5 Solution Realization and Monitoring Module

In our approach, we employ a double-thresholding mechanism to enhance anomaly localization and improve robustness against overlapping score distributions [30]. In the first stage, a preliminary threshold is applied to identify and exclude regions that are confidently classified as normal. This filtering step effectively removes background regions and structures that are common to the majority of images, thereby reducing the influence of normal structures on the subsequent analysis.
In the second stage, the final decision threshold is computed using only the remaining candidate regions that exhibit potentially abnormal behavior.
This two-step procedure mitigates the challenges associated with overlapping distributions of normal and anomalous scores, which commonly occur when abnormal images share substantial structural similarities with normal samples.
Figure 25 illustrates this challenge using the MVTec AD Leather category as an example, where both normal and anomalous samples contain similar textures and patterns, leading to significant overlap in their score distributions when applying pure PatchCore approach.

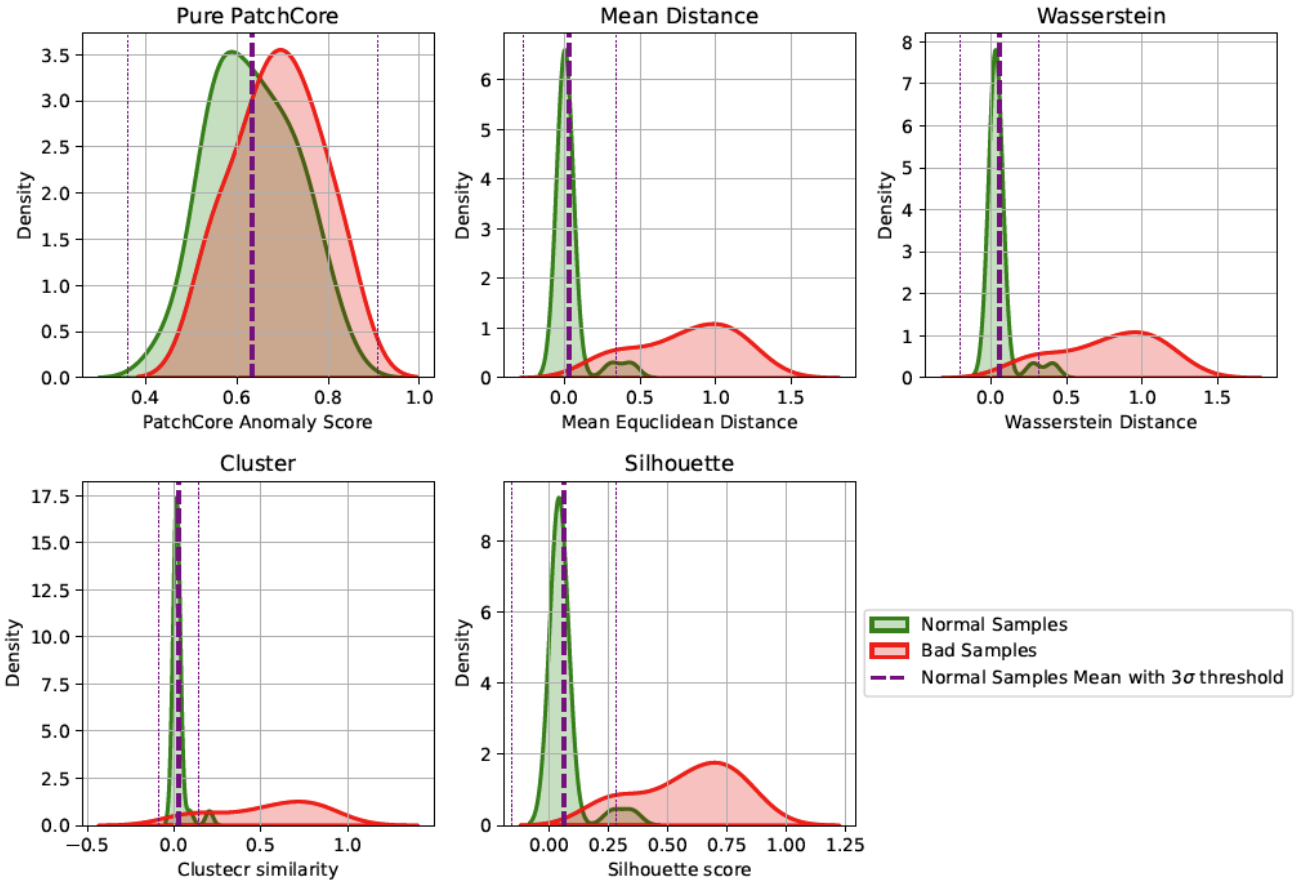


Figure 25 Example of overlapping distributions of normal and anomalous samples in the MVTec AD Leather category, illustrating the challenge of threshold selection in scenarios with structural similarities between normal and abnormal images in pure PatchCore approach vs Double-Thresholded and other custom methods.

On the other hand, this overlap is significantly reduced when using our double-thresholding method, resulting in a clearer separation between normal and abnormal samples. For PatchCore implementation we used the official code from the anomalib library[4].

For all of the methods and datasets we used *resnet18* backbone without any additional fine-tuning or augmentations. Here we summarize the methods in more detail.

The means method computes the per-image mean of patch-wise anomaly coefficients - defined as the dissimilarity of each patch from nominal patches - and flags anomalies using a threshold set classical 3-sigma, making it effective for detecting globally elevated anomaly evidence. The wasserstein method measures the Wasserstein distance between an image's distribution of these coefficients and the aggregated normal distribution; values exceeding the 3-sigma threshold indicate distributional shifts. The cluster method applies KMeans clustering to nominal embeddings and evaluates each image by comparing its cluster-membership distribution to the nominal reference via cosine similarity, where low similarity denotes anomalous patterns. Finally, the silhouette method computes a silhouette score over an image's patch-level cluster labels, using deviations in score magnitude to flag unusual structural configurations.

Moreover, the overall detection procedure operates in an iterative and sequential manner, where the output of each stage directly informs and refines the next. The preliminary filtering

[4] https://anomalib.readthedocs.io

step produces an initial separation of confidently normal and ambiguous regions, and this reduced subset becomes the sole input for the subsequent threshold-selection phase. This sequential pipeline may be repeated or refined based on the user's objectives—such as preferred balance between false positives and false negatives—until the resulting threshold and anomaly maps converge to a level that satisfies the user's expectations. In this way, the process forms a feedback-driven loop, progressively improving anomaly discrimination while incorporating user-defined performance criteria into the convergence condition.

3.3.6 Feedback and Learning Module

To integrate human expertise into the machine's learning process, we employ human-assisted refinement of the threshold estimation explained above on the ambiguous areas, allowing the method to achieve a clearer separation between normal and abnormal regions, improving the quality of anomaly detection.
Importantly, this step also serves as a feedback point for the broader human-AI workflow. Based on the insights gained from the current iteration, the user may decide to revisit the environment model or the user model to adjust or redefine their operational goals. This may include refining expectations regarding annotation quality, relaxing constraints on labeling time or effort, or enriching the environment model by providing additional data. Within this module, the system presents the results, explains the reasoning behind the current thresholding outcome, and highlights the uncertainties or remaining ambiguities. The user is then given the opportunity to restart the process from the beginning, incorporating the knowledge and experience accumulated during the current iteration. This establishes a cyclic, iterative improvement loop in which both the user and the system continually refine their understanding of the task until a satisfactory operating point is reached.

3.3.7 Conversation Management Module

This module impelements the user-AI conversation through pre-computed explanations using heatmaps as shown in Figure below. For this we use PatchCore which generates anomaly heatmaps by comparing local image features to a memory of normal patterns learned during training. Conceptually, it assumes that every small region (patch) of a normal image has a characteristic feature representation, so at test time it checks how "familiar" each patch looks: regions that resemble previously seen normal patches are considered normal, while those that differ are flagged as anomalies. Algorithmically, a pretrained convolutional neural network extracts spatial feature maps from normal training images, and each spatial location is treated as a feature vector (patch embedding). These embeddings are stored in a reduced memory bank via coreset subsampling. For a test image, the same feature extraction is performed, and for each patch embedding, the algorithm computes the distance to its nearest neighbor in the memory bank; this distance serves as the anomaly score. These scores are arranged according to their spatial positions to form a low-resolution anomaly map, which is then upsampled and optionally smoothed to produce the final heatmap highlighting anomalous regions as shown in Figure 26.



Figure 26 Example of heatmaps generated for user during conversation.

As explained above, in MVP#2, the explanation are pre-computed and mocked. However, our results as shown in Section 2.2.1.3 and Section 2.2.2.2 suggests that augmenting the interaction with both heatmaps and natural language explainers can enhance the user-AI conversation and further enhance the proposed semi-supervised method to actively query the user on uncertain samples during lablization process.

3.3.8 Next steps

The next stage of development will focus on bringing the system closer to real-world deployment and enabling richer human-in-the-loop interaction. First, we plan to construct a benchmark dataset tailored to the Proditec use case, allowing us to evaluate the complete pipeline with real PDT experts and customers. Such a dataset will complement the MVTec AD benchmark by capturing domain-specific characteristics, operational constraints, and defect types relevant to industrial partners.

Second, hyperparameter optimization HPO will transition from its current simulated role into an operational component of the pipeline. Instead of being abstracted away, hyperparameter optimization will become an interactive mechanism that users can influence directly. This will allow human-driven adjustments of both technical and soft parameters—such as data-quality conditions, execution-time constraints, or annotation-effort limits—enabling more meaningful control over the training process.

Finally, these enhancements will support a more complete human-AI collaborative loop, where domain experts can inspect model behavior, provide targeted feedback, refine model thresholds, and iteratively guide optimization. Together, these steps will prepare the system for evaluation in realistic industrial workflows and ensure that the final product aligns closely with end-user needs and expectations.

3.4 Datacation. AI-assisted warehouse configuration optimization

3.4.1 Environment Model

Two complementary environment models were developed to capture the problem of product allocation in a warehouse: the digital twin and the research simulator. While each model

serves a distinct purpose, they are designed to work together, and bridging the gap between them is a priority for future work.

The digital twin offers a high-fidelity emulation of the existing Warehouse Management System (WMS), maintaining a real-time link to physical operations. However, while the real warehouse spans thousands of slots, the digital twin utilizes a representative layout of 50 storage locations. A key distinguishing feature is the explicit modelling of both product and shelf dimensions, which prevents products from being assigned to physically infeasible locations. Location-type incompatibilities with unloaded products are also captured. As such, this model prioritises realism over optimisation focus.

The research simulator, by contrast, is designed to capture the sequential nature of the allocation problem and is built within a sequential reinforcement learning framework. Its primary strength lies in replicating the real-world dynamics of incoming products: it simulates their arrival over time and accounts for uncertainty about which products will arrive, when, and for how long they will remain in the warehouse. Although an expected duration of stay may be available for a given product, this is treated as uncertain rather than deterministic. The limitation of the research simulator is that it operates under unitary capacity constraints, meaning that storage locations are modelled solely by the number of products they can hold, without accounting for physical product dimensions such as height. This simplification is the primary gap between the two environment models, and addressing it is a key step towards full alignment with the digital twin.

The developed solver operates within this environment and is called sequentially at each time step, representing one day of operations. At each step, the solver receives the current warehouse layout, the expected departure times of stored products, and the anticipated incoming orders, along with their expected durations of stay. Using this information, it produces an optimised warehouse layout for the next *n* periods, where *n* corresponds to the solver's lookahead depth. However, only the layout for the immediately following period is applied; the simulation then advances, and the process repeats. This process is explained in detail in Section 3.4.4.1– MIP Solver.,

Solving a model with temporal lookahead introduces significant scalability challenges. Planned next steps therefore include developing strategies to reduce the problem size prior to solving – for example, by restricting reallocation consideration to a subset of products or limiting the search space to a defined zone within the warehouse. This is discussed further in Section 3.4.4.2 – Increasing Scalability.

The research simulator has proven valuable in enabling the development of a solver that moves beyond myopic, single-period assignments and instead optimises product allocation with a forward-looking perspective. The natural next step is to deploy the algorithms developed within the research simulator into the digital twin. Two modifications are required for this transition: first, the research solver must incorporate a physical capacity constraint,

ensuring that products are only assigned to locations compatible with their dimensions; second, the digital twin must be extended to compute distances between storage locations, generating a distance or travel-time matrix that the solver requires as input.

3.4.2 User Model

Within MVP2, the user model is implemented as a dynamic, interaction-driven preference component integrated with the optimization backend. Rather than representing operators through static profiles or predefined parameter presets, preferences are captured through natural-language interaction and translated into formally defined optimization inputs.

The current implementation is directly coupled to the MIP problem solver. Through natural-language input, operators can emphasize or de-emphasize specific components of the objective function, as well as adjust selected structural parameters of the optimization model. All user-provided input is interpreted by a locally deployed LLM-based reasoning agent, that operates within the Abstraction Module and is grounded in the current warehouse state via Digital Twin (DT) integration. The agent translates qualitative operational statements (e.g., “retrieval is critical during peak season” or “avoid too many re-allocations”) into quantitatively bounded solver parameters.

Specifically, user input is used to generate and tune the weights of the following objective function components, further detailed in the Problem Solver section below:

- Storage time
- Retrieval time
- Product reallocation time
- Virtual cost associated with changing location types

In addition to objective weights, the user model supports configuration of selected solver model parameters:

- Planning time horizon
- Maximum number of product reallocation movements
- Maximum number of physical location type changes

Preference capture is therefore implemented as a mapping from qualitative intent to numerical values. All generated weights are validated against predefined admissible ranges before being passed to the solver. This ensures that user influence remains meaningful while preserving model feasibility and mathematical guarantees.

The user model is interaction-driven rather than profile-driven. Preferences are inferred from explicit instructions (e.g., direct weight adjustments) and from contextual cues expressed during dialogue (e.g., emphasis on congestion reduction). Interaction history is maintained to support incremental refinement of configurations within a session.

While the current implementation does not rely on historical training data, it establishes a foundation for future human-in-the-loop learning. Logged interactions, agent tool invocation traces, accepted parameter suggestions, and override patterns may be leveraged in future iterations to refine the translation mechanism and improve alignment between system recommendations and operator intent.

3.4.3 Abstraction Module

In this warehouse optimization use case, a human–AI interpreter has been developed to operationalize the abstraction module within the PEER system architecture. The interpreter acts as a mediation layer between the human operator and the MIP-based Problem Solver. Its purpose is to process natural-language operator input, together with warehouse data and operational constraints, and to translate these inputs into validated solver parameters and weights. In addition, it serves as an intelligent access point to warehouse information.

The implementation is based on a locally deployed Qwen 32B language model operating within an agentic workflow. Based on the human's input, the model orchestrates structured reasoning and controlled tool invocations, including API calls to a Digital Twin (DT) of the warehouse for real-time grounding. All outputs must conform to predefined typed schemas before being finalized into JSON files exposed to the solver. This enforces consistency, traceability, and bounded generation, preventing unsupported responses. The response time varies by query and is approximately eight seconds on average, enabling practical interactive use.

The abstraction module implementation directly addresses feedback from preliminary operator evaluations, who reported limited access to detailed WMS status metadata. In the current implementation, warehouse DT data is retrieved and processed into structured, readable summaries (e.g., explicit slot counts per row, utilization percentages, storage availability by location type). This reduces cognitive load, improves situational awareness, and significantly accelerates access to operational information that previously required manual inspection Figure 27.



Figure 27 Example conversation flow including information retrieval and solver configuration

Beyond information retrieval, the interpreter supports collaborative solver configuration. Suggested parameter values (such as storage and retrieval penalties, reallocation limits, and lookahead planning horizon) are generated based on contextual factors provided by the operator and the DT, such as seasonal demand, labor constraints, and current warehouse occupancy. Rather than presenting abstract scores or opaque confidence metrics, each suggestion is accompanied by a generated rationale that highlights the operational considerations or contextual elements likely influencing the suggested parameter value Figure 28.

Episodic conversation history is maintained throughout the interaction, enabling incremental refinement of solver configurations. Operators may accept, reject, override, or further tune suggested parameters. This free-form conversational interaction addresses earlier concerns raised in MVP1 regarding limited immersion in the Wizard-of-Oz prototype. In the present implementation, genuine question-answer exchanges, persistent history, and detailed solution exploration are supported within a single continuous interaction thread.

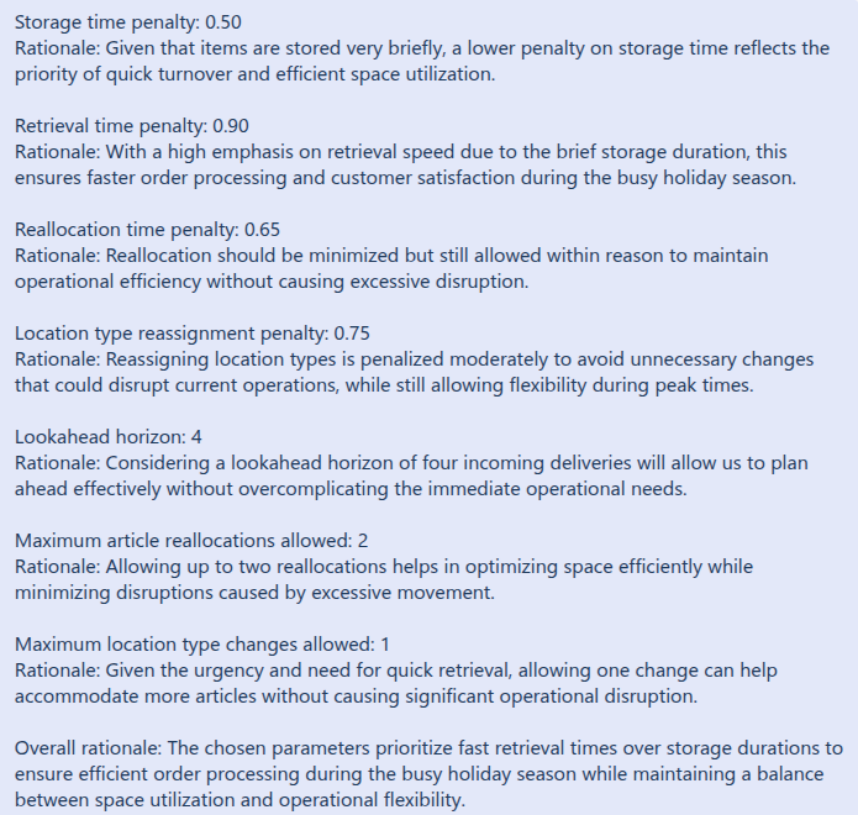


Figure 28 Example solver parameter configuration suggestions provided by AI agent based on conversation

Overall, the abstraction module transforms the optimization workflow from a static and abstract configuration procedure into an interactive, intuitive, grounded, and deliberative process. By combining natural-language access, structured validation, Digital Twin grounding, and collaborative parameter tuning, the system enhances usability and operator informed decision-making in industrial decision-support environments.

3.4.4 Problem Solver

Datacation currently uses a Warehouse Management System (WMS), referred to as the "industry solver", to allocate new orders as they arrive. However, when the warehouse reaches capacity, there may be insufficient space for incoming orders. In such cases, a manual reallocation procedure is conducted by human operators. This reallocation must be completed quickly enough to avoid blocking the storage process. To assist operators in this challenging and time-constrained task, we propose the development of an optimization solver for the reallocation procedure (referred to as the "scientific solver").

3.4.4.1 Scientific Solver

The developed scientific solver is non-myopic, meaning that it optimises decisions with respect to an objective function that accounts for the future impact of current choices. Rather than optimising solely for the present period, the solver plans the warehouse layout across the upcoming *n* time periods, where *n* defines the lookahead depth – that is, the number of periods considered when making each decision.

Each time the solver is called, it plans over this rolling horizon, but only the action corresponding to the next period is executed, as illustrated in Figure 29. This rolling-horizon approach ensures that every decision made today is informed by its projected consequences in future periods, avoiding the pitfalls of myopic planning. As the simulation advances to the next time period, new information is revealed – such as incoming products and updated expected durations of stay – and the solver is called again with this updated state.

The key advantage of this formulation over the WMS industry solver or a manual quick-fix approach is that it does not simply seek the best allocation for the current moment. Instead, it aims to make decisions that improve warehouse performance over time, resulting in more efficient and sustainable allocation strategies across the entire planning horizon.

Figure 29

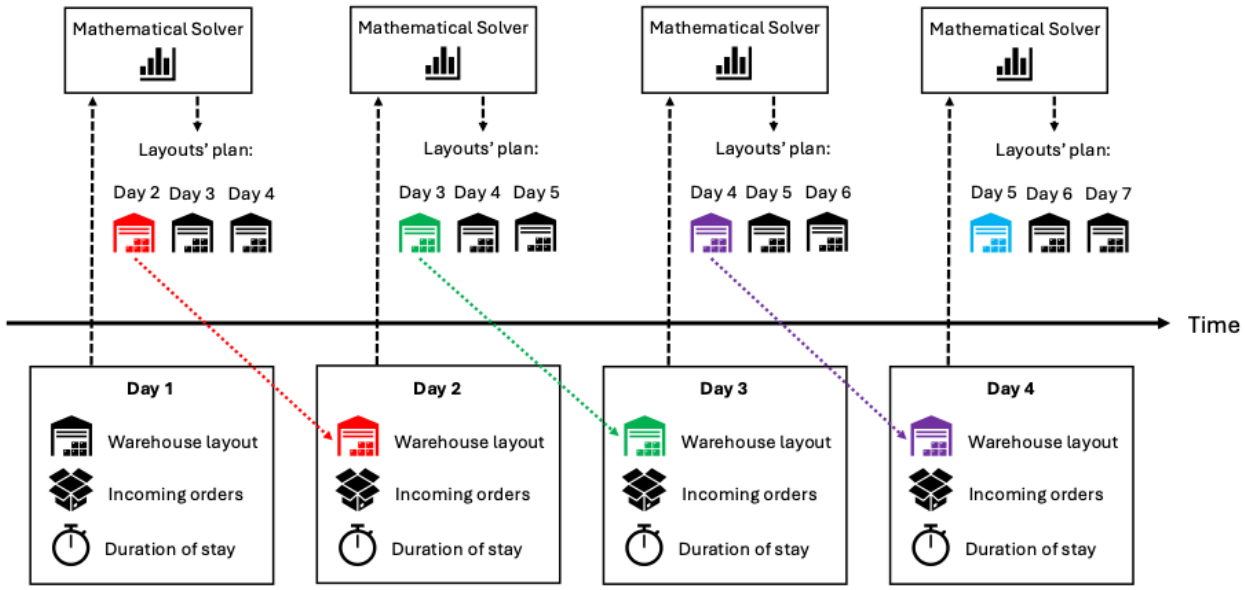


Figure 29 Rolling-horizon solver approach.

The developed Mixed-Integer Programming Model (MIP) contains decision variables representing the actions to be taken regarding the warehouse layout (such as where to allocate products, which products to reallocate, and which location types to change). The model aims to optimize a given objective function (minimizing total costs associated with storage, retrieval, and reallocation) while satisfying constraints that reflect the physical limits and product flow in the warehouse (such as capacity limits, product-location compatibility, and movement restrictions).

The developed optimization model considers three types of decision variables:

1. **Allocation of a product to a given location**: When a new product arrives, the system must decide where to store it.

2. **Reallocation of a product to a new location**: An already-stored product can be moved to a different location.
3. **Reallocation of a storage location type**: Each product can be assigned only to a compatible location type. It is possible to perform the administrative task of changing a location's type designation.

The objective function consists of four main components:

- **Storage time/distance**: Cost of storing new products in a warehouse location.
- **Retrieval time/distance**: Cost of retrieving a product for final delivery. Without reallocation, storage and retrieval times match; however, reallocation actions may cause them to differ.
- **Product reallocation time/distance**: Cost associated with moving products between locations.
- **Virtual cost of changing location type**: Administrative cost for changing location type designations.

Several model parameters can be adjusted:

- **Time horizon (number of periods to look ahead)**: A longer time horizon considers the arrival of more products over time, while a shorter horizon focuses on a more myopic, near-present allocation. If only the current day is considered, the model returns the most straightforward way to store products, without accounting for future incoming orders.
- **Maximum number of product reallocation movements** and **location type changes** per time period.

Figure 30 provides an example of an action plan for a specific moment for an illustrative small warehouse example. The colors and top-right labels represent location types, while the square, circular, and triangular shapes represent product types. In this example, circular products can be stored in location type B1 (purple), square products in location type C2 (blue), and triangular products in location type F3 (yellow).

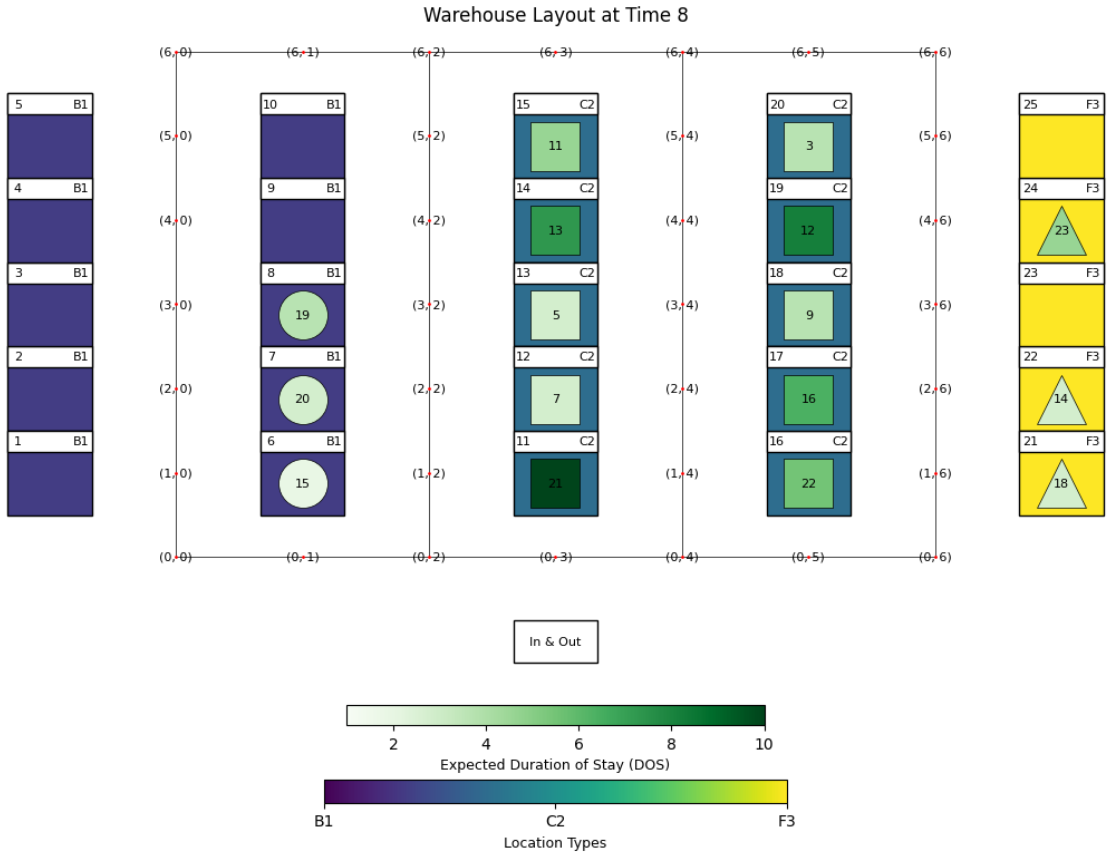


Figure 30 Solution example.

Currently, the model focuses on minimizing a personalizable cost function with tunable constraints and parameters. However, all its elements entail real costs (allocation, retrieval, reallocation, and changes in location type). One possibility to improve performance over time and handle real-world uncertainty is to incorporate new constraints or objectives. For example:

- **Reserving space for anticipated large orders**: Privilege solutions that maintain large contiguous blocks of empty space for expected bulk orders from the same customer
- **Maintaining buffer zones**: Keep available space at the warehouse entry to ensure effective storage and retrieval of incoming high-rotation products

3.4.4.2 Increasing scalability

The current formulation has been applied to small examples representing only part of the full warehouse. When applied to the complete warehouse, the number of possible decisions (variables) increases rapidly, potentially preventing the system from providing solutions in a useful timeframe.

One approach we have begun investigating is to use a learning algorithm to restrict the solution search space at each decision point by focusing only on a smaller, more promising part of the warehouse. The idea is to leverage machine learning to identify promising locations

for product allocation, thereby enabling the optimization model to run faster with fewer decision variables, as illustrated in Figure 31.

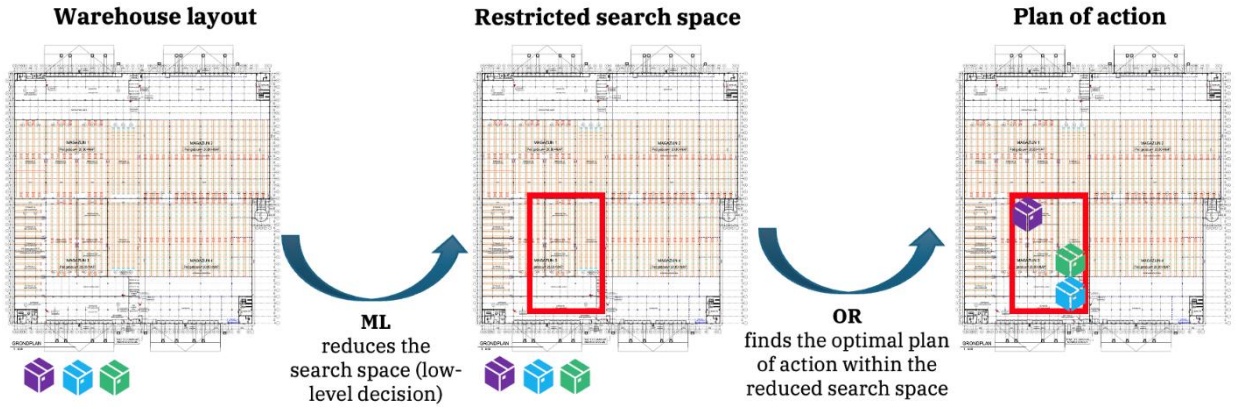


Figure 31 Connecting learning algorithms with classical optimization.

3.4.5 Solution Realization and Monitoring Module

The solver is designed to propose a warehouse layout that ensures strong performance over time rather than optimizing only for the present moment. To achieve this, it incorporates a lookahead perspective, accounting for expected incoming products and their anticipated duration of stay. By anticipating future flows, the model generates a sequence of layouts that remain efficient over a given planning horizon.

However, real operations are subject to uncertainty. Unexpected high-priority deliveries, delays, or orders that take significantly more or less time than forecast may require updating the proposed plan. For this reason, an important future development is implementing a solution-realization layer that compares the expected sequence of layouts with the actual sequence resulting from product arrival and storage variability. Such a mechanism would allow operators to understand the rationale behind the originally proposed layout – based on expected events – while clearly identifying when deviations arise from uncertainty and when replanning becomes necessary. This integration remains part of future work.

3.4.6 Conversation Management Module

Within the Datacation use case, conversation management is realized through an LLM-based agent that structures and coordinates the interaction between the user and the optimization backend. The LLM-based agent acts as an interaction orchestrator, managing turn-taking, interpreting follow-up questions in context, and ensuring that each conversational step is linked to the current optimization state. An agentic workflow is employed to manage a continuous dialogue, integrating structured reasoning, tool-mediated grounding, persistent interaction history, and schema-compliant output generation with retry mechanisms.

Rather than treating each request as independent, the agent maintains a persistent interaction history within a session. This allows it to reference previous configurations, compare alternative solutions, and support incremental refinements. For example, an operator can

request adjustments relative to a previously generated configuration without restating all underlying assumptions.

A central function of the conversation layer is abstraction control. Operators who are likely unfamiliar with the mathematical formalism of optimization models, interact at the level of operational goals and trade-offs, while the solver operates on formal mathematical representations. This integration allows us to extract information that complements the solver with human decision-maker knowledge (See Section 3.4.3):

- **Tune objective function parameters**: If the decision-maker wants to minimize location type changes, they can increase the penalty for such changes in the objective function. Similarly, if they are comfortable with product reallocations, they can assign a lower cost to this action.
- **Modify decision variables**: Remove them to reduce the model size. For example, the human operator may guide the solver to consider allocation only in a specific part of the warehouse, reducing the model size by leveraging domain knowledge about the desired plan of action.
- **Adjust model parameters**: If the operator knows that upcoming days will be particularly challenging, they may opt for better reallocation with a longer time horizon, accounting for future orders. Conversely, if there is a high staff shortage and the focus is on minimizing immediate allocation costs, they may choose a shorter time horizon, leading to more myopic decision-making.

Moreover, the agent ensures that discussion remains at a level that is meaningful for the user, presenting results in terms of warehouse-relevant metrics (e.g., congestion levels, picking efficiency, storage utilization) rather than internal model variables. This supports clarity and reduces the expertise required to use advanced optimization tools.

Another important aspect is the integration of interpretive rationales into the dialogue. Users can ask for clarification regarding trade-offs or contextual factors influencing a suggested configuration. The agent manages these interpretation requests as part of the ongoing interaction loop, linking them directly to solver outputs and warehouse data.

3.4.7 Next Steps

Two concrete, already-planned next steps have been identified for the solver. The first concerns integration with the digital twin. The solver is currently integrated with the research simulator, and a natural next step is to extend this integration to the digital twin. This requires two changes: the digital twin will provide a distance/time matrix as input to the solver, and the solver formulation will be extended to include more complex capacity constraints that enforce feasibility based on product dimensions – a feature already implemented in the digital twin but not yet reflected in the solver.
The second next step is to improve scalability in preparation for larger, more realistic scenarios involving a greater number of storage locations and incoming products. As instance

size grows, so does model complexity, and providing solutions within a practically useful timeframe becomes increasingly challenging. The planned approach is to leverage learning algorithms to intelligently guide the search for a solution – for example, by identifying which regions of the warehouse are more promising for storage and which products are the most likely candidates for reallocation. By reducing and focusing the search space in this way, the lookahead solver can deliver meaningful plans within operationally acceptable time limits.

An extension of the current abstraction module reasoning agent is to enable the generation and comparison of multiple solutions derived from different parameter configuration sets. In the current implementation, the system proposes a single parameter set configuration accompanied by contextual rationales, and this set of parameters may be continuously tuned throughout the active session. Future work can expand this capability to allow the agent to construct several alternative parameter configuration sets that all satisfy the operator's intent, execute the solver under each configuration, and compare the resulting solutions. This would enable explicit what-if analysis within the conversational interface. The agent could automatically construct alternative configurations that satisfy the operator's intent and that reflect plausible trade-offs. For example, one configuration may prioritize retrieval efficiency, while another may emphasize storage time minimization or reduced reallocation frequency. The solver would then be executed under each configuration by the reasoning agent, and the resulting differences in key operational indicators (e.g., retrieval time, congestion levels, number of reallocations, storage layout) would be summarized and presented comparatively.

Importantly, this extension would make the solver not only a downstream execution component but also an actively invoked deliberation tool within the agentic workflow. The agent would be able to call the solver iteratively to generate counterfactual outcomes, analyze the resulting performance metrics, and integrate them into the ongoing dialogue. In this manner, a closed feedback loop is established: user intent informs parameter generation, the agent explores several configurations, the solver produces measurable consequences, and those consequences are fed back into the reasoning process to support refined deliberation. Optimization thus becomes not only a backend computation step, but a structured exploration mechanism embedded directly within the human-AI interaction loop.

Abstraction module – Adaptive personalization: cross-episode preference learning

Another extension of the abstraction module is the integration of longitudinal preference learning mechanisms. Beyond maintaining episodic memory within a single interaction, the system could learn from operator decisions across past interaction episodes, identifying recurring adjustment patterns and consistent trade-off preferences (e.g., systematic prioritization of reduced reallocations over retrieval speed). These learned behavioral tendencies could inform future parameter suggestions, enabling progressively more personalized, context-aware, and adaptive configuration support over time.

The learning process would be implemented through an explicit memory system supported by external LLM-based agents. These agents would analyze historical interaction episodes, extract structured representations of operator preferences and decision tendencies, and store them in dedicated long-term memory components. The abstraction agent could then retrieve these stored preference summaries during subsequent sessions, allowing them to inform parameter generation and solver configuration suggestions. In this way, personalization would not rely solely on short-term conversational context, but on accumulated cross-episode knowledge that systematically influences future outputs.

Lessons Learned

Sonae

The transition from a controlled simulator environment to a real in-store system highlighted the critical importance of well-defined triggers. While the simulator relied on simple arrow-based navigation, real-world deployment required dynamic responses to customer behavior. In particular, distinguishing between triggers that update customer location and those that rerun routing and recommendation algorithms proved essential. Events such as route deviations or changes to the shopping list directly affect both recommendation stability and computational effort. Poorly calibrated triggers can lead either to erratic recommendation changes or to outdated suggestions, making balance a key design consideration. Additionally, the continuously evolving product catalog underscored the need for algorithms to operate on updated data, requiring periodic retraining or tuning to maintain recommendation quality.

Datacation

Within the research activities, work on abstract warehouse modules demonstrated that lookahead decision-making is a major performance driver. Compared to myopic allocation strategies, forward-looking models better handle uncertainty regarding incoming products and their duration of stay, leading to improved expected performance. However, these gains come with higher computational costs. Ensuring real-time applicability, therefore, requires reducing the search space through learning mechanisms and intelligent heuristics, balancing solution quality with computational feasibility.

City of Amsterdam

One of the key lessons learned from this use case in the City of Amsterdam is that the current dataset features such as walk path width and static obstacles have not yet been fully validated. This validation is essential before progressing to MVP#2 and should be carried out in parallel with the algorithmic evaluation. Another important insight is related to user expectations: several users indicated a preference for a fully mobile application, whereas the current implementation is primarily web-based. While this does not impact the technical evaluation of the solution, it is a valuable consideration for future deliverables, as aligning the format of the application with user expectations will be key to ensuring adoption and user satisfaction.

Proditec

A key lesson learned from this use case is the importance of ensuring strong alignment between the intended end users and the assumptions made by researchers during algorithmic development. The WP4 evaluation results highlighted that the actual participants involved in the evaluation did not always fully match the profiles or expertise anticipated during WP3 developments in MVP#1. This mismatch can significantly affect both performance outcomes and user satisfaction. In addition, certain use cases, such as PDT, face constraints in accessing

cloud services (e.g., Amazon, for test and deployment), which directly impacts the feasibility of computationally intensive algorithms. As a result, computational requirements must be carefully considered during development, and infrastructure limitations should be clearly defined and agreed upon at the very beginning of the project. Early clarification of these constraints is critical for ensuring realistic design choices and ultimately achieving successful implementation.

Deviations occurred and corrective actions taken

None.

Conclusions

In conclusion, WP3 has made significant progress toward achieving its objectives, as reflected in this D3.2 report, which provides a comprehensive overview of the ongoing research and its application across the four use cases. The work has advanced key areas such as AI explainability, user modelling, and AI–user interaction, while also addressing domain-specific challenges including anomaly detection and multi-objective routing.

Moreover, the practical development of the proposed PEER agent architecture marks an important milestone. Building on the previous MVP presented in D3.1—delivered as Figma designs enabling Wizard-of-Oz style AI simulations—this second release (MVP2) represents the first functional code implementation of the PEER agents. This implementation enables direct user interaction in the upcoming evaluation phase. The report has outlined the algorithms integrated into the agents, including initial Problem Solver modules based on both novel approaches, such as multi-objective pathfinding, and established methods like mixed integer programming, as well as the first prototypes for XAI and user feedback mechanisms across domains. Finally, the next steps focus on advancing the systems toward real-world deployment, strengthening human-in-the-loop capabilities, further developing the Solution Realization and Monitoring modules, and leveraging abstraction techniques to scale the underlying solvers effectively.

References

[1] M. Asadi *et al.*, “PEER D3.1 v2 - MVP of the PEER collaborative AI assistant for lab evaluation,” Sep. 2025, *Zenodo*. doi: 10.5281/zenodo.17206411.

[2] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, “Towards total recall in industrial anomaly detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 14318–14328.

[3] H. Zhang *et al.*, “Superad: A training-free anomaly classification and segmentation method for cvpr 2025 vand 3.0 workshop challenge track 1: Adapt & detect,” *arXiv preprint arXiv:2505.19750*, 2025.

[4] G. Peng, E. Pauwels, and H. Baier, “Multi-Objective Utility Actor Critic with Utility Critic for Nonlinear Utility Function,” in *Eighteenth European Workshop on Reinforcement Learning*, 2025.

[5] A. De Greef, “Adaptive route planning with human-AI collaboration: A preference-driven approach to multi-objective decision-making,” Vrije Universiteit Brussel, 2025.

[6] W. Röpke, M. Reymond, P. Mannion, D. M. Roijers, A. Nowé, and R. Rădulescu, “Divide and conquer: Provably unveiling the pareto front with multi-objective reinforcement learning,” *arXiv preprint arXiv:2402.07182*, 2024.

[7] M. Bello, R. Bello, M.-M. Garc\'\ia, A. Nowé, I. Sevillano-Garc\'\ia, and F. Herrera, “A three-Level Framework for LLM-Enhanced eXplainable AI: From technical explanations to natural language,” *Information Systems Frontiers*, pp. 1–22, 2025.

[8] Y. Wang and K. Li, “Large Language Models in Operations Research: Methods, Applications, and Challenges,” *arXiv preprint arXiv:2509.18180*, 2025.

[9] A. AhmadiTeshnizi, W. Gao, and M. Udell, “Optimus: Scalable optimization modeling with (mi) lp solvers and large language models,” *arXiv preprint arXiv:2402.10172*, 2024.

[10] M. Turpin, J. Michael, E. Perez, and S. Bowman, “Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting,” *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 74952–74965, 2023.

[11] Z. Zhang, J. Ruan, X. Gao, T. Liu, and Y. Fu, “Eiad: Explainable industrial anomaly detection via multi-modal large language models,” in *2025 IEEE International Conference on Multimedia and Expo (ICME)*, 2025, pp. 1–6.

[12] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, “The MVTec anomaly detection dataset: a comprehensive real-world dataset for unsupervised anomaly detection,” *Int. J. Comput. Vis.*, vol. 129, no. 4, pp. 1038–1059, 2021.

[13] S. Akcay, D. Ameln, A. Vaidya, B. Lakshmanan, N. Ahuja, and U. Genc, “Anomalib: A deep learning library for anomaly detection,” in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 1706–1710.

[14] H. Baier, M. T. Keane, S. Sreedharan, S. Tulli, and A. Verma, “Explanations for Sequential Decision-Making–an Overview,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026, pp. 40948–40953.

[15] M. Srivastava, C. Colas, D. Sadigh, and J. Andreas, “Policy learning with a language bottleneck,” *arXiv preprint arXiv:2405.04118*, 2024.

[16] M. Tec, G. Xiong, H. Wang, F. Dominici, and M. Tambe, “Rule-Bottleneck Reinforcement Learning: Joint Explanation and Decision Optimization for Resource Allocation with Language Agents,” *arXiv preprint arXiv:2502.10732*, 2025.

[17] R. O. Moraes, Q. A. Sadmine, H. Baier, and L. H. S. Lelis, “Innatecoder: Learning programmatic options with foundation models,” *arXiv preprint arXiv:2505.12508*, 2025.

[18] S. Izquierdo-Badiola, C. Rizzo, and G. Alenyà, “Raider: Tool-equipped large language model agent for robotic action issue detection, explanation and recovery,” *arXiv preprint arXiv:2503.17703*, 2025.

[19] L. Wang *et al.*, “A survey on large language model based autonomous agents,” *Front. Comput. Sci.*, vol. 18, no. 6, p. 186345, 2024.

[20] S. Yao *et al.*, “React: Synergizing reasoning and acting in language models,” in *The eleventh international conference on learning representations*, 2022.

[21] Z. An *et al.*, “LogiEx: Integrating Formal Logic and LLMs for Explainable Transit Planning,” in *Proceedings of the 17th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS 2026)*, 2026.

[22] Z. An *et al.*, “Combining LLMs with logic-based framework to explain MCTS,” *arXiv preprint arXiv:2505.00610*, 2025.

[23] N. Wojak-Strzelecka *et al.*, “Explainable Visual Anomaly Detection with Multimodal Models and Metadata-Augmented Prompts,” in *Machine Learning and Knowledge Discovery in Databases: ECML PKDD 2025 Workshops (IXAIT)*, to be appeared in Lecture Notes in Computer Science.

[24] Roman Bartak et al., “A Modular Architecture for Collaborative AI Companions.” *Work in Progress, Under Review*.

[25] M. Ariannezhad, S. Jullien, M. Li, M. Fang, S. Schelter, and M. De Rijke, “ReCANet: A repeat consumption-aware neural network for next basket recommendation in grocery shopping,” in *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 1240–1250.

[26] O. Lashinin, D. Krasilnikov, A. Milogradskii, and M. Ananyeva, “Saferec: Self-attention and frequency enriched model for next basket recommendation,” in *European Conference on Information Retrieval*, 2025, pp. 294–302.

[27] F. Fazaeefar et al., “Personalized Next Basket Recommendation.” In 26th *International Conference on Computational Science (ICCS2026) Workshops (SmartSys’26), to be appeared in* in Lecture Notes in Computer Science.

[28] W. Duivesteijn, A. J. Feelders, and A. Knobbe, “Exceptional Model Mining: Supervised descriptive local pattern mining with complex target concepts,” *Data Min. Knowl. Discov.*, vol. 30, no. 1, pp. 47–98, 2016.

[29] W. Duivesteijn, A. Feelders, and A. Knobbe, “Different slopes for different folks: mining for exceptional regression models with cook’s distance,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 868–876.

[30] S. Bobek et al., “Combining unsupervised representation learning with Human-Guided Refinements to Reduce Labeling Effort in industrial anomaly detection tasks.”, *Work in Progress*

[31] Heider, F., & Simmel, M. (1944) An experimental study in apparent behavior. The American Journal of Psychology, 57, 243-259.

PEER will focus on how to systematically put the user at the centre of the entire AI design, development, deployment, and evaluation pipeline, allowing for truly mixed human-AI initiatives on complex sequential decision-making problems. The central idea is to enable a two-way communication flow with enhanced feedback loops between users and AI, leading to improved human-AI collaboration, mutual learning and reasoning, and thus increased user trust and acceptance. As an interdisciplinary project between social sciences and artificial intelligence, PEER will facilitate novel ways of engagement by end-users with AI in the design phase; will create novel AI planning methods for sequential settings which support bidirectional conversation and collaboration between users and AI; will develop an AI acceptance index for the evaluation of AI systems from a human-centric perspective; and will conduct an integration and evaluation of these novel approaches in several real-world use cases.





This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101120406.