Deliverable 3.1

MVP of the PEER collaborative AI assistant for lab evaluation

Version	2.0
Lead partner	CU
Dissemination level	Public
Type of deliverable	Demonstrator
Contractual date	31/03/2025
Actual delivery date	31/03/2025

History of changes:

Version	Date	Changes	Pages concerned
v1.0	Submitted on 31.03.2025	First draft, excluding MVP for DAT usecase	n/a
v2.0	Submitted on 30.06.2025	Included MVP for DAT usecase	Section 5.4, Section 6

D3.1. MVP of the PEER collaborative AI assistant for lab evaluation

Deliverable description as per DoA: Minimal viable product of planning algorithms, XAI and user modelling, and human-AI conversation loop for lab evaluation.

(Report + Prototype)

Authors	**Organisations**
Mehrdad ASADI	VUB
Paolo SPEZIALI	VUB
Diederik M. ROIJERS	VUB
Ann NOWE	VUB
Hendrik BAIER	TU/e
Grzegorz J. NALEPA	JU
Szymon BOBEK	JU
Krzysztof KUTT	JU
Roman BARTÁK	CU
Contributors	
Adriana Maria BASBOUS MOUKARZEL	EURECAT
Silvia IZQUIERDO-BADIOLA	EURECAT
Gennaro SCARATI	EURECAT
Fábio NEVES-MOREIRA	INESCTEC
Miguel LUNET	INESCTEC
Rafael ARRAIS	INESCTEC
Revised by	
Hendrik BAIER	TU/e
Adriana Maria BASBOUS MOUKARZEL	EURECAT
Silvia IZQUIERDO-BADIOLA	EURECAT
Gennaro SCARATI	EURECAT

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	**9**
1.1	Objectives of Work Package 3	9
1.2	Deliverable Structure	10
2.	Process Overview	**11**
2.1	Stakeholder Engagement	11
2.2	Scientific Literature Review	12
2.3	Onboarding of a New Use Case	12
3.	State-of-the-art and Work Performed	**13**
3.1	Planning and Search Algorithms for Human-AI Collaboration (Task 3.1)	13
3.1.1	Introduction to Sequential Decision Making	13
3.1.2	Decision and Utility Theory	14
3.1.2.1	The Role of Decision Maker Preferences	15
3.1.3	Planning and Search Problem	15
3.1.3.1	Search Algorithms	16
3.1.3.2	Reinforcement Learning for Policy Optimization	19
3.1.3.3	Making Search More Flexible and Adaptive	20
3.1.3.4	Planning Algorithms	22
3.1.3.5	Planning and Learning in Multi-Objective Environments	23
3.1.3.6	Learning in Multi-Agent, Multi-Objective Environments	24
3.1.3.7	Nature-inspired Optimization for Search and Planning	25
3.1.3.8	User-centric Planning and Search	26
3.2	Bidirectional Information Flow Between Users and AI (Task 3.2)	29
3.2.1	Information Flow from AI to User (explainable AI)	29
3.2.1.1	Prototypical and concept-based explanations	30
3.2.1.2	Comprehensibility of Explanations	31
3.2.1.3	Knowledge Mediation in Explanation Process	32
3.2.1.4	Towards Explainable Sequential Decision Making	33
3.2.1.5	Explaining Actions of Search Algorithms	35
3.2.1.6	Explaining Actions of RL Agents	39
3.2.1.7	Explaining entire policies of RL agents	39
3.2.1.8	Transparent Hyperparameter Tunning	40
3.2.2	Information Flow from User to AI (user modelling and feedback)	41
3.2.2.1	Recommender systems	41
3.2.2.2	Human-in-The-Loop Learning	45
3.3	User - AI Conversation Management (Task 3.3)	48
3.3.1	Human-in-The-Loop Approaches	48
3.3.2	Conversation Management	49
3.3.2.1	Key Aspects of AI Conversation Management	49

3.3.3 Computational Argumentation 51
3.3.4 Knowledge Granules and Levels of Abstraction in Conversational Management Systems 52

4. PEER Building Blocks for Collaborative and Adaptive Decision System 54
4.1 PEER Modular Architecture – Overview 55
4.2 Environment Model 56
4.3 User Model 57
4.4 Problem Abstraction Module 57
4.5 Problem Solver 57
4.6 Solution Realization and Monitoring Module 58
4.7 Feedback and Learning Module 58
4.8 Conversation Management Module 58

5. Realization of PEER Modules in Use Cases 59
5.1 Sonae. A Personalized In-Store Navigation Assistant for Customers (SigaApp) 59
5.1.1 Environment Model 60
5.1.2 User Model 61
5.1.3 Problem Abstraction Module 61
5.1.4 Problem Solver 62
5.1.5 Solution Realization and Monitoring Module 62
5.1.6 Feedback and Learning Module 63
5.1.7 Conversation Management Module 63
5.1.8 Next steps 64
5.2 City of Amsterdam. A Personalized Navigation App for Wheelchair Users 64
5.2.1 Environment Model 65
5.2.2 User Model 66
5.2.3 Problem Abstraction Module 66
5.2.4 Problem Solver 67
5.2.5 Solution Realization and Monitoring Module 67
5.2.6 Feedback and Learning Module 68
5.2.7 Conversation Management Module 70
5.2.8 Next steps 70
5.3 Proditec. Human-in-the-loop and Explainable Anomaly Detection Trainer 72
5.3.1 Environmental Model 72
5.3.2 User Model 73
5.3.3 Problem Abstraction Module 73
5.3.4 Problem Solver 73
5.3.5 Solution Realization and Monitoring Module 74
5.3.6 Feedback and Learning Module 74
5.3.7 Conversation Management Module 74
5.3.8 Next steps 75
5.4 Datacation. AI-assisted warehouse configuration optimization 75
5.4.1 Environment Model 76
5.4.2 User Model 76
5.4.3 Abstraction Module 77

5.4.4 Problem Solver 77
5.4.5 Solution Realization and Monitoring Module 77
5.4.6 Conversation Management Module 78
5.4.7 Next Steps 78
6. Lessons Learned **79**
7. Deviations occurred and corrective actions taken **81**
Conclusions **82**
References **84**

Table of Figures

Figure 1. WP3 Gantt Chart 10
Figure 2. Symbolic Representation of MDP[4] 14
Figure 3: Outline of Monte-Carlo Tree Search [23] 18
Figure 4: Partial Taxonomy of Modern RL Algorithms [27] 20
Figure 5. Overview of the libraries related to MOMAland within the Farama Foundation 25
Figure 6. Collaborative Human-Robot Planning Architecture 27
Figure 7. Explaining search with Computation Tree Logic (CTL) 37
Figure 8. Different Active Learning strategies. Top left: initial decision boundary of some ML algorithm. Top right: Uncertainty sampling. Bottom left: Diversity sampling. Bottom right: combination of diversity sampling and uncertainty sampling strategies [145] 47
Figure 9. Architecture of AI agent [1] 54
Figure 10. Architecture of utility-based agent 55
Figure 11. Architecture of general learning agent 55
Figure 12. Architecture of the PEER Companion Agent. 56
Figure 13. PEER building blocks mapping to personalized in-door shopping assistant use case 60
Figure 14. A screenshot of simulator and the interface of envisioned PEER shopping assistance (Sonae use case) 60
Figure 15. A screenshot of explicit user feedback form by which AI will learn to adapt for user needs 63
Figure 16. PEER building blocks mapping to explainable and personalized and route planner for wheelchair users 65
Figure 17. An example of an explanation of the differences between two routes in terms of objectives the user cares about (from the MVP). 68
Figure 18. An example of explicit user feedback on user preferences, accepted before the trip (from the MVP). 69
Figure 19. An example of explicit user feedback on solution quality, accepted after the trip (from the MVP). 69
Figure 20. PEER building blocks mapping to explainable anomaly detection use case 72
Figure 21. PEER building blocks mapping to smart warehouse configuration management 76
Figure 22. Possible abstract model for warehouse configuration management using network flows 77

List of Abbreviations

Abbreviation	Definition
AC	Algorithm Configuration
ACO	Ant Colony Optimization
AD	Anomaly Detection
ADL	Action Description Language
AF	Argumentation Framework
AI	Artificial Intelligence
AL	Active Learning
AMM	Agent Markov Model
API	Application Programming Interface
AR	Association Rules
AZ	AlphaZero
BPR	Bayesian Personalized Ranking
CBF	Content Based Filtering
CBM	Concept Bottleneck Model
CF	Collaborative Filtering
CRISP	CRoss Industry Standard Process
CTL	Computation Tree Logic
DLA	Direct Lookahead Approximation
DM	Data Mining
DP	Dynamic Programming
EHDA	Emulation of Human Decisions and Actions
EU	Expected Utility
EUT	Expected Utility Theory
FM	Factorization Machines
FMAR	Factorization Machines with Association Rules
FP	False Positive
GBDT	Gradient Boosting Decision Trees
GCN	Graph Convolutional Network
GDPR	General Data Protection Regulation
GNN	Graph Neural Network
HATP	Human Aware Task Planning
HiTL	Human in The Loop
HTN	Hierarchical Task Network
HTP	Hierarchical agent-based Task Planner
IRL	Inverse Reinforcement Learning
k-NN	K-Nearest Neighbor
LLM	Large Language Model
MARL	Multi-Agent Reinforcement Learning
MC	Monte Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MIP	Mixed-Integer Programming

ML	Machine Learning
MLP	Multi Layer Perceptron
MODM	Multi-Objective Decision Making
MOMADM	Multi-Objective Multi-Agent Decision Making
MOMARL	Multi-Objective Multi-Agent Reinforcement Learning
MOMDP	Multi-Objective Markov Decision Process
MORL	Multi-Objective Reinforcement Learning
MVP	Minimum Viable Product
NCF	Neural Collaborative Filtering
NLP	Natural Language Processing
OCC	One-Class Classification
PCBM	Post-hoc Concept Bottleneck Model
PCFG	Probabilistic Context Free Grammar
PDDL	Planning Domain Definition Language
PIAC	Per-Instance Algorithm Configuration
POMDP	Partially Observable Markov Decision Process
PSAC	Per-State Algorithm Configuration
RHIP	Receding Horizon Inverse Planning
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SDM	Sequential Decision Making
SSER	Shortest-Safe Evacuation Routes
SVD	Singular Value Decomposition
SVM	Support Vector Machine
ToM	Theory of Mind
TSP	Travelling Salesman Problem
VFA	Value Function Approximation
WP	Work Package
XAI	Explainable Artificial Intelligence
XAIP	eXplainable AI Planning
XBM	Explanation Bottleneck Model
XSDM	eXplainable Sequential Decision Making

1. Executive summary

Work Package 3 (WP3) has successfully advanced towards its objectives, and this D3.1 report presents a comprehensive literature review in progress as well as ongoing stakeholder discussions across multiple use cases. The report points out key deficiencies in current AI solutions that the literature review has identified, including inflexible search and planning methods, lack of transparency in recommender systems, limited interactivity in explainable AI (XAI), and rigid conversation management techniques.

To address these challenges systematically in practice, we report on the proposed PEER agent architecture that has been developed, enabling independent customization of AI components. In the first release of our Minimum Viable Product (MVP), this architecture has been instantiated in four PEER use cases:

- Sonae – A personalized in-store navigation assistant
- City of Amsterdam – A navigation app for wheelchair users
- Proditec – A human-in-the-loop anomaly detection trainer
- Datacation – AI-assisted warehouse configuration optimization

In D3.1 we also highlight the next steps which involve enhancing decision-making for Sonae, developing interactive prototypes for Amsterdam, Datacation and Proditec, and further integrating user modeling and explainability features across all applications.

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 3.1, provides a report on the progress of these three tasks so far, and details our work towards a "minimal viable product

of planning algorithms, XAI and user modelling, and human-AI conversation loop for lab evaluation" (MVP).

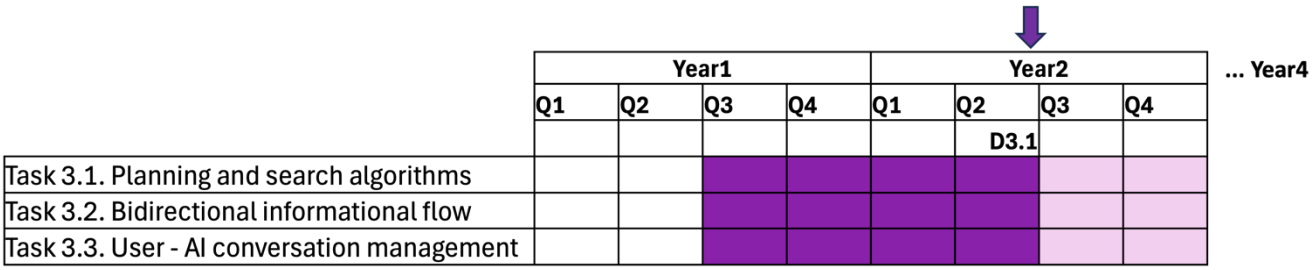


Figure 1. WP3 Gantt Chart

1.2 Deliverable Structure

This report is structured to provide a comprehensive overview of work performed in WP3, its processes, research, and integration with other work packages. Section 2 gives an overview of our work processes that led to the MVP, primarily literature study and stakeholder engagement. It also highlights the integration with other work packages and documents changes in one of the use cases. Section 3 focuses on the literature study conducted for each task, along with the first results and contributions made in WP3 as well as promising research directions to explore in this project. In Section 4, we propose a modular architecture for the envisioned PEER AI systems, composed of generic building blocks. Finally, Section 5 outlines how the proposed architecture has been realized in the MVPs for the individual use cases within PEER, together with planned next steps for their development.

2. Process Overview

WP3 officially commenced in Month 6 (M6), which explains why much of our work being reported in this deliverable is still in its early stages. Beside reviewing the literature and identifying research gaps for collaborative sequential decision-making algorithms, AI researchers in WP3 have been working on defining the concrete scope and functionalities of the **first release of Minimum Viable Product of the PEER Collaborative AI Assistant for Lab Evaluation.** This section reports the key activities undertaken so far, including stakeholder engagement, task specific literature review, exploration of relevant algorithms, and integration with the use cases.

2.1 Stakeholder Engagement

To ensure that the project aligns with real-world needs and expectations, we have actively engaged with key stakeholders of each use case, including domain experts and potential end-users. This engagement has been instrumental in gathering insights into practical challenges and requirements that will shape the development of the **PEER Collaborative AI Assistant**.

- **Meetings and Discussions:** A series of meetings has been conducted with stakeholders of each use case to align project objectives with their issues, workflows, and expectations. These interactions have facilitated the identification of critical user needs and relevant AI approaches, ensuring that our deliverables will be impactful.
- **Integration and interaction with WP2:** WP3 is tightly integrated with WP2. WP3 is building upon reports such as Deliverable 2.1, in which social and technical requirements were captured from end users to ensure that the AI algorithms being developed are in line with expectations and human value. WP2's work on e.g. novel low fidelity simulation method workshop formats (Deliverable 2.3) is informed in turn by the AI approaches taking shape in WP3.
- **Interaction with WP4 and WP5:** WP3, WP4, and WP5 work in close synergy to ensure the development and evaluation of AI-driven collaborative planning solutions. WP3 focuses on advancing the state-of-the-art methods and developing new AI approaches, while WP4 assesses their acceptability through user studies. To bridge these efforts, WP3 has initiated an analysis of AI system behaviors and engaged with use case stakeholders, leading to the creation of Figma mockups that simulate AI interactions. These mockups serve as a crucial tool for WP4, facilitating user evaluations and gathering insights on system usability and acceptance. Simultaneously, the feedback from WP4 helps refine the envisioned AI solutions in WP3, fostering an iterative design process that ensures the systems align with user needs and expectations. These mock-ups and simulated behavior will be reported as part of Deliverable 5.3 (WP5).

2.2 Scientific Literature Review

A comprehensive review of existing literature and state-of-the-art algorithms is currently underway as part of **Tasks 3.1 to Task 3.3.** This review aims to assess current methodologies, identify gaps in existing approaches, and inform the development of innovative solutions. We have conducted a systematic evaluation of contemporary AI models for sequential decision-making, as well as applicable explainable AI techniques and user modelling approaches relevant to WP3 objectives. This has helped us understand the strengths and limitations of existing approaches. To further align with PEER methodology and user-centric design, alongside the scientific literature review, regular discussions with stakeholders across all use cases helped us to identify key deficiencies in current AI solutions. These findings will guide the development of our collaborative AI assistant, ensuring that it addresses real-world challenges effectively.

2.3 Onboarding of a New Use Case

As part of WP3's ongoing efforts, we have successfully onboarded Datacation as a replacement use case for Continental. The Datacation use case is focused on AI support for logistics engineers in warehouses, whose tasks include organizing and re-organizing storage space and allocating deliveries efficiently. Challenges include complex legal and physical constraints on the storage of hazardous goods; uncertainty about exact timing and volume of future in- and outflows; efficient optimization with regard to the frequencies of handling different goods; preferences of customers; and others. This process involved aligning Datacation's expectations and contributions with the work already completed, ensuring smooth integration into ongoing tasks.

- **Alignment with Existing Tasks:** We have structured the collaboration in a way that minimizes disruptions while maximizing the contributions from Datacation. This includes ensuring that their requirements are adequately addressed within the existing framework of WP3.
- **Strategic Allocation of AI Beneficiaries:** Recognizing that all use cases will require solutions spanning multiple tasks, we have strategically assigned AI beneficiaries to specific use cases while allowing them to leverage their areas of expertise. Each AI beneficiary has been designated as a primary contact for one or two use cases, ensuring continued engagement and minimizing the need for redundant meetings.

3. State-of-the-art and Work Performed

3.1 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 work on automated construction/ updating of planning models (action models and control structures).

3.1.1 Introduction to Sequential Decision Making

The PEER project focuses mainly on problem settings where agents must make decisions sequentially and plan to reach a designated goal. Sequential Decision Making (SDM) is the process of making a series of interdependent choices over time, often in uncertain and dynamic environments. Unlike single-step decision-making problems, SDM requires considering how current actions influence future states and long-term objectives. This is fundamental in fields such as Artificial Intelligence (AI) [1], robotics, operations research, and finance, where agents must optimize their behavior over extended time horizons.
A widely used formal framework for these problems is the Markov Decision Process (MDP), which assumes the Markov property—i.e., the future state of the environment depends only on the current state and the agent's current action, and is independent of history [2]. In cases where the state is partially observable, the problem is extended to Partially Observable Markov Decision Processes (POMDPs), where the agent can infer hidden state information using a belief distribution over possible states. Solution techniques for these models include Dynamic Programming (e.g., value iteration, policy iteration), heuristic search (e.g., A* search, Monte Carlo Tree Search), and Reinforcement Learning [3] (e.g., Q-learning, policy gradient methods). Figure 2 gives a convenient symbolic representation of MDP problems. These models provide a formal structure for representing states, actions, transition probabilities, and rewards (e.g., goals), enabling the use of methods for optimal decision-making.

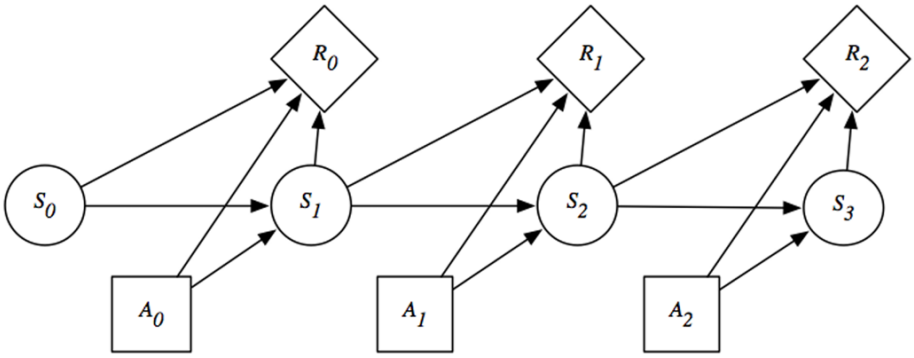


Figure 2. Symbolic Representation of MDP[4]

This section briefly introduces planning and classical search algorithms and highlights the research gaps and requirements for human-AI collaboration.

3.1.2 Decision and Utility Theory

Decision theory provides a formal framework for making rational choices under uncertainty. It encompasses normative models, which describe how decisions *should* be made to maximize expected utility, and descriptive models, which study how decisions are *actually* made by humans [5]. The foundational concept in decision theory is Expected Utility Theory (EUT), where an agent selects actions to maximize the expected value of a utility function that encodes decision makers' preferences over outcomes. Under this framework, rational agents should always choose the action that maximizes their expected utility. However, human decision-making often deviates from this ideal due to cognitive biases, loss aversion, and bounded rationality [6]. Alternative models, such as Prospect Theory, account for these psychological factors by weighting probabilities of outcomes non-linearly and incorporating reference-dependent preferences. Utility theory extends decision theory by defining preference structures that can be either ordinal (ranking outcomes) or cardinal (assigning numerical values). A fundamental challenge in decision-making is defining an appropriate utility function, particularly when multiple objectives or stakeholders are involved. This leads to the distinction between single-objective and multi-objective problems, which are crucial in PEER AI assistant designed to fine-tune AI models or make decisions on behalf of humans.
In single-objective decision problems, there is a well-defined single goal, such as minimizing cost, maximizing reward, or achieving a specific performance metric. These problems are common in robotics (e.g., shortest path planning), reinforcement learning (e.g., maximizing cumulative reward), and finance (e.g., portfolio optimization). Traditional approaches, such as MDPs and DP, provide solutions for these problems by computing an optimal policy that maximizes a scalar objective function [2]. However, many real-world decision problems involve trade-offs between competing objectives, leading to multi-objective decision-making (MODM). In these cases, an agent must balance conflicting goals, such as cost vs. quality in manufacturing, safety vs. speed in autonomous driving, or energy consumption vs. performance in cloud computing [7]. A standard approach to multi-objective decision-making is to identify Pareto-optimal solutions, where no objective can be improved without degrading another. Multi-objective reinforcement learning (MORL) techniques, such as Pareto Q-

Learning and *Scalarization* methods, address these challenges by *learning* policies that balance competing criteria dynamically [8].

3.1.2.1 The Role of Decision Maker Preferences

Preferences play a crucial role in decision-making, especially when AI systems act on behalf of humans. In single-objective settings, preferences are often encoded implicitly within the objective function, whereas in multi-objective settings, they must be explicitly modelled to capture trade-offs between different criteria. Reference elicitation methods, such as utility surveys, interactive learning, and inverse reinforcement learning (IRL), aim to infer human preferences from behavior and demonstrations. One key challenge is the alignment problem—ensuring that AI agents make decisions that align with human values [9], ethical considerations, and societal norms. This is particularly relevant in high-stakes domains such as healthcare, where an AI system planning treatments procedure must balance effectiveness, patient comfort, and ethical considerations. In both *Task 3.1* and *Task 3.2*, we study how methods such as Explainable AI (XAI), seek to bridge this gap by incorporating human feedback, enabling collaboration in the planning and reasoning process [10].

3.1.3 Planning and Search Problem

Formally, a SDM can be defined as a tuple (S, A, T, R, γ), where S is the world states, A is the set of possible actions, $T(s'|s,a)$ represents the transition dynamics defining the probability of reaching state $\acute{s}$ from state s after taking action a, $R(s,a)$ is a reward function that quantifies the immediate benefit of taking action a in state s, and $\gamma \in [0,1]$ is a discount factor that weighs the importance of future rewards [11]. The goal is to find an optimal policy or plan π^* that maximizes the (expected) cumulative reward over a time horizon. Considering the utility theory, formally, the decision maker faces a set of possible actions a and each action $a \in A$ leads to a set of possible states/outcomes s' with associated probabilities $T(s'|s,a)$ and utilities $U(s')$, the Expected Utility (EU) of an action is given by:

$$EU(a) = \sum_{\acute{s} \in S} T(s'|s,a)U(s')$$

The most basic form of AI planning is defined by a single-agent system operating in a deterministic and fully observable environment. In such a setting, every action always succeeds as intended, the agent has complete knowledge of the world state at all times, and actions occur instantaneously without any temporal constraints. A classical planning problem consists of four key components: (a) a set of all possible world states, S ; (b) a set of valid state transitions, $T \subseteq S \times S$; (c) an initial state,
$s_0 \in G$; and (d) a collection of goal states, $G \subseteq S$. A plan, then, is simply a sequence of transitions—each corresponding to an action—that takes the system from the initial state s_0 to one of the goal states $s_g \in G$ [12]. We can visualize an environment as an explorable state-space graph where nodes represent states of the world, which are total assignments of value to each feature, and edges correspond to actions that can be carried out from the source state

node to the destination one. An algorithm that searches the state-space graph from the initial state looking for a state that satisfies a goal description is called a forward planner, this is more convenient than just using a state-based representation of the actions since it dynamically generates the parts of the search graph that are relevant, based on available actions and, eventually, a heuristic function that estimates the cost of solving the goal from the state, like in the case of A* [4]. Various representations exist for defining such transition systems, with the Planning Domain Definition Language (PDDL) [13] being the most widely adopted standard. PDDL was designed to unify and standardize AI planning languages, drawing inspiration from earlier formalisms such as STRIPS [14] and ADL (Action Description Language) [15]. Unlike these earlier propositional logic-based representations, PDDL employs a more compact, first-order logic approach using predicates. A PDDL-based planning problem is divided into two key components: the domain and the problem instance. The domain file specifies the fundamental structure and rules governing the environment, independent of any specific planning task. It defines: object types categorizing the elements within the world, predicates that describe possible relationships between objects, and operators (or actions), made of parameters, preconditions and effects. Preconditions must be satisfied for the action to be executable in a given state, and the effects describe how the state changes after executing the action. The problem file, on the other hand, is tailored to a specific planning instance, detailing the concrete objects in the world, the initial conditions, and the desired goal state. Over time, PDDL has been extended to incorporate additional features. For example, PDDL2.1 introduced time and numerical effects, PDDL3 added support for constraints and preferences, and PDDL+ expanded the language to model processes and events.

When applying PDDL to human-AI collaborative planning scenarios, the aforementioned planning definition needs to be extended and framed as a temporal PDDL problem with differing action costs and execution durations, accounting for different agent capabilities and preferences. For instance, a particular action may be faster for one agent but come at a higher cost due to personal preferences. To account for such trade-offs, the planning metric to be minimized is defined as a weighted sum

$g(p) = \alpha \cdot m(p) + (1 - \alpha) \cdot c(p)$ where $m(p)$ represents the makespan of the plan (i.e., the total time required for execution), $c(p)$ denotes the total cost of all actions in the plan, and $\alpha \in [0,1]$ is a weight parameter that determines the balance between minimizing makespan and cost [16], [17]. By adjusting α, the planning process can be adapted to accommodate varying human preferences and agent capabilities, ensuring that the generated plan aligns with different optimization priorities.

3.1.3.1 Search Algorithms

In SDM, finding optimal paths through state spaces can be approached through various search algorithms, with the choice depending on available information and the trade-off between optimality and computational efficiency. Consider a path-planning problem in a geographical zone, where we can represent the environment as a state space S with well-defined transition dynamics T and rewards R. All of this can be treated as a state-space graph to explore with

a forward planner. Informed search algorithms leverage domain knowledge about the goal state to guide exploration. A* search, a prominent best-first search algorithm, makes decisions by evaluating states using a function $f(s) = g(s) + h(s)$, where $g(s)$ represents the actual cost (negative reward) accumulated from the initial state to the current state s, and $h(s)$ is a heuristic function estimating the cost to reach the goal state [1]. When the algorithm arrives in a node s, A* decides to choose the adjacent node with the lowest $f(s')$ deterministic value, this can be formalized in our SDM framework (S, A, T, R, γ), as a process where A* selects actions $a \in A$ at each state s based on minimizing $f(s')$, where s' represents the next state according to the transition function $T(s'|s, a)$. For geographical pathfinding, a common heuristic is the Euclidean distance to the goal, which maintains consistency properties that guarantee optimality, however A* star can be used in many other fields like Natural Language Processing (NLP), where, for instance, in PCFG (Probabilistic Context Free Grammars) parsing a specific heuristic called "Viterbi score" is used to predict the most probable completion [18]. Going back to the example of pathfinding in a geographical zone, a good choice as a heuristic function $h(s)$ is the Euclidean distance between the reached point in the map and the goal, this is because it is considered consistent so it maintains the property $h(A) \leq cost(A, B) + h(B)$ [19].

While A* excels in single-agent scenarios, its effectiveness diminishes in multi-agent settings where the transition dynamics T become more complex due to agent interactions, potentially leading to recurring collisions and replanning cycles.

Dijkstra's algorithm can be viewed as a special case of A* where $h(s) = 0$, making it an uninformed search method that doesn't utilize problem-specific information. Within our SDM framework, Dijkstra's algorithm exhaustively explores the state space S by evaluating all possible transitions $T(s'|s, a)$ and their associated rewards $R(s, a)$. While this comprehensive approach guarantees optimality for non-negative reward functions, it typically requires more computational resources than A* when a good heuristic is available [20].

However, both A* and Dijkstra's algorithm face significant challenges when the transition function $T(s'|s, a)$ changes during execution. Dijkstra's algorithm, assuming static transition dynamics, requires complete recalculation when the environment changes, making it computationally expensive in dynamic settings. While modifications exist to handle predictable changes in the reward function $R(s, a)$ (like varying traffic conditions), these adaptations struggle with rapid or unexpected environmental changes.

A* encounters similar issues: its heuristic function may become unreliable in dynamic environments. When state transitions change unexpectedly, the heuristic estimates of future costs may no longer accurately guide the search, potentially negating the efficiency advantage of A* over uninformed methods [21]. For example, in path-finding problems, sudden environmental changes can invalidate Euclidean distance heuristics, necessitating complete path recalculation.

While in some tasks using a regressive function with a heuristic to estimate the value of possible outcomes is something feasible and that has led to excellent results (i.e. chess and checkers), in several other tasks building said function is a complex issue, one classic example is the game of Go.

Monte Carlo (MC) Simulations is a widely used class of methods based on random sampling that managed to obtain really good results in many fields, by constructing a best-first search method guided by results of MC Simulations, e.g. random exploration of the search space, Monte Carlo Tree Search was born [22].
The basic framework consists in constructing a tree of evaluated states, this set of states is built incrementally iterating over four steps:

1. **Selection:** Traverse from a root state s to a leaf state s_{leaf} using a selection policy π that balances exploration and exploitation
2. **Expansion**: Create child states s' according to the transition function $T(s'|s_{\text{leaf}}, a)$ for available actions $a \in A$
3. **Simulation:** From the expanded state, simulate trajectories using a rollout policy, obtaining cumulative rewards R
4. **Backpropagation:** Update value estimates of all traversed states using the accumulated rewards, weighted by the discount factor γ [23].

The outline of the method is shown in Figure 3.
In contrast to traditional search methods, MCTS offers a distinct advantage as an online planning approach. Rather than computing complete policies in advance, MCTS performs decision-time planning using the current state of the environment. This means when the transition function $T(s'|s, a)$ changes, MCTS can immediately incorporate these updates into its simulation model, adapting its decision-making process to the new dynamics.
However, this flexibility comes with a trade-off. While MCTS can rapidly adjust to environmental changes, its convergence to optimal decisions can be slow in domains with large state-action spaces [24].

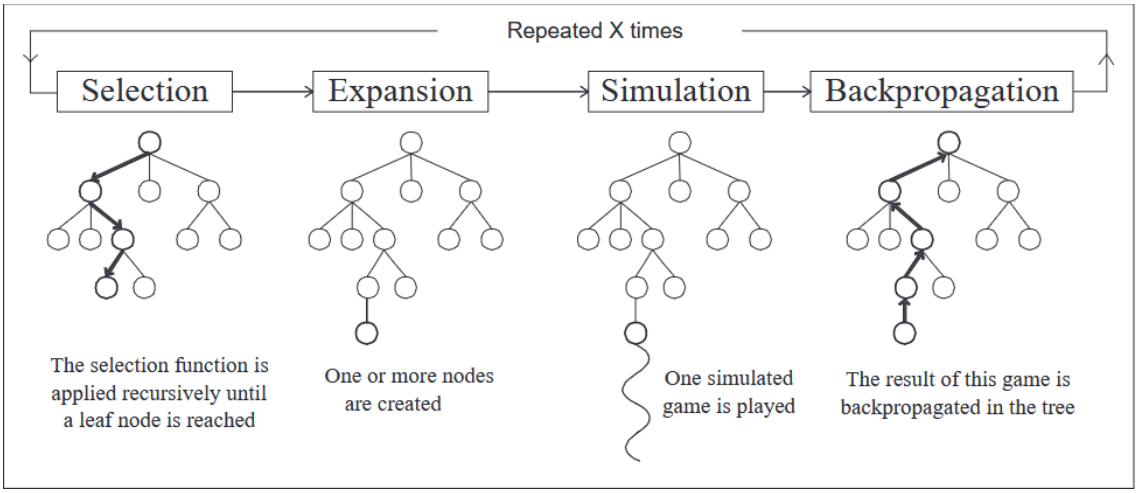


Figure 3: Outline of Monte-Carlo Tree Search [23]

Efficient decision-making in dynamic and uncertain environments is a fundamental challenge in planning and optimization. Traditional search algorithms, such as Dijkstra's algorithm and A*, rely on static heuristics and predefined costs, which limit their adaptability in environments with *changing conditions*. For example, considering an autonomous car navigating a city, Dijkstra and A* compute the shortest path to its destination before starting the trip and if suddenly a segment of route becomes blocked, the generated path is invalid,

and the algorithms must restart or perform replanning. There are variants of A* that extends the algorithm to more dynamic settings and was studied by Likhachev et al. [25]in their Anytime Dynamic A*, a graph-based planning algorithm that adapts the quality of the solution to the available time and repairs its previous solution when the environmental graph gets updated. With an analogous technique, Oyola et al. [26] introduced SSER (Shortest-Safe Evacuation Routes), a Dijkstra-based algorithm based on vector graphs designed to work in Safe Evacuation Routes in dynamic environments.

3.1.3.2 Reinforcement Learning for Policy Optimization

Reinforcement Learning (RL) emerges as a powerful paradigm for solving complex sequential decision problems (e.g., MDPs) where agents must learn optimal behaviors through interaction with their environment. While planning algorithms like A* and MCTS assume known transition dynamics T and reward functions R, RL addresses scenarios where these components must be learned through experience.

RL formalizes how agents learn to make sequential decisions by trial and error, where positive rewards reinforce beneficial behaviors, and negative rewards discourage unfavorable ones. The fundamental goal remains consistent: finding an optimal policy π^* that maximizes expected cumulative rewards.

Value functions serve as the cornerstone of many RL algorithms, quantifying the expected return an agent can achieve from different states or state-action pairs. There are four primary value functions to consider:

1. The on-policy value function $V^{\pi}(s)$ represents the expected return when starting from state s and following policy π
2. The on-policy action-value function $Q^{\pi}(s,a)$ extends this by considering the initial action explicitly
3. The optimal value function $V^{*}(s)$ captures the best possible expected return from state s
4. The optimal action-value function $Q^{*}(s,a)$ combines both concepts of capturing the best expected return by considering an explicitly defined initial action.

A fundamental distinction in RL algorithms lies in whether they utilize a model of the environment's dynamics. Model-based methods explicitly learn or use the transition function T and reward function R for planning, like the search algorithms discussed earlier. However, learning accurate models for complex environments often proves challenging.

Model-free methods, which form our primary focus, learn optimal behaviors without explicitly modelling T and R. These approaches fall into two main categories:

1. **Policy Optimization:** These methods directly learn a parameterized policy $\pi_{\theta}(a|s)$, optimizing the parameters θ to maximize expected returns. This typically occurs on-policy, using data collected from the current policy version.
2. **Q-Learning:** These algorithms learn to approximate $Q^{*}(s,a)$, typically in an off-policy manner that can utilize data collected under any policy. The learned Q-function produces directly a policy through action selection: $a(s) = argmax_a\, Q_{\theta}(s,a)$

Actor-Critic methods bridge these approaches by combining a policy (actor) with a value function approximator (Critic). A partial taxonomy of modern algorithms in RL can be seen in **Error! Reference source not found.**.
The learning process can be either online, where the agent continuously updates its knowledge during interaction with the environment, or offline, where learning occurs on a fixed dataset of experiences. This distinction becomes particularly relevant when considering real-world applications where live interaction might be costly or risky [27].

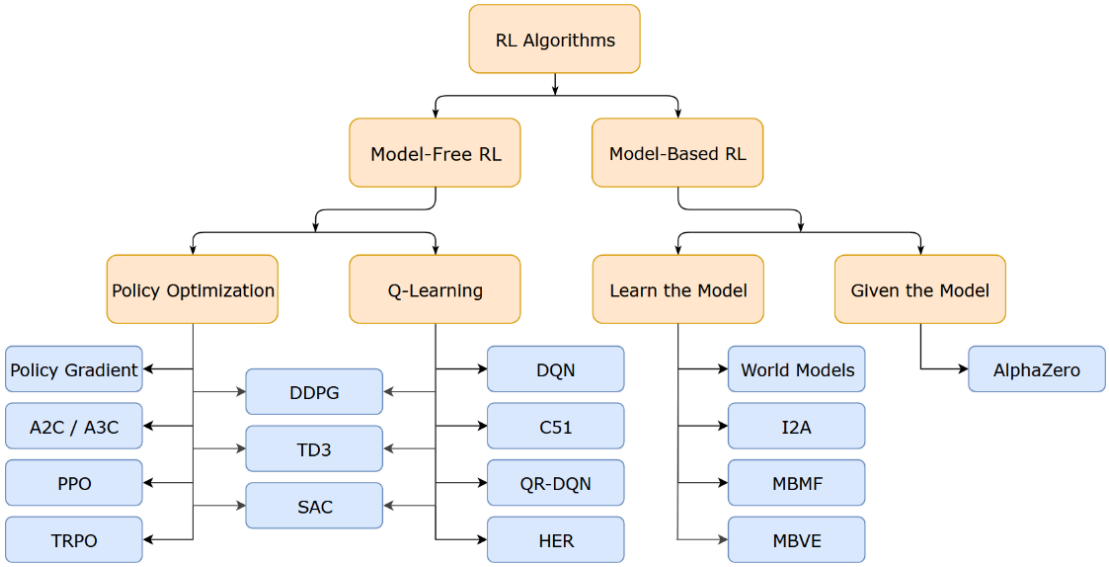


Figure 4: Partial Taxonomy of Modern RL Algorithms [27]

3.1.3.3 Making Search More Flexible and Adaptive

The goals of PEER require search and planning algorithms to be more flexible and adaptive – in the Amsterdam use case for example to potentially changing (AI estimates of) user needs, as well as to potentially changing and non-stationary environments. This has been reflected so far in two strands of our prior and ongoing work.

Algorithm Configuration in Sequential Decision Making [28]**.** Proper parameter configuration is essential to ensure that algorithms operate efficiently and effectively. This can be a time-consuming and difficult task, especially when many parameters need to be tuned
or when the evaluation of parameter settings is expensive to compute – which is commonly the case with *anytime* algorithms like MCTS that make better decisions with more runtime. This algorithm configuration (AC) problem has been widely studied by the AI community [29], and its approaches can be categorized as static or dynamic, depending on whether the parameters are set at the beginning of algorithm execution, or changed at every iteration of the running algorithm [30]. Static approaches can be further divided into Per-Distribution AC, where the goal is to find a single set of parameters that works well across an entire

distribution of problem instances, and Per-Instance AC (PIAC), which tries to find a function that maps each individual problem instance to a specific parameter configuration.
The algorithm configuration literature mainly focuses on approaches for solving optimization problems, such as local search or genetic algorithms, which aim to find an optimal solution by cleverly exploring the space of solutions, typically by iteratively improving candidate solutions. In our work, we focus on SDM problems instead. SDM algorithms can work online, by outputting one decision at a time instead of a complete solution (sequence of decisions). They must therefore be applied multiple times with different initial environment conditions to obtain a full solution. This approach introduces a new dimension which should be taken into account by algorithm configuration methods. While SDM algorithms are often applied in practice with a fixed set of parameters for decision-making in a given problem instance, it can be beneficial to change them according to the current state of the instance at hand.
The initial board position in a game of chess for example, when all the pieces are still available, is usually very different in terms of complexity from the board state in a position close to the end of the game, when most of the pieces have likely been captured. This can have a strong impact on the behavior and performance of SDM algorithms, leading to the natural idea of setting algorithm parameters accordingly.
We have formalized this as Per-State Algorithm Configuration (PSAC) and have proposed a framework based on Reinforcement Learning (RL) to learn how to do it in an automated way. We have demonstrated the effectiveness of our method on two popular SDM algorithms: Monte Carlo Tree Search (MCTS) and AlphaZero (AZ) [31], a state-of-the-art combination of MCTS and deep learning. We tested the first on a warehouse assignment problem and the second on the game of Connect Four. The results highlighted the potential of our algorithm configuration approach to improve the performance of sequential decision-making algorithms.

Decision making in non-stationary environments with policy augmented search [24]**.** An open challenge in sequential decision-making settings are non-stationary environments, where the dynamics of the environment can change over time. A decision agent must adapt to these changes to avoid taking sub-optimal actions.
Two well-known approaches for sequential decision-making are reinforcement learning (RL) and online planning [32]. In RL approaches, an agent learns a policy, i.e., a mapping from states to actions, through interacting with the environment. The learning can also take place before execution using environmental models. Once a policy is learned, it can be invoked nearly instantaneously at decision time. Deep RL methods, which use a neural network as a function approximator for the policy, have achieved state-of-the-art performance in many applications. However, when faced with non-stationary environments, a policy can become stale and result in sub-optimal decisions. Moreover, retraining the policy on the new environment takes time and considerable computational effort, particularly in problems with complex state-action spaces. While RL algorithms have been designed to operate in non-stationary environments [33], there is a delay between when a change is detected and when the learning framework converges to the updated policy. Depending on the problem setting, such delays might be very expensive.

An alternative approach is using algorithms such as Monte Carlo tree search (MCTS) to perform online planning. These approaches perform their computation at decision time using high-fidelity models of the current environment to find promising action trajectories. These models can be updated as soon as environmental changes are detected, and such changes can be immediately incorporated into decision-making (assuming that the generative model used to build the search tree can be updated quickly). MCTS has been proven to converge to optimal actions given enough computation time [34], but convergence can be slow for domains with large state-action spaces. The slow convergence is a potential issue for problem settings with tight constraints on the time allowed for decision-making, e.g., in emergency response, when an incident occurs, any time used for decision-making increases the overall response time [35].
Both RL and MCTS thus have weaknesses when applied individually to complex decision-making problems in non-stationary environments. In our work, we have developed a hybrid decision-making approach that integrates RL and online planning, aiming at combining their strengths while mitigating their individual weaknesses in non-stationary environments. The intuition is that if the environment has not changed too much between when an optimal policy was learned and when a decision needs to be made, the policy can still provide useful information for decision-making. Similar hybrid approaches have been explored and have achieved state-of-the-art performance in many domains. For example, AlphaZero integrates a policy and value network within a modified MCTS and has achieved state-of-the-art performance in games such as Go. However, such a hybrid approach has not been developed before specifically for non-stationary environments.

3.1.3.4 Planning Algorithms

Automated planning deals with selection and organization of actions to achieve some goal. The goal is frequently specified as a goal condition that the world state should satisfy, and the plan is a totally ordered sequence of actions. Various approaches to obtain a plan exist and among them the forward state planning (progression) is the most widely used. The algorithm starts at the initial state, collects all actions applicable at that state and then selects one and applies it to the state. This process is repeated until a state satisfying the goal condition is reached. Alternatively, one can start with the goal condition, collect actions relevant for that condition, and then select one and modify the goal condition to describe states from which the selected action is applicable and reaches the goal state. The plan is built backwards, and the algorithm ends when the initial state is encountered. This is called backward (regression) planning. Search algorithms (such as A*) are used to implement the exploration of possible action sequences. A more flexible approach that can construct partially ordered plans is based on exploration of partial plans. The algorithm starts with a plan consisting of action modelling the initial state in its effect and action modeling the goal condition as its precondition. Then the algorithm is correcting the partial plan by adding actions to cover so called open goals (preconditions of actions in the plan that are not yet satisfied by some action – there is no causal link for them) and ordering the actions to remove

threats when one action possibly destroys a causal link between two other actions. This process is repeated until a complete plan without flaws is obtained. The approach is called plan space planning, and it is also implemented using a search algorithm that explores possible plan corrections. Other approaches to planning exploit, for example, a planning graph, where independent actions are applied in parallel (and hence their ordering in the plan can be arbitrary), or compilation to other solving approaches such as Boolean satisfiability.

3.1.3.5 Planning and Learning in Multi-Objective Environments

To navigate the complexities of real-world environments, agents must construct temporally extended plans where each decision influences future outcomes. While most research focuses on environments with a clear goal, exemplified by the prevalence of game-like environments where the sole objective is to win, many real-world scenarios are more complex and involve trade-offs between different objectives [36]. For instance, the travelling salesman problem may involve additional objectives such as minimizing gas emissions and ensuring safety conditions; and the end users of the Amsterdam use case in PEER may trade off the length of a route through the city with its complexity (number of street crossings to traverse), its distance to public transportation (for safety), how busy the route is (lively shopping streets are harder to navigate with wheelchairs), etc.

To solve single-objective sequential decision-making problems, the *expert iteration* approach combines model-based reinforcement learning with planning [37]. Expert iteration proposes a two-phase process where an expert (such as MCTS) generates trajectories in the environment and an apprentice (such as a neural network) is trained on the collected dataset to imitate the expert. Alternating between these phases results in strong policies that achieve super-human performance in various challenging settings [31].

In our work [38], we extend the expert iteration framework to multi-objective sequential decision-making. Specifically, we consider an agent optimizing its expected utility under a given utility function in a multi-objective Markov decision process (MOMDP). We have found that it is possible to reduce the MOMDP to a single-objective MDP with a terminal reward. This MDP can then be solved using standard expert iteration algorithms such as AlphaZero. While this baseline can be applied with minimal modifications, it explicitly eliminates the multi-objective nature of the environment.

Inspired by *distributional reinforcement learning* [39], we have therefore proposed a novel algorithm that learns the distribution over vector returns in the environment. This allows the expert iteration framework to utilize properties of this distribution during learning and enables transfer to alternative utility functions post-learning. For instance, during learning, the expert can use properties such as variance and conditional value at risk in its planning routine. After training, these statistics can also be provided to the user to enhance interpretability. Moreover, having access to a distribution facilitates efficient transfer to different utility functions.

3.1.3.6 Learning in Multi-Agent, Multi-Objective Environments

Often, domains of critical social relevance such as smart electrical grids, traffic systems, taxation policy design, or infrastructure management planning involve controlling multiple agents while making compromises among several conflicting objectives. This is potentially also true of PEER use cases such as Sonae, where we could imagine supporting many customers at the same time, each with different shopping lists that could interfere with each other in various ways - leading to e.g. wasted time if bottlenecks at a high-demand meat counter are not avoided by finding compromises between the optimality of different customers' shopping routes. The multi-agent aspect of the above-mentioned domains has been well studied, and there is a wealth of literature on multi-agent approaches spanning several decades [40]. Separately, over the last 20 years, there has been an increasing interest in the multi-objective aspect of such domains. However, the intersection of these two aspects, multi-objective multi-agent decision making (MOMADM), has not received much attention. Indeed, problems are often simplified by hard coding a trade-off among objectives or centralizing all decisions in a single agent. We argue that many, if not most, complex problems of social relevance have both a multi-objective and a multi-agent dimension. This is because such problems often affect multiple stakeholders, who may care about different aspects of the outcome, and may have different preferences for them. As such, it is crucial to advance the field of MOMADM to enable future progress in the application of artificial intelligence (AI).

The development of standardized benchmarks is a key factor that has driven progress in various areas of AI over the years. Without standardized, publicly available benchmarks, researchers spend a lot of unnecessary time re-implementing test environments from published papers, reproducibility is made much more difficult, and results published in different papers are potentially incomparable [41]. Suites of standardized benchmarks have helped to address these issues already in some fields of AI such as reinforcement learning. Such benchmarks are exemplified by the seminal Gymnasium library [42] for single-objective single-agent RL, the PettingZoo library [43] for multi-agent RL (MARL), and MO-Gymnasium [44] for multi-objective RL (MORL). Yet, there is no existing library dedicated to multi-objective multi-agent reinforcement learning (MOMARL).

In our work, we have introduced MOMAland (see Figure 5), the first publicly available set of MOMARL benchmarks under standardized Application Programming Interfaces (APIs). MOMAland, incorporated within the Farama Foundation ecosystem, draws inspiration from analogous projects and currently offers over 10 configurable environments encompassing diverse MOMARL research settings. By embracing open-source principles and inviting contributions, we anticipate that MOMAland will evolve in tandem with research trends and host new environments in the future. MOMAland aims to contribute to the unification of rather fractured evaluation practices, in order to provide researchers with clear and objective data on how well their algorithms perform with respect to other methods.

Additionally, MOMAland includes utilities and learning algorithms intended to establish baselines for future research in MOMARL. Notably, it offers utilities enabling the utilization of existing MORL and MARL solving methods through centralization or scalarization strategies.

Importantly, while the provided baselines can find solutions for certain MOMARL settings, MOMAland also features challenges with no known solution concept. Addressing these challenges requires tackling open research questions before deriving appropriate solving methods.

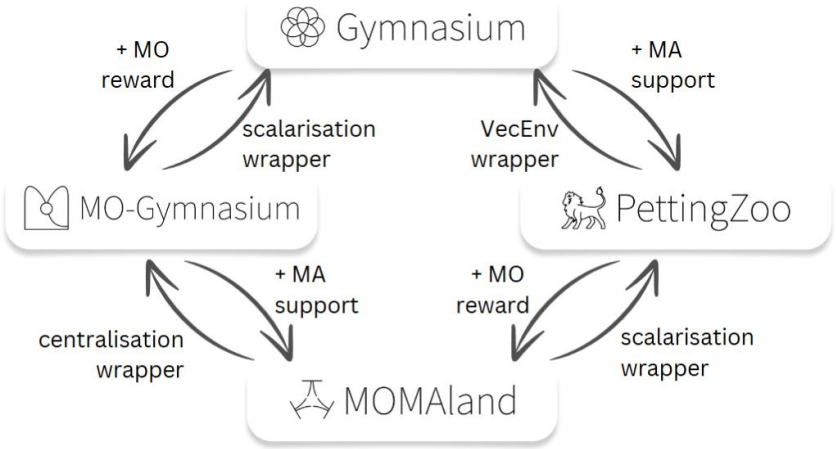


Figure 5. Overview of the libraries related to MOMAland within the Farama Foundation

3.1.3.7 Nature-inspired Optimization for Search and Planning

Although MCTS and RL focus on probabilistic reasoning, they may struggle with combinatorial explosion in large-scale planning problems [24], [45]. In these settings, algorithms such as Ant Colony Optimization (ACO) present a bio-inspired alternative for solving SDMs. Originally designed for combinatorial optimization, ACO models decision-making as a probabilistic path selection process, where artificial ants explore a graph-based representation of the problem space, e.g., static Travelling Salesman (TSPs). However, variants of ACO have been developed to adapt to dynamic environments, making it useful for real-time planning and decision-making. Through pheromone-based reinforcement, the system converges toward optimal or near-optimal solutions over time. This makes ACO particularly well-suited for planning problems requiring adaptive, decentralized, and scalable decision-making strategies. Algorithm 1 shows a generic structure of ACO framework for *single objective* (e.g., shortest path) setting.

```
Algorithm 1:
Procedure AntColonyOptimization:
    Initialize necessary parameters and pheromone trials;
    while not termination do:
        Generate ant population;
        Calculate objective values associated with each ant;
        Find best solution through selection methods;
        Update pheromone trial based on the value of objective function;
    end while
end procedure
```

To further extend this algorithm to optimize more than one objective (e.g., multi-objective optimization problem), previous research explored methods that can reveal all solution trade-offs at once, leading to Multi-Objective Ant Colony System. However, providing all trade-offs to the user not only adds extra complexity for the decision-maker to choose a plan from but also requires more computational capacity. Moreover, assuming that preference of the user is subject to change in every decision step, computing the stepwise *Pareto front* plans are not feasible. To address this challenge, we propose to consider m colonies corresponding to the number of objectives, each one solving a single objective optimization problem. Having pheromone information for each objective, we can combine the normalized pheromone structures using a *utility function* parametrized by a set of weights w. For TSPs, Algorithm 1 below provides a general overview of this approach where the probability of choosing vertex v_i by colony c while building the solution is p_S^c. Each colony has its own heuristic function and objective specific phremone factor.

$$p_S^c(v_i) = \frac{[\tau_S^c(v_i)]^\alpha . [\eta_S^c(v_i)]^\beta}{\sum_{v_j \in Cand} [\tau_S^c(v_j)]^\alpha . [\eta_S^c(v_j)]^\beta}$$

Data: $G = (V, E), \#COL(m - objectives), initialized - \tau$
Result: Updated Pheromone Structures
repeat
  for *each colony* $c \in m$ **do**
    for *each ant* $j \in c$ **do**
      Construct a feasible solution S by adding vectors v
    end
  end
  Update pheromone structure τ_c according to objective c
until *maximal number of cycles reached*;

Algorithm 1. Multi-Objective Ant Colony System

3.1.3.8 User-centric Planning and Search

Path planning algorithms like A* that have traditionally operated as purely computational systems can now benefit from the emergence of Large Language Models (LLMs), which have opened new possibilities for integrating human insight and commonsense reasoning into even the most technical algorithmic processes. LLM A* [46] evaluates states using a cost function that combines path costs and heuristics but extends this framework by leveraging LLMs to identify feasible moves while considering contextual understanding of the environment and available actions. The algorithm proceeds in two main stages: first, the LLM is oriented to comprehend the specific environment and action space; second, it conducts path planning between initial and goal states while enabling human intervention. This human-in-the-loop approach allows for guidance during planning, verification of proposed moves, and efficient token usage through focused communication.

When applying PDDL to human-AI collaborative planning scenarios, the aforementioned planning definition needs to be extended and framed as a temporal PDDL problem with differing action costs and execution durations, accounting for different agent capabilities and preferences. For instance, a particular action may be faster for one agent but come at a higher cost due to personal preferences. To account for such trade-offs, the planning metric to be minimized is defined as a weighted sum
$g(p) = \alpha \cdot m(p) + (1 - \alpha) \cdot c(p)$ where $m(p)$ represents the makespan of the plan (i.e., the total time required for execution), $c(p)$denotes the total cost of all actions in the plan, and $\alpha \in [0,1]$ is a weight parameter that determines the balance between minimizing makespan and cost [16], [17]. By adjusting α, the planning process can be adapted to accommodate varying human preferences and agent capabilities, ensuring that the generated plan aligns with different optimization priorities.

The objective of this methodology is to adapt and influence decision-making processes based on human feedback and preferences, which are often expressed in natural language and difficult to directly translate into structured planning and search problems. To address this, the proposed method integrates an interface module that converts unstructured inputs—such as natural language commands or external database information—into structured formats suitable for automated planning. This is achieved by combining traditional decision-making techniques with LLMs, enhancing flexibility and adaptability in planning systems. Figure 6 illustrates how human feedback is processed and structured, reducing the need for manual translation, which is typically rigid and time-consuming. This approach has been previously applied to collaborative human-robot environments [47], where an LLM-based system is used as a translator between high-level natural language inputs and a structured PDDL planning problem, specifically defining planning goals and action costs. In this way, adaptive planning can handle high-level goals and influence the action allocation based on the varying human preferences, states and conditions, enabling more intuitive interactions and improved responsiveness to user preferences.

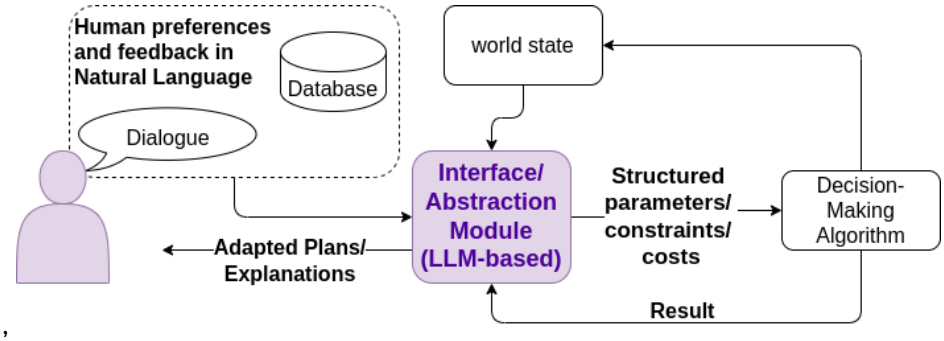


Figure 6. Collaborative Human-Robot Planning Architecture

In the Sonae use case (AI-based in-store navigation), this methodology could translate a customer's natural language request—such as "Show me the fastest route to gluten-free products"—into specific parameters within a structured planning problem. It would extract key constraints like product category, store layout, and user preferences (e.g., avoiding crowded aisles) and encode them into a formalized navigation model. The actual route would

then be computed by the underlying planner based on these structured inputs. For the City of Amsterdam's use case, this approach would process user input like "Find a wheelchair-friendly route with minimal slopes" by identifying relevant constraints such as accessibility requirements, terrain difficulty, and distance. These constraints would be formally represented within a structured planning model, which the navigation planner would then use to generate an optimal, accessible route tailored to the user's needs.

Human-aware planning may be considered as an instance of multi-agent planning [48] which involves incorporating a model of the human's perception and beliefs [49] into traditional task planning. It enables agents to consider human expectations and responses when searching for a feasible sequence of actions that allows the fulfillment of a desired state of the world. The planning problem may be seen as one where the planner must produce a "joint plan" which reconciles the different models of the planning problem while conforming to human goals and beliefs, albeit at the cost of optimality for the agent [49]. Human-aware planning plays a key part in explainable AI planning (XAIP), where plans must be interpretable and acceptable from the human's perspective. By adapting plans based on both task feasibility and human satisfaction, agents become more responsive to human needs, optimizing both task outcomes and user experience. Several approaches to human-aware planning have been identified in the literature. Shekhar et al. [50]use techniques drawing from Theory of Mind (ToM) for perspective-taking to model the human's beliefs about the agent's actions in the Human-Aware Task Planning with Emulation of Human Decisions and Actions framework (HATP/EHDA) [51]. The authors then combine this model with a novel AND/OR search-based solver to handle uncertainties and diverging beliefs. Alternatively, Canal et al. attempt to model human behavior and preferences with symbolic task planning during execution [52] by building a user model based on the human's responses to simple questions, and integrating it into the planning domain. The method uses a probabilistic planner and adapts its actions according to both user satisfaction and task execution outcomes by adjusting penalizations applied to the planner's rules. Izquierdo-Badiola et al. [16] present a symbolic planning framework that integrates human states as PDDL action costs to prevent action failures due to unforeseen human behavior, by replanning and reassigning the tasks in anticipation when the agent state changes. The authors further develop the approach by optimizing for heterogeneous action costs and durations for different agents during the planning process [17]. In another approach, Cheng et al. create a hierarchical task model from human demonstrations which captures sequential and parallel relationships within the task [53]. An optimization-based planner with the objective of minimizing task completion time then prioritizes actions parallel to human actions. Buisan et al. take inspiration from Hierarchical agent-based Task Planners (HTPs) to explore several Hierarchical Task Networks (HTNs) to produce a human-agent joint plan [54]. In the CommPlam framework presented by Unhelkar et al. [55], the human model is represented as an Agent Markov Model (AMM) and Partially Observable Markov Decision Processes (POMDPs) are used to compute the agent's policy for collaboration while reasoning about its actions and communication with the human, under partial observability of the human's mental states.

Generative AI models such as LLM have also been explored to model and simulate human behavior [56], [57], to learn human preferences from past experiences [58], and to produce

plans. In Han et al. the planning process is optimized and aligned with the human's preferences through an iterative process, combining imitation learning and reinforced self-training [59]. A combination of symbolic task planning with generative AI is explored in Izquierdo-Badiola et al. as an extension of prior work towards a more intuitive and flexible method for generating plans in human-robot collaboration [47]. Here, LLMs are employed as translators from natural language instructions and human conditions into structured PDDL planning representations enabling the processing of high-level abstract instructions, and the consideration of human preferences and conditions within task allocation.

3.2 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.

3.2.1 Information Flow from AI to User (explainable AI)

A black-box model is a machine learning model whose internal workings and decision-making process are not easily interpretable or understandable by humans. These models take input data and produce outputs without providing explicit explanations of how specific features contribute to the final prediction.

Black-box models, such as deep neural networks and ensemble methods, are often highly complex and rely on intricate mathematical computations, making it difficult to trace or justify their reasoning. This lack of transparency can pose challenges in fields requiring accountability, fairness, and interpretability, such as healthcare, finance, and legal decision-making.

Black-box machine learning models present several significant challenges and shortcomings, primarily due to their lack of transparency and interpretability. Since these models, such as deep neural networks, make predictions without providing clear insight into the underlying decision-making process, it becomes difficult to understand how specific features influence outcomes.

This opacity raises concerns about accountability, particularly in high-stakes fields like healthcare, finance, and criminal justice, where incorrect decisions can have severe consequences. Additionally, black-box models often fail to identify potential biases in data, which can perpetuate or even amplify societal inequalities.

Their complex nature also makes it challenging to detect and fix issues like overfitting or unintended model behavior, resulting in reduced trust from users and stakeholders.

The issues related to black-box models implementation in critical areas (e.g. healthcare, finance) are doubled down by the fact that the end users of these systems have no background in data science, and the proper interpretation of the model decision, as well as assessment of its operational quality is beyond their capabilities.
As a result, there is a growing need for methods that offer interpretability, transparency, and fairness, ensuring that models can be effective, ethically sound and tailored to user needs.
To address the aforementioned challenges and needs, in recent years, a vast amount of research [60] has been devoted to the development of XAI methods whose goal is to bring transparency and interpretability into the decision-making process governed by black-box machine learning models.
The vast number of XAI methods developed, driven by the needs of high-risk areas such as Industry 4.0, medicine, and financial markets, etc., has led to high expectations for their applicability in these domains. These expectations were driven not only by the rapid development of XAI algorithms, but also by regulatory frameworks such as the General Data Protection Regulation (GDPR [61], [62]

While addressing black-box AI problems, it is also important to consider this task from the human-AI collaboration perspective. Decision-making tasks across various applications follow a common process: evidence (e.g., state) is presented, a decision is made from discrete/continuous choices, and follow-up effects typically occur. Within this framework, decision-making becomes a human-AI collaborative scenario, where at least two parties—the human and the AI—contribute to solving the problem. The dynamics of this collaboration can vary, resulting in different interaction models, from simple recommendations to more complex, iterative exchanges [63], [64]. Insights documented in Deliverable 2.1 reveal different preferences of end-users in human-AI collaboration level which aligns with the interaction models defined in the literature [65]. This collaborative scenario suggests bridging the gap between algorithmic decision-making and human expectations through human-centric AI design. Within this design framework, it is important to identify ***what*** information AI should provide its users while ensuring operational transparency. It is also crucial for AI agents to understand their users on ***how*** and ***when*** to best communicate the required information [66].

3.2.1.1 Prototypical and concept-based explanations

Most explanation mechanisms in AI, such as SHAP [67], LIME [68], Grad-CAM [69], Integrated Gradients [70], primarily focus on feature importance, highlighting which input features contribute the most to a model's decision. While these methods provide valuable insights, they are often insufficient for real-world users who require explanations tailored to their level of understanding. Many users, particularly domain experts and decision-makers, need more than just feature attributions—they require interactive, concept-based explanations that align with their mental models and prior knowledge. Concept-based explanations address this gap by linking model behavior to higher-level, human-interpretable

concepts rather than individual features, enabling a more intuitive and user-centric understanding of AI decisions.

The first approach to introducing such a solution for image classification tasks was ProtoPNet [71], which enhances interpretability by learning class-specific prototype representations. These prototypes are learned ante hoc and used to make predictions based on their similarity to the learned representations.

ProtoPNet has several successors, such as ProtoPShare [72], ProtoPool [73], and ProtoTree [74], which further refine and extend the original idea. Building on similar assumptions, concept-based explanations generalize the notion of prototypes to more abstract entities, which can then be used to construct higher-level explanations [75], [76], [77], [78]. Both prototype-based and concept-based approaches are primarily designed for images. However, they do not assume any level of user interaction or allow for personalization in terms of the prototypes or concepts generated.

Multimodal language models have the potential to pave the way for more interactive and understandable explanations. However, they are most often fine-tuned and developed for image and text modalities, leaving behind other types of data that could benefit from similar advancements.

3.2.1.2 Comprehensibility of Explanations

It is increasingly evident that the predominant efforts in explainable artificial intelligence (XAI) are predicated largely on researchers' subjective assumptions about what constitutes an effective explanation, rather than empirically validating its utility for the intended human recipients [79]. Frequently, these explanations prove inadequate or entirely unhelpful to end-users, particularly those lacking specialized domain knowledge or proficiency in data science. This misalignment is not merely a theoretical concern; it carries significant implications for high-stakes domains such as medicine, industrial operations, legal systems, and economic decision-making, where misinterpretations of AI-driven insights can precipitate catastrophic outcomes, including loss of human life or substantial financial damage. A critical limitation of contemporary XAI methodologies lies in their failure to adapt outputs to the diverse, human-centric requirements of their users [80]. This deficiency in comprehensibility and customization has undermined numerous attempts to operationalize AI systems in practical settings. Initial optimism surrounding the potential for machine learning and AI frameworks to deliver transparent, user-accessible insights has been tempered by empirical evidence from real-world deployments and expert critiques. These evaluations consistently demonstrate that, far from enhancing understanding, current XAI approaches often exacerbate confusion among stakeholders. Such observations have been rigorously documented across multiple critical sectors [81], [82], [83], [84], with even the developers of XAI frameworks

acknowledging these shortcomings [85], [86]. This growing body of evidence underscores the necessity for a paradigm shift toward a human-in-the-loop framework. Effective AI systems, including those envisioned as personal assistants, must prioritize rigorous user-centered design, dynamically tailoring explanations to accommodate individual cognitive contexts and limitations. Absent such an approach, the field risks perpetuating the development of sophisticated yet opaque tools that fail to meet the practical needs of human decision-makers in consequential applications.

3.2.1.3 Knowledge Mediation in Explanation Process

There have been works devoted to enhancing understandability and comprehensibility of explanations. Most of them focus on the visual presentation of such results. This includes saliency maps for Deep Neural Networks, task specific visualizations [87] and more general frameworks such as [88], which are still narrowed to only one phase of Data Mining (DM) process and hardly use any domain knowledge to enhance the explanations or interpretability of the models. The interpretation of the results of ML explanation models such as LIME, SHAP or Anchor highly depends on expert knowledge and domain knowledge. However, these frameworks do not provide any means of encoding such knowledge, relying purely on manual examination and interpretation of their results. Many attempts were crafted for the purpose of aiding domain experts or data scientists in the interpretation and incorporation of explanation results into the DM process. In [89], the authors demonstrate how the combination of deep tensor and knowledge graph embedding methods can be used for generating explanations for a model in intrusion detection and genomic medicine that are more human-readable. An interesting approach was given in [90], where the process of explanation was partially formalized and divided into stages that relate to standard CRoss Industry Standard Process (CRISP) methodology and reflect the stages in which the data analysis is performed. The proposed grammar was designed for further conversational-like data and model analysis with the usage of explanations. However, in all of the above examples user interaction was limited and did not allow for flexible knowledge mediation. Most of the approaches are also designed for researchers or data scientists, providing little of utility for lay end-users. This aspect is particularly significant in the domain of personal assistants, where the primary objective is to adapt to the needs and preferences of the individual being supported.

The shift from static explanations to more interactive and better suited for specific user needs explanations is visible in recent research. A growing shift toward systems that empower users to actively engage with AI explanations rather than passively receive them is gaining more attention recently.

Interactive explanations must align with users' evolving needs, which vary depending on the context and specific tasks at hand. However, it rarely addresses modeling users' evolving

mental models, which limits alignment with user thought processes. The range of explanations available in interactive systems is constrained by the types implemented, such as local, global, counterfactual, or anchor-based methods, with only pre-implemented options appearing in interactions. This restricts flexibility[91]. The computational demands of XAI methods can cause delays, disrupting real-time interactions. Furthermore, frameworks require extensive customization for specific domains, hindering scalability [92].

Conversational explanations enable users to understand AI decisions through dialogue, while multimodal explanations use various media to meet diverse needs. The survey of these methods was recently proposed in [93]. The application of such explanations was investigated in different scenarios but mostly focusing on modalities such as images, text, audio or video other modalities including time-series, tabular data, or sequential data are not visibly researched yet.

3.2.1.4 Towards Explainable Sequential Decision Making

Although the research area of XAI has rapidly developed in recent years, the focus of XAI to date has largely been on explaining the input-output mappings of "black box" models such as neural networks, which have been seen as the central problem for the explainability of AI systems. While these models are certainly important, intelligent behavior often extends over time and needs to be explained and understood as such. The challenge of explaining SDM, such as that of robots collaborating with humans or software agents engaged in complex ongoing tasks, has only recently gained attention. We may have AIs that can beat us in Go, but can they teach us how to play? We may have search and rescue robots, but can we effectively communicate with them and coordinate missions in the field?

Initial attempts at making the behavior of SDM algorithms more understandable have recently appeared in different fields such as classical AI planning, reinforcement learning, multi-agent systems, or logic-based argumentation – but often focused on ad hoc solutions to specific problems, with an emphasis on area-specific terminology and concepts developed in isolation from other fields. Many of these approaches are therefore restricted in their scope and applicability. To truly trust an AI agent and collaboratively work with it towards human goals, and to increase successful AI adoption and acceptance in many fields from logistics to robotics, and from smart cities to production planning, we need considerable progress in this new field of XAI for SDM (XSDM).

Under-researched challenges for XSDM include for example: XAI for complex decisions, e.g., on plans or policies instead of single output labels; conversational XAI that continuously interacts with users over time, aiming to understand and support them; contestable and collaborative XAI, which can successfully work with users in areas where neither user nor AI is omniscient nor infallible; and flexible decision-making for XAI, able to adapt to users, respect their autonomy, and go beyond one-size-fits-all explanations. These are among the challenges tackled by the WP3 researchers in the PEER project.

In *September 2024*, we organized a Dagstuhl research seminar[1] on the topic of "Explainable Sequential Decision Making", bringing together 27 participants from 10 countries, including academic researchers and industry experts from communities such as RL, planning, game AI, robotics, and cognitive science. This interdisciplinary group discussed their work on explainability in sequential decision-making contexts, and aimed to move towards a shared understanding of the field and a common roadmap for moving it forward. One of the outcomes of the seminar was taking stock of the field and developing a taxonomy of XSDM.

SDM involves an agent making a series of related decisions over time, where each decision can be influenced by earlier decisions (i.e., a history), current observations and/or projected future states, to optimize some goal, objective or reward. What sets SDM apart from the usual settings considered by XAI, is (a) its temporality - as in the dynamics of the environment itself, leading to unique features such as trajectories, key events, and forward and backward looking aspects of decision-making; (b) goals – involving temporally extended intent, temporally extended goals, subgoals, and consistency as the policy unfolds; (c) complexity – SDM has the usual issues of XAI but with the increased complexity of decision sequences and policies; and (d) exploration – there are specific issues around credit assignment, and one cannot easily sample anywhere in the state space: there is no fixed training set as in supervised learning.

This leads us to a first sketch of a proposed taxonomy of XSDM that covers the following questions:

1. What might need to be explained in sequential decision making? What are the targets for the explanations generated?
2. How might sequential decision making be explained? How will an explanation module compute explanations for SDM? What are the inputs and outputs to this module, and what does it need to take into account to engender understanding and trust in users?
3. What metrics might be used to assess the efficacy of the explanations generated by any XAI system explaining SDM?

The proposed answers to these questions can be useful to structure the work described below, as well as the work envisioned in the future for the PEER project. Part of the answer to 1., for example, is that "every aspect of a decision sequence (or abstractions of those aspects) can be the focus of an explanation whether it be the decision outcome(s), its environment state or states, its action or actions, its stated goals, choices or objectives with respect to the past, present or future; or with respect to beliefs about or comparisons to any aspects of other decision sequences. This explanation may also address the processes/factors that led to its generation. In particular, explanations in sequential decision-making can focus on different levels of "decision outcomes": on an individual action decision; on a sequence of decisions or a plan; or on an entire policy, i.e. the full set of available behaviors of an AI agent. In the following, we will outline some of the literature

[1] https://www.dagstuhl.de/en/seminars/dagstuhl-seminars

and our ongoing research first on the topic of explaining individual actions – those of search algorithms, and those of more general reinforcement learning agents – and then on generating entire interpretable policies.

3.2.1.5 Explaining Actions of Search Algorithms

Our work in this category focuses on explaining why a given agent driven by a search algorithm such as MCTS made a certain decision.

Our research on explainable search, building on prior work published before PEER project [94], began with the realization that much of explainable RL research at the time fell into one of two main categories: Either it attempted to explain an agent's behavior in its entirety, i.e. tried to simplify, characterize, or summarize a complete and fully optimized agent policy [95]; or it focused on explaining a single agent decision "in a vacuum" instead, i.e. tried to illustrate how and why a given policy maps a specific state input to a specific action output [96]. This is largely still the case in the literature [97]. In many application scenarios however, including those tackled in PEER, a *high-level explanation of an entire policy* is likely to be either too abstract or too verbose to be helpful. Imagine a chess tutor agent trying to explain to a human student how it plays chess in general, instead of explaining its next move recommendation to increase pressure on the opponent's rook; or a search & rescue robot explaining to a human mission supervisor the general structure of its decision making process, instead of outlining its plan for getting access to the first floor of the partially collapsed building it is in. A *myopic explanation of a single decision*, without explicitly discussing the future situations and further decisions it might lead to, would not give the user enough relevant information. For example, a chess tutor merely stating that the upper left corner of the board contained the most relevant input features for a move does not explain the reasoning behind the decision Similarly, a the search & rescue robot saying that moving left leads to an 11% higher probability of clearing the first floor within ten minutes provides little practical insight without further context. Humans would probably want to know: Why? What exactly could happen next? Which possible outcomes were explored, how were they interpreted, compared, and selected from?

Learning complete high-quality policies for the entire state space in complex domains can be very challenging. In cases where domain models are either available or can be learned, the most promising RL approach is often online planning, or the repeated *search* for the best next action, starting from the current state in which a decision is needed, and exploring possible futures in the immediate future of the agent.

Some of the most successful RL approaches at the heart of recent AI breakthroughs have been powered by search [31], often using variants of MCTS. Sampling-based search algorithms such as MCTS tackle huge search spaces through selective search, provably converge to optimal behavior in many settings, provide approximate solutions when stopped at any time, and also proved to show excellent synergy with integrated deep neural networks [31]. They are now being developed for and applied to countless domains from board and video games to robotics to drug discovery to self-driving cars and more [98]. However, the decision-making and behavior of MCTS can be hard to understand, as it results from often

large and complex search trees, consisting of thousands of simulated possible futures, their evaluations, and relationships to each other. This can be a challenge e.g. for developers trying to debug and improve MCTS; for end users who need to build trust in and successfully collaborate with MCTS-based agents to deploy solutions in practice; and for the development of completely new applications, such as turning algorithms that can play chess on a superhuman level into helpful and effective chess teachers. We are seeing the explanation of search decisions as one of the central challenges of PEER. Our ongoing work on explainable search has developed into three main directions so far.

1. Explaining search with Computation Tree Logic (CTL) [99]**.** In this work, we focus on dynamic vehicle routing in public transportation (which has potential applicability to explainable routing in both the Amsterdam and the Sonae use cases of PEER). Algorithmic and data-driven approaches have shown particularly promising results in public transportation [100].However, transportation has traditionally involved human operators optimizing decisions manually. As a result, AI-based approaches are often viewed with skepticism even when they demonstrate higher efficiency. This lack of trust primarily stems from a lack of understanding.

Path planning algorithm can for example be responsible for assigning a passenger request to one of several available vehicles. The algorithm is equipped with real-time information, including the current locations of all vehicles and their availability status. While generating plans, the algorithm is guided by specific objectives, such as maximizing the overall percentage of completed trips. Moreover, it is provided with certain non-negotiable constraints on time windows that must not be violated during the planning process.

Leveraging its internal mechanisms, the planning algorithm generates the best approximation to an optimal solution it can find and presents the user with the corresponding result. However, the algorithm is a black-box when presented to users such as route dispatchers who may not have a technical background in search algorithms. They are unable to analyze the consequences of solutions in detail, and are therefore unable to determine if they can trust those solutions or if they can improve them with their domain knowledge. We tackle this challenge by developing computation tree logic (CTL)-based explainable MCTS. CTL is promising for us as a formal language with which we can formulate a variety of queries related to search trees, and find provably correct answers to them by applying off-the-shelf CTL solvers.

Our explainer categorizes user queries into three varieties: *factual queries* that provide insights into the algorithm's decisions derived from states and actions in the search tree, *contrastive queries* that compare user-suggested decisions with the algorithm's recommendations by identifying CTL violations or inferior rewards, and *alternative plan explanations* that involve extending the search to additional (user-requested) options.

Specifically, our explainer first leverages language templates to capture user queries about the outcomes of the MCTS algorithm, such as “Based on the suggested vehicle assignment, is it expected that the passenger will be dropped off too late?” Second, each templated natural language query is converted into a logic specification by a many-to-many mapping process, in this case $\phi = \mathrm{AG}\,(t_{\mathrm{est.}} > t_{\mathrm{allowed}})$. Third, our explainer triggers a formal specification checking mechanism of CTL that takes logic formulas as input and traverses the MCTS tree

to generate results. Fourth, the checking and verification results obtained in this step are then translated back into natural language. Continuing from the previous example, a response to the query would be: “The passenger has specified a desired drop-off time of $5{:}33\ PM$. The route planning algorithm has simulated approximately 150 potential future scenarios and requests. Potential Late Arrival: There's a chance that the passenger might encounter a delay in their drop-off time. This expected delay averages around 23 minutes. The primary reason for this delay is that the proposed vehicle is expected to make stops at about 4 other locations prior to reaching the passenger's drop-off point. However, the delay can be as short as 19 minutes or extend up to 27 minutes. The percentage of times the suggested vehicle doesn't meet the desired drop-off time is about 10%.” The question and answer templates are domain-dependent and can be adapted on the fly.

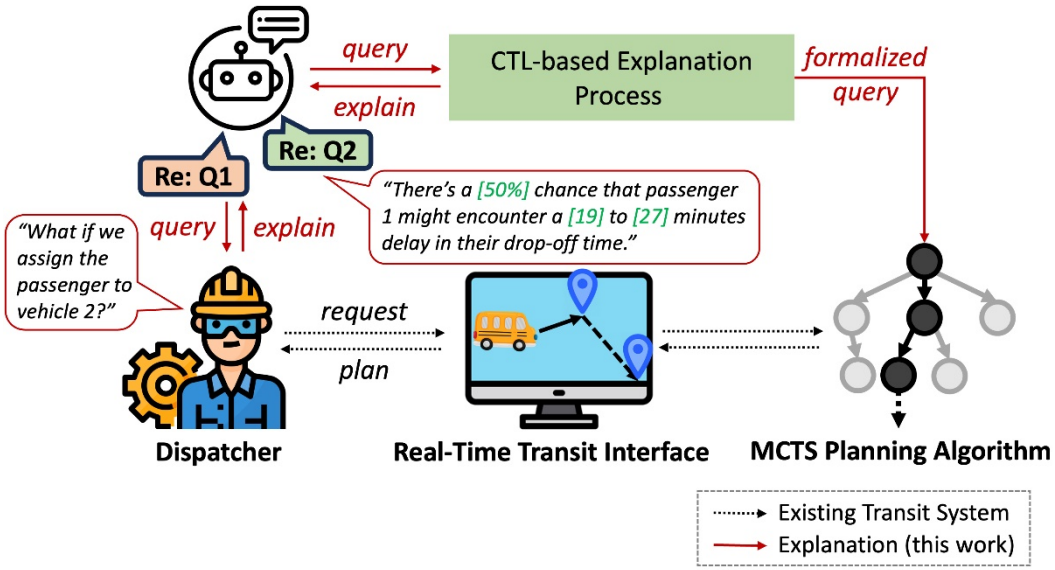


Figure 7. Explaining search with Computation Tree Logic (CTL)

This work is currently being extended [101] to improve its flexibility with regard to the user queries it can process. Instead of the user having to choose a question template, we are testing a free-form query interface. Queries are then mapped by an LLM to a question bank that has been created on the basis of an earlier user survey. In this way, natural language inquiries submitted via a chat interface are turned into parameterized logic expressions. This also further increases the generality of the approach, simplifying its potential application to the PEER use cases.

2. Explaining MCTS in terms of user-understandable scenarios. In earlier work [94], we developed the first concrete building blocks of MCTS explanations that can be expanded and adapted for a variety of explanation scenarios. This explanation toolset drew on the search tree over expected futures available to MCTS. It aimed at explaining current decisions by explicitly discussing possible future situations and further decisions that given choices could lead to. Due to the complexity of search trees and the multitude of questions a human user could have about the future plans of MCTS, our toolset made the first steps towards understanding MCTS explanations as interactive conversations between user and AI agent,

rather than giving one-size-fits-all explanations. And due to the nature of online planning settings, in which often neither the AI nor the user can consistently make optimal decisions on their own, our toolset suggested first, simple ways to "discuss" MCTS decisions before committing to them -- aiming at explanations that (in limited ways) support true human-AI collaboration.

The information we presented on any given search tree or subtree of interest was defined by the probabilities of a pre-specified, domain-dependent set of positive and negative *scenarios* (sets of states), as defined by the proportion of MCTS simulations in the subtree that reached them. The scenarios were assumed to be shared knowledge among AI and human user. Capturing an opponent piece in chess, for example, can be considered a commonly understood positive scenario for the user.

Using sets of preferred states, chosen such that "users understand that this is a good thing" [102], is a relatively common method in XAI to ground explanations; it has been argued that users need to be given explanations using domain-like language [102], as opposed to the expected action values and future rewards used internally by many RL algorithms. This is also a matter of the right level of abstraction: When using higher-level concepts such as scenarios, "these concepts can be designed to be more descriptive and informative for the potential user"[103].

In our current work, we are exploring the automatic learning of such scenarios, for example by extracting them from conversations with users with as little effort as possible [104]. Furthermore, we tackle the limitation of our previous approach that it could only explain actions in terms of positive or negative scenarios actually found in the search tree – by learning predictors for our *future expectations* over these scenarios, we hope to be able to provide explanations with a deeper temporal horizon (with appropriate communication of prediction uncertainty [105]).

3. Explaining MCTS in terms of a minimally sufficient set of state or evaluation features. Early work on explainable RL also attempted to explain actions with the agent's expectations over a set of positive or negative scenarios; however, the emphasis here was to derive explanations that are both *sufficient* and *minimal*. An explanation was here defined as a set of scenarios with the expected frequencies of reaching them, assuming the action in question is chosen - for example, "After choosing this action, we will capture the opponent's rook with 27% probability". An explanation was defined as sufficient if the frequency of reaching the selected scenarios is enough to show that the action in question is optimal (without needing to mention additional scenarios – the frequency of all other scenarios is irrelevant for choosing the action in question). A sufficient explanation was defined as minimal if it includes the minimum number of scenarios needed to ensure it is sufficient. In our current work in this area, we are aiming at transferring these concepts to search-based agents such as MCTS agents.

3.2.1.6 Explaining Actions of RL Agents

Our work in this category focuses on explaining why a given RL agent (not necessarily guided by a search algorithm, but also a more commonly used reactive agent based on e.g. a neural network) made a given decision.
We are taking inspiration here from a popular approach to explaining the decisions of supervised learning models: *Concept Bottleneck Models (CBMs[77]*). These models try to make their decisions (typically classifications of input images) understandable for human users by not mapping the input image directly to an output decision but first predicting an intermediate set of human-specified concepts c, and then using c to predict the final output. By forcing the final decision to depend on the set of concepts specified by a human user (at training time), CBMs don't only enable easier interpretation, but also allow users e.g. to correct model mistakes at test time by directly intervening on incorrectly recognized concepts.
A rich literature is based on this basic idea; for example, *Post-hoc Concept Bottleneck Models (PCBMs)*[106] attempt to overcome two limitations of CBMs: 1) requiring dense concept annotations in training data to learn the bottleneck, and 2) often reduced accuracy in practice due to users not specifying powerful enough concepts in advance. The idea of PCBMs is to be applicable to any pre-trained network, and converting it into a concept bottleneck model, enhancing it with the desired interpretability benefits post-hoc. This is done by learning the concept activation vectors [107] of the concepts of interest in the activation space of the network; projecting the embeddings produced by the network for a given input onto the concept subspace defined by this set of vectors; and then training an interpretable predictor to classify examples from their projections. When the user-specified set of concepts is insufficient, PCBMs can also be extended with a residual predictor that handles the "unexplainable" part of a decision which cannot be explained with the given concepts.
The recently proposed *Explanation Bottleneck Models (XBMs)* [108] go one step further by dispensing with user-defined concepts entirely. XBMs generate a text explanation from the input by leveraging pre-trained foundation models (vision-language encoder-decoder models), and then predict a final task prediction based on the generated explanation. By end-to-end training of this pipeline with suitable regularization, they ensure target task performance without losing explanation quality (without allowing the intermediate text to become meaningless to humans).
In our current work in this area, we explore the transfer of these ideas to explaining the behavior of – and intervening/editing the behavior of - RL agents. Such agents could help steer planning algorithms in several of the PEER use cases, or replace planning in cases where decision times are short.

3.2.1.7 Explaining entire policies of RL agents

Our work in this category focuses on making entire policies of agents understandable. We focus here on the growing research area that uses *programmatic representations* to encode solutions to sequential decision-making problems [109]. Depending on the language used, programs that encode programmatic solutions can generalize to unseen scenarios better than

neural representations [110]. Such solutions can also be interpretable [111], which could allow users to manually modify computer-generated solutions. The challenge of using programmatic representations is that one needs to search in large and discontinuous spaces of programs, typically defined by a domain-specific language.

In our first work on programmatic reinforcement learning [112], we leverage the ability of large language models to write computer programs to speed up the synthesis of programmatic policies. We use an LLM to provide initial candidates for the policy, which are then improved by local search. LLMs are especially effective in this setting when we give them access to the outcomes of the policies' rollouts. That way, LLMs can try policies encoding different behaviors, once they observe what a previous policy has accomplished. This process forces the search to explore different parts of the space through "exploratory initial programs". Since our system only queries the LLM in the first step of the search, it offers an economical method for using LLMs to guide the synthesis of policies.

In ongoing work, we are extending this system to leverage human knowledge encoded in foundation models even more effectively. We extract programmatic policies from LLMs that encode "innate skills" in the form of temporally extended actions, or options. In contrast to existing approaches to learning options, we thus learn them from the general human knowledge encoded in foundation models in a zero-shot setting, and not from the knowledge the agent gains by interacting with the environment. Then, we search for a programmatic policy by combining the programs encoding these options into larger and more complex programs. We hypothesize that this way of learning and using options could further improve the sampling efficiency of current methods for learning programmatic policies.

Additionally, we are working on: 1) a literature review on programmatic reinforcement learning; 2) a combination of gradient-free (e.g. local search, evolutionary algorithms) with gradient-based techniques (policy gradient methods), to learn more from the trajectories generated by programmatic RL approaches and so improve their sample efficiency; and 3) the application of programmatic RL to domains in which explainability is seen as key, leading to suboptimal heuristics being used in practice instead of strong, black-box RL agents. This includes e.g. scheduling problems but also warehousing/logistics problems similar to the Datacation use case of PEER.

3.2.1.8 Transparent Hyperparameter Tunning

With particular application to the Sonae use case, we examined a collection of state-of-the-art frameworks for hyperparameter optimization that incorporate aspects of human-in-the-loop methodologies and extend optimization beyond conventional model performance to include external factors such as user goals, preferences, and time constraints.

ixAutoML [113] exemplifies a system with native interactive and explainable capabilities, allowing users to directly inspect and adjust the automated learning process.

In contrast, frameworks such as Optuna [114], Ray Tune [115], FLAML [116], Hyperopt [117], BOHB [118], Auto-WEKA [119], SMAC [120], and Spearmint [121] leverage sophisticated surrogate models and Bayesian techniques to explore hyperparameter spaces; however, their

capabilities for explainability and human-in-the-loop integration are primarily realized through post hoc analysis and external tool integration rather than being provided natively.

Overall, these frameworks demonstrate a progressive shift from purely automated hyperparameter tuning toward systems that can incorporate human expertise into the optimization process. They enable experts to introduce external criteria - such as operational constraints, user preferences, and computational time limits - into the optimization workflow, thus extending beyond standard ML model performance metrics. Although most frameworks (with the notable exception of ixAutoML) require external post hoc analyses for comprehensive interpretability, they nonetheless offer flexible environments that can be augmented to address non-trivial real-world optimization challenges. This evolution represents a significant advancement in the development of adaptive, user-centric optimization systems in machine learning. However, it still raises questions about the effective utilization of these frameworks in iterative, human-in-the-loop scenarios with varying user preferences.

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

As humans are different and are not rationaldue to cognitive and emotional limitations of information processing [122], [123]. Preference modelling and elicitation are task-specific and domain-specific, as we are looking for relevant features of the modeled domain that differentiate various people/groups of people. So, in the remainder of this section we will focus on the case of recommender systems, the improvement and expansion of which is part of the current research in the project. This thread will be more comprehensively covered in the *MVP#2* and reported in the *D3.2 report*, however preliminary research was already done for Sonae use case.

3.2.2.1 Recommender systems

Recommendation systems use different methods to suggest products, services, or content to users according to their behaviors or preferences. Several approaches have been created over the years, from the initial **Collaborative Filtering (CF)** techniques to more complex ML-based approaches like deep learning and factorization models. This section delves into the most important approaches and state-of-the-art in recommendation systems, highlighting their benefits and challenges which we plan to tackle or adapt for *MVP#2* and report in *D3.2*.

Content-Based Filtering (CBF): CBF is a popular recommendation method that relies on analyzing the characteristics of items to recommend similar ones to the user. Unlike collaborative filtering, which focuses on user behavior, content-based methods assess the content (e.g., textual data, product features, metadata) to make predictions about what a user might like. This technique uses attributes of items, such as keywords, descriptions, and tags, and matches them with the user's preferences based on their past interactions [124].

For example, a movie recommendation system might suggest films based on the genres, directors, or actors that a user has previously watched and rated highly. Content-based filtering has the advantage of not requiring user interactions with other users, making it a useful solution for handling the cold start problem for new users or items. However, it also faces challenges such as limited diversity in recommendations, as the system tends to recommend items that are very similar to the ones a user has already interacted with [125].

Collaborative Filtering: CF is one of the earliest and most widely used techniques for recommendation. It operates by predicting a user's preference based on the preferences of other similar users. Two main types of CF exist: memory-based and model-based methods [126].

Memory-based methods rely on user-item interactions to compute similarity scores between users or items. These methods use a user-item matrix to recommend items based on the preferences of users with similar behaviors. For instance, k-nearest neighbors (k-NN) is a classical approach where similarity between users is computed using metrics such as Pearson correlation [126].

Model-based CF algorithms attempt to obtain latent factors of user tastes. These models decompose the user-item interaction matrix into lower-rank matrices representing items and users and make predictions based on such latent factors. Algorithms based on Singular Value Decomposition (SVD) have been found to cope with implicit feedback like clicks and purchases well. Data sparsity, where most of the entries of the user-item matrix are missing, is a fundamental issue of CF, especially for implicit feedback. Recent advances, such as Bayesian Personalized Ranking (BPR), address this by learning models for personalized ranking instead of explicit ratings prediction [127].

In recent years, more advanced techniques such as deep learning have made significant progress in the field of recommendation systems. Deep learning offers advantages in modeling complex non-linear relationships within user-item interactions, improving recommendation quality and robustness [128].

A notable contribution in deep learning for recommender systems is Neural Collaborative Filtering (NCF), which replaces traditional matrix factorization's inner product with a multi-layer perceptron (MLP) to learn the user-item interaction function [129]. This approach allows the model to capture more complex relationships between users and items compared to traditional methods.

Moreover, in session-based recommendation, where no historical data is available, Recurrent Neural Networks (RNNs) have demonstrated impressive results. RNNs model the temporal sequence of interactions in a session, offering significant improvements over classical collaborative filtering methods. By utilizing Top-k ranking loss functions, RNN-based systems have improved recommendation accuracy by up to 35%, particularly in e-commerce and video platforms [130].

Factorization Models: Factorization models are particularly effective in collaborative filtering tasks where data sparsity is a significant issue. These models factorize the user-item matrix into low-rank matrices, capturing latent interactions between users and items [131].

Factorization Machines (FM) generalize matrix factorization models, allowing for the modeling of interactions between features [132]. These models have been shown to handle large, sparse datasets efficiently, outperforming traditional methods like Support Vector Machines (SVMs) in recommender systems. FMs offer the flexibility to incorporate both implicit and explicit feedback, making them versatile for a wide range of recommendation tasks.

Most of the currently developed and researched recommender systems are focused on hybrid models that are combinations of two or more methods.
K-GBDT combines the strengths of two key components: k-nearest neighbors and Gradient Boosting Decision Trees (GBDT) to enhance accuracy and provide personalized recommendations [133]. k-NN algorithm identifies and gathers data from users and items that are similar to the target user and item. It serves as an input to the GBDT model, which captures complex patterns and interactions within the data. It iteratively learns and finds intricate patterns, boosting the predictive performance by focusing on hard-to-predict instances. Proposed hybrid combination results in capturing both local and global patterns. k-NN focuses on local similarities, while GBDT reveals global patterns, so that the model benefits from a comprehensive understanding of user-item interactions. Another advantage of the presented algorithm is that while GBDT handles the feature interactions effectively, k-NN helps in scenarios where data is sparse by relying on neighborhood information.
Another novel approach integrates Factorization Machines with Association Rules (AR) into a hybrid approach – FMAR. In the first step, frequent itemsets revealing products often purchased together are identified. Then, the pool of items for which the FM model must predict ratings is narrowed down by utilization of derived association rules. By focusing on a subset of relevant items, the computational load is reduced, leading to faster recommendation generation while maintaining accuracy comparable to traditional FM-based systems. This idea offers a promising solution for real-time recommendation scenarios, where both speed and accuracy are critical, like the ones covered in PEER project.

The advent of Graph Neural Networks (GNNs), such as Graph Convolutional Networks (GCNs), which leverage the inherent graph structure, have provided new opportunities for recommender systems' development. By viewing the user-item preference data as a bipartite and undirected graph, GCN-based techniques have shown the potential to mine near- and long-distance preferences of users towards items. Differently from the classical CF models, which utilize the information from user-item interactions only in the optimization of their objective functions, graph convolution is designed to encapsulate user-item multi-hop relationships right from the representation of their embeddings. Such learned profiles are suitably exploited to produce more accurate recommendations [134].
Another idea is integrating GCNs with attention mechanism like in GARec. It employs an attention mechanism alongside a spatial GCN to assess the significance of neighboring nodes in a user-item bipartite graph. This setup allows the model to weigh the influence of related users and items appropriately when forming the final representation of a target entity. GARec's performance outperforms existing model-based methods, including both non-graph

neural networks and other graph neural network approaches, in terms of Root Mean Square Error (RMSE) [135].
Graphs can represent not only user-item interactions, but we can also incorporate personalized features into graph-based recommendation system by augmenting the node features vectors with additional user-specific data such as demographics. For example, instead of initializing user nodes with only random embeddings, we can concatenate additional features (e.g., age, gender, location) to the initial embedding. This way, the GNN will learn representations that combine both the graph structure and personalized user data.

Graph-based methods are also useful in the context of recurring item prediction. They have been applied to construct item-user networks that capture historical recurrence trends, enabling better long-term user engagement modeling [136]. Self-attention mechanisms, as implemented in transformer-based models, improve recurring item prediction by capturing complex long-range dependencies in user sessions [137]. Real-time adaptation remains an open challenge, with incremental learning techniques being explored to dynamically adjust recommendation weights without requiring full model retraining. Reinforcement learning-based approaches have also been investigated for optimizing recurring recommendations through real-time user interactions.

While recommendation systems have made significant progress, there are still a few key challenges to address:

- **Data Sparsity**: A common issue with CF is that it struggles when there isn't enough data. This can reduce the accuracy of recommendations. However, techniques like BPR and Factorization Machines (FM) are designed to solve this problem by focusing on ranking items instead of trying to predict specific ratings [132].
- **Cold Start Problem**: When new users or items are introduced, there isn't enough data to make informed recommendations. Hybrid models and content-based filtering methods can help here by using additional sources of information to better handle new situations [126].
- **Scalability and Real-Time Processing**: As the amount of data increases, making sure that recommendation systems can still handle it efficiently becomes crucial. Deep learning models and FM have the advantage of being scalable, helping them to manage and process large datasets in real-time [130].
- **Explainability and Fairness**: As recommendation systems become more complex, there is a growing demand for transparency. Users and developers want to understand how recommendations are made. Research is moving towards making these systems more explainable, which could boost user trust by providing clear explanations for why certain recommendations are given [138].

- **Preference Modelling and Elicitation:** Preference modeling plays a key role in modern recommendation systems. Review-based models extract user preferences from textual data, with neural attention mechanisms improving personalized ranking and recurring item recommendations. Sentiment-aware preference models further refine user preference understanding by dynamically adjusting recommendations.
  LLM-driven preference modeling has introduced generative retrieval frameworks that learn evolving user behaviors over time, enhancing long-term personalization. Hybrid preference models combine explicit feedback (e.g., ratings, preferences) and implicit signals (e.g., clicks, dwell time, past purchases) to improve adaptability to user behavior shifts. Multi-modal preference modeling, leveraging interaction logs, session history, and time-based features, further refines recommendations by detecting long-term user interest patterns.
- **Context-awareness and Personalization:** Future research will also focus on interactive preference modeling, where users can provide real-time feedback to refine recommendations dynamically. Context-aware recommendations, incorporating external factors such as time of day, location, and historical consumption cycles, will be further developed to enhance personalization accuracy.

Recommendation systems have evolved from basic CF methods to sophisticated deep learning and factorization techniques. With data becoming more complex and heterogeneous, recommendation systems continue to evolve with new algorithms being introduced to improve accuracy, scalability, and user satisfaction. The future of research will be focused on addressing existing issues such as data sparsity, the cold start problem, and the need for more explainable and personalized recommendation systems.

In our ongoing work, we are evaluating all of the above-mentioned state-of-the-art approaches for their suitability in the Sonae use case. After preparing the basic models, as next steps we plan to: (a) test various hybrid approaches and diverse GNNs architectures, (b) evaluate supervised algorithms after determining the target feature which must be some implicit rating, (c) address the main challenges mentioned above, (d) transfer the knowledge gathered with experiments on Sonae data to other use cases.

3.2.2.2 Human-in-The-Loop Learning

Human-in-the-loop learning is a set of strategies combining human and computational agents (e.g., an AI model). The goal is typically to (1) increase the accuracy of the agent's performance acting on behalf of the user (e.g., planner), (2) combine human and AI agent's knowledge to reach the maximum accuracy faster, or (3) assist the user in doing an ML task efficiently.

Within our SDM framework (S, A, T, R, γ), we have thus far assumed that the reward function $R(s, a)$ is explicitly defined. However, in many real-world scenarios, particularly when attempting to replicate human or animal behavior, this assumption breaks down. Inverse Reinforcement Learning (IRL) addresses this challenge by attempting to recover the underlying reward function from observed behavior. The fundamental premise of IRL stems from human-in-the-loop learning: while we can often observe expert demonstrations of desired behavior (whether from humans, animals, or existing systems), articulating the precise reward function that drives these behaviors is notably challenging, especially when the reward function is multi-attribute (how can we determine the relative weights a priori?). Even with extensive behavioral studies and neurophysiological evidence, extracting accurate reward functions through empirical investigation remains difficult. IRL can be formalized as the inverse problem of RL: given observed optimal (or near-optimal) behavior of the experts following some policy π, we aim to infer the reward function R that makes this behavior optimal. This becomes particularly complex in scenarios with multiple competing objectives, where the observed behavior might result from balancing various factors - a situation common in human decision-making [139]. Not only this, IRL requires to solve a RL problem at every gradient step other than performing standard backpropagation. To try and mitigate the problems of an high computational challenge and the scalability issue that IRL on big data can introduce, Receding Horizon Inverse Planning (RHIP) was formalized t introduces not only a brand new way of initializing the backward pass closer to the desired solution by leveraging the value of dominant eigenvectors (MaxEnt++) but also a novel IRL algorithm for path planning that balances its load through a trade-off among cheap, deterministic planners and expensive yet robust stochastic policies [140]. While algorithms and frameworks like RHIP were able to successfully extract latent preferences in path planning scenarios, this was only done in single-objective tasks. However, in path planning for humans, the agents often differ in their objectives, especially when considering factors like disabilities. Implementing MORL techniques within the RHIP could significantly enhance its capabilities for personalized path planning. As users select certain types of paths or deviate from suggested plans, the system could build personalized preference models that predict which trade-offs a particular user might prefer in new situations. This approach would be particularly valuable for users with disabilities, as their path planning needs often involve complex trade-offs that single-objective systems struggle to accommodate. For example, a user with mobility impairments might sometimes prioritize the shortest accessible path, while at other times preferring a slightly longer route with fewer elevation changes.

Another way of human-machine collaboration that emerges from AI is Active Learning (AL) [141],[142]. It draws inspiration from a set of instructional methods in educational literature [143] that aim to engage students as active participants in the learning process, reducing their reliance on the teacher.

AL focuses on identifying data samples that help ML algorithms improve in achieving a particular objective. It often requires having human-in-the-loop and user involvement, such as labeling selected data points, to facilitate the learning process. This approach often helps

supervised learning models to converge to desired accuracy with more labeled data. AL assumes that the learning agent (AI) has full control of data and can query an entity with extensive knowledge of the domain (e.g., typically an expert) for data annotation [144].

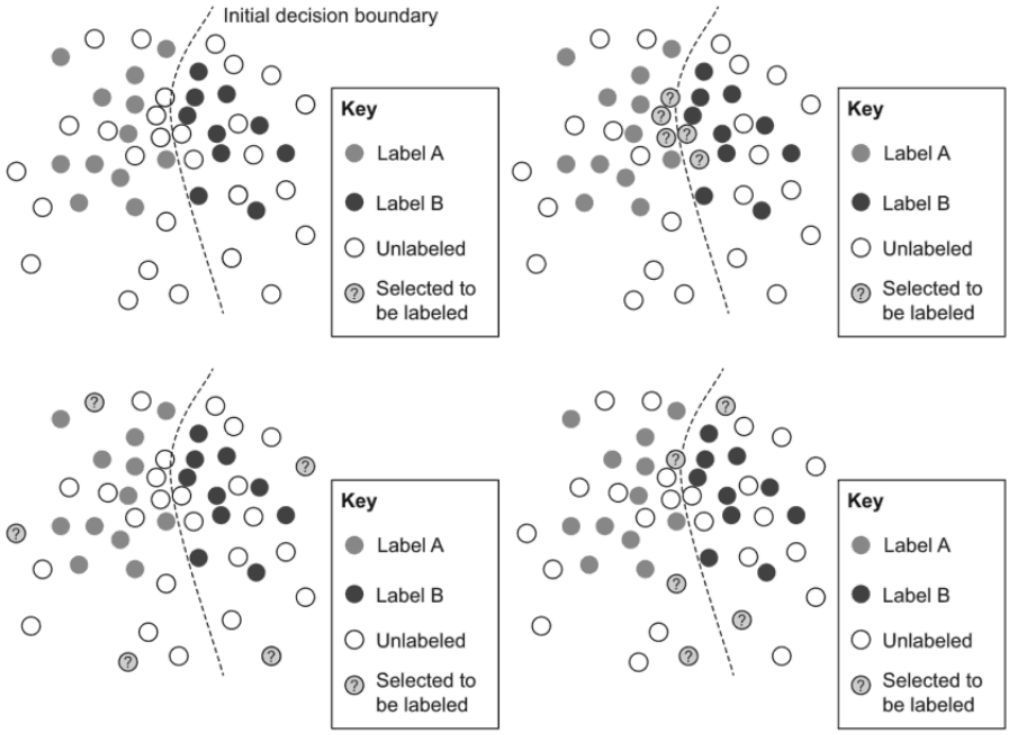


Figure 8. Different Active Learning strategies. Top left: initial decision boundary of some ML algorithm. Top right: Uncertainty sampling. Bottom left: Diversity sampling. Bottom right: combination of diversity sampling and uncertainty sampling strategies [145]

Explained visually in Figure 8, Munro recently identified three types of sampling strategies: random, uncertainty, and diversity [145]. The random sampling approach is the most straightforward, as it involves selecting data for labeling at random. In contrast, the other two strategies are more intriguing, as they reflect a familiar challenge—balancing exploitation and exploration [146]. As shown in Figure 8-top right, in uncertainty sampling (exploitation), the learner query instances which have the least certainty in prediction and diversity sampling (exploration) on the other hand selects data points that have been seen in the batch of data in previous iterations (see Figure 8-bottom left). There are as well AL strategies that try combine the benefits of the two approaches (see Figure 8-bottom right). We refer the reader for more details to a recent publication [144].

Although ML techniques has shown a very successful result in Anomaly Detection (AD) tasks in visual inspection (e.g., Proditec use case), these methods focus mostly on one-class learning, where nominal data are used to train a representation of normal cases, and deviations from this representation are considered anomalies.

However, most of the research does not consider the practical aspect of implementing such a system. In real-life examples, data labeling is inevitable to obtain the nominal samples but also to obtain the anomalous samples for the purpose of validating the quality of the model. Thus, the real effort that is not yet fully explored should be placed for active-anomaly detection that will combine the strengths of unsupervised methods with active learning for

the sake of dataset creation. Another aspect of enabling such a paradigm in visual inspection tasks is to incorporate human knowledge to improve model performance from human annotated samples. This knowledge/preference not only varies subjectively but also objectively due to regulations and restrictions applied to different regions and countries during the quality assurance process. Therefore, we argue that actively involving human operators within the anomaly inspection process not only decreases annotation effort but also increases model adaptability by considering operator feedback during validation. In PEER, we plan to study an AL setup for inspecting visual anomalies in industrial settings, enabling a two-way interaction between domain experts and the anomaly modeling process. This collaborative approach enables an iterative thresholding mechanism that refines anomaly detection over time. Furthermore, by incorporating annotated anomalous samples in AD pipeline, we ensure that such information would still be beneficial for One-Class Classification (OCC) improvement. Additionally, we hypothesize that our approach mitigates the costs associated with manual data augmentation and data creation tasks for model evaluation, ultimately leading to a more robust data representation during model validation. Moreover, as shown by some recent works, combining the benefits of both AI paradigm, namely XAI and AL would enable a fruitful dialogue between human and AI and we would like to study how different XAI methods would enhance both user's and learning agent's performance in visual anomaly detection domain.

3.3 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.).

3.3.1 Human-in-The-Loop Approaches

To deliver truly personalized experience [147], humans should play an active role in the system, providing feedback that adjusts the system since tasks of modelling human agents exhibit complexity arching beyond current capabilities of algorithms alone [148]. Approaches aiming to solve problems that include human feedback as a part of a system can be subsumed under an umbrella term of Human in The Loop (HiTL) [149].

The general HiTL approach can be considered as a three-step process [150]: (a) *Data acquisition:* direct and indirect feedback from the user, physiological data indirectly describing

central nervous system activity, information about the person's current activity, and a description of the external environment; (b) *State inference:* based on the acquired data, the system tries to infer the user's current state; (c) *Actuation:* when knowledge has been distilled from the gathered data, a system should decide on the action to be taken.

In fact, the situation in which both the AI and the user are in the same loop can be considered an interaction of two loops that are similar at a very high level of abstraction [150]:

- An AI-centered loop in which artificial intelligence attempts to obtain data about the user, tries to understand them and adjust its behavior accordingly,
- A user-centered loop in which the user receives data from the AI, tries to understand it, and takes the next steps to achieve their goal (e.g., asks the AI for explanation).

Various HiTL approaches differ in terms of which particular user characteristic is inferred, and this in turn entails acquiring the relevant data in the *Data acquisition* step. In affective computing (*emotion-in-the-loop*), the focus is on the user's emotions, so the data collected – such as the heart rate signal – directly or indirectly helps infer the current emotional state [151]. In *personality-in-the-loop*, the focus is on personality, so behavioral clues and questionnaire data are collected that help infer personality [150]. The scenarios considered in the project can be called *mental-model-in-the-loop* because, on the one hand, the user is trying to understand AI, i.e., create a mental model of the AI assistant and understand its decisions, and, on the other hand, AI is trying to make a mental model of the user trying to gather information about the user's preferences and expected levels of detail of explanations.

HiTL approaches can be considered a generic framework within which a human-AI interaction is formed However, in this approach the focus is on achieving the goals of one or the other side, as it is an interaction of two similar feedback loops. True collaboration between human and AI will require dedicated frameworks that assume close cooperation between the two parties from the very beginning. One such approach was recently presented in [152]. As part of the current work, we are continuing our review in search of promising approaches to this challenge.

3.3.2 Conversation Management

Conversation management in AI refers to the strategies and techniques used to ensure coherent, engaging, and contextually relevant interactions between an AI system and a human user. The goal is to simulate natural conversation, maintain user engagement, and provide useful responses. Currently, applications of conversation management – known to the broader audience as intelligent assistants or chatbots – span across different disciplines, including e-commerce [153], healthcare assistants [154], and financial advisors [155].

3.3.2.1 Key Aspects of AI Conversation Management

We can identify following key components in AI conversation management:

- **Intent recognition:** identifying the user's intent (e.g. request, complaint, expressing emotion) [153].
- **Context awareness and memory:** AI should remember prior parts of the conversation to avoid repetition and improve coherence. It is applicable to short-term memory (session-based) and long-term memory (persistent user preferences).
- **Turn taking & flow control:** AI must know when to speak, listen, ask follow-ups, or hand off the conversation smoothly [156].
- **Emotional intelligence:** detecting sentiment (analyzing tone, word choice) to response accordingly, e.g., if a user sounds frustrated, the AI may switch to supportive tone, offering sympathy before giving solution.
- **Personalization & adaptation:** AI should adapt its responses based on user preferences, previous interactions or sentiment, e.g. if a user frequently asks about tech news, the AI prioritizes tech-related responses [156], [157], [158].
- **Error handling:** AI must detect misunderstandings and correct itself e.g. by asking clarifying questions.
- **Multi-turn conversation & proactiveness:** AI should handle long conversations, understand follow-ups, and proactively offer suggestions.
- **Ethical & privacy considerations:** AI should ensure transparency, legal compliance [159], respect user privacy [154], and avoid manipulation. It should also handle sensitive topics with care (e.g., mental health) [158].

There are a few existing solutions to tackle the problem of conversation management in AI. A foundational solution in the development of conversational agents are rule-based systems, offering structured and interpretable frameworks for dialogue management [160]. These systems rely on if-then logic, decision trees, or state machines to determine responses based on user input. They are deterministic - they follow strict patterns and do not learn from interactions. While they offer reliability and control, they can struggle with handling unexpected inputs or complex conversations [161], [162]. Some modern approaches integrate rule-based logic with machine learning to improve flexibility and adaptability. In [163] authors propose a hybrid approach where pre-trained language models are incorporated into traditional rule-based frameworks.

Another solution that is used in conversational AI is RL. It is a machine learning paradigm where agents learn optimal behaviors through trial and error, receiving feedback from their actions to maximize cumulative rewards. RL enables dialogue systems to improve interactions by learning from user feedback and adjusting responses accordingly. The papers [164], [165] propose ways of using this paradigm to develop agents capable of adapting to complex and changing scenarios.

Recent advancements in AI conversation management are significantly influenced by transformer-based models like BERT and GPT. By leveraging their advanced language understanding and generation capabilities, these models contribute to the development of conversational agents that are more responsible, context-aware, and human-like in their interactions [166]. The authors of [167] are going a step further and propose integration of

graph structures into language transformers to improve the prediction of subsequent actions in complex conversational scenarios. Instead of relying on external knowledge graphs, the study builds graph representations of conversations by capturing relationships between human utterances and actions. This means that the model can understand contextual dependencies and improve predictions without needing external databases.

A crucial aspect for building trust and handling ethical issues in conversation management is explainability. It ensures that users understand the reasoning behind AI responses, fostering transparency and reducing bias. Additionally, explainability allows for better debugging, regulatory compliance, and seamless human-AI collaboration, making interactions more reliable and accountable [159].

Most of the key aspects of conversation management in AI are still in the development phase. These include enhancing emotional intelligence [168], addressing ethical and legal considerations, and improving explainability[159]. Additionally, there is a growing need to better handle ambiguity in statements like sarcasm[169] and humor [170].

3.3.3 Computational Argumentation

Several techniques for generating explanations in XAI can be viewed as *argumentative* [171]. For example, attribution methods, such as model-agnostic approaches [67] and model-specific methods [172], establish connections between inputs and outputs through positive and negative relationships, which could be seen as arguments in favor and against. Additionally, contrastive explanations highlight reasons for and against particular outputs [173]. However, there are also more explicitly argumentative approaches to XAI that employ *computational argumentation* techniques [174]. Computational argumentation or argumentation theory can be useful for XAI due to its solid theoretical and algorithmic underpinnings, as well as the flexibility it offers, especially in the diverse range of argumentation frameworks (AFs) available in the literature [175]. These AFs provide methods for defining arguments and the dialectical relationships between them, along with semantics to assess the dialectical acceptability or strength of arguments, although they can vary significantly in how these components are defined. When AFs are applied for generating explanations, (weighted) arguments and dialectical relations can effectively represent elements such as input data (e.g., categorical data or pixels in an image), knowledge (e.g., rules), internal components of the method being explained (e.g., filters in convolutional neural networks), problem formulations (e.g., planning, scheduling, or decision-making models), and outputs (e.g., classifications, recommendations, or logical inferences). Argumentation is highly applicable to decision making, e.g. through support or opposition of a decision, reasoning for a decision, tackling knowledge with uncertainty, and recommendations. *Argumentation dialogues* are maybe the most natural method of providing an explanation [176]. We are therefore also studying the literature on computational argumentation as a potential framework for user-AI conversations, which could offer targeted on-demand explanations in response to the user questioning specific AI decisions, assumptions, or conclusions.

3.3.4 Knowledge Granules and Levels of Abstraction in Conversational Management Systems

Adjusting the level of abstraction is crucial for intent recognition, contextualization, and personalization because it enables the system to tailor its responses according to the depth and complexity of the user's needs. For intent recognition, varying the level of abstraction allows the system to better understand and address user requests, whether they are simple inquiries or more complex emotional expressions. In terms of contextualization, dynamically adjusting abstraction levels ensures that the system can maintain coherence throughout the conversation by recalling relevant prior exchanges, avoiding repetition, and offering responses that fit the evolving context. Furthermore, for personalization, this flexibility allows the AI to adapt to the user's preferences and behaviors—such as providing more detailed answers on topics the user cares about—creating a more responsive and engaging interaction that aligns with the user's expectations.

At the heart of this process lies the need for explainability, as explaining something to someone is fundamentally the act of conveying knowledge from an entity that understands the causal relationships between events to another entity, the user [177]. Just as human understanding is rooted in recognizing and linking causes and effects, a conversational management system must provide transparent and understandable explanations to ensure users grasp the reasoning behind its responses. As such, explainability should be an essential core component of any conversational management system, enabling users to trust, interpret, and engage with the AI in a meaningful and informed manner. Therefore, managing abstraction levels in conversational systems is crucial, as it allows the AI to adapt its explanations to the user's cognitive level, ensuring clarity and relevance while fostering deeper understanding and engagement.

While concept-based explantionsexplanations discussed previously focus on substituting low-level explanations with higher level concepts, the level of abstraction that they operate is usually fixed. However, depending on the target audience of explanations, there might be a need in proper adjustment of this abstraction level. This research problem was brought up by several groups in recent years.

In [159] authors emphasize the need for XAI methods that can provide explanations at various abstraction levels, from detailed, instance-level insights to high-level, conceptual understanding, to enhance transparency and trust in AI systems.

Similar arguments were brought in [178], where authors emphasize the need of tailoring explanations to the audience's knowledge level and cognitive capacity, ensuring that the complexity and transparency of the explanation are adjusted accordingly, such as using simpler natural language for laypeople or more technical details for domain experts. It also highlights the goal of making explanations simple enough for the audience to fully understand and simulate the decision process, a concept known as simultability, which may involve adjusting granularity, like explaining a diagnosis differently to doctors versus patients.

One of the works in the area of data conceptualization for XAI was presented in [179], where information granules and granular computing for XAI are introduced along with quality measures that are related to the level of abstraction of the information granules (concepts).

The work shows how to adjust the level of abstraction of explanations by composing concepts out of raw data. Although such conceptualization is definitely a step ahead towards intelligible XAI, it focuses solely on the data as the only input for forming granules. It does not take into consideration a wider spectrum of the concept's dependencies, neither background knowledge nor XAI-related quality measures.

4. PEER Building Blocks for Collaborative and Adaptive Decision System

The use of computer programs to support decision-making dates back to early works such as Bonini in the 1960s [180] and Gorry in the 1970s [181]. Since then, a substantial body of research has been developed in this field, leading to increasingly sophisticated architectures for AI-based decision support and planning systems.

The classical agent's architecture defines an agent as an entity that perceives its environment through sensors and makes decisions upon that environment through actuators (Figure 9) [1]. The internal structure between sensors and actuators depends on the required capabilities of an agent. For decision-support and planning systems in complex domains (e.g., non-stationary environments and multi-objective problems) purely reactive agents are not capable enough as reasoning on future actions is required. For the PEER Companion, we also need to assume human user preferences, so simple achievement of a goal is not enough, and a utility-based agent is more appropriate (Figure 10). The utility-based agent uses a model of environment (How the world evolves), a model of agent capabilities (What my actions do), and a model of agent's wishes (Utility). Moreover, the agent should be able to accommodate changes in the environment and to obtain and maintain the utility function via learning (Figure 11). Anything in agent's architecture (Performance element) can be improved via learning (Learning element). Learning is driven by feedback from the environment (Critics) but the agent may also explore actions that will lead to new and informative experiences (Problem generator).

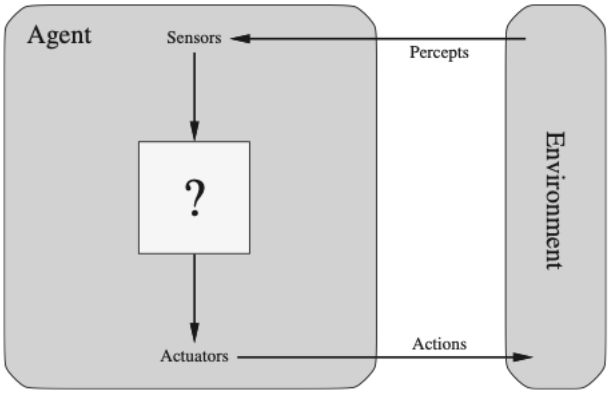


Figure 9. Architecture of AI agent [1]

Additionally, the PEER Companion has some specifics that should be reflected in the agent's architecture, differentiating it from classical architectures:

- sensor inputs can go directly to the PEER Companion but information about the environment is also going via communication with the human user,
- the PEER Companion is only suggesting actions to a human user to execute (but does not execute them directly),

- the PEER Companion mimics the utility function of a human user and needs to learn this function and also to accommodate it in time.

The above specifics require customization of the general architecture of autonomous agents to suit companion agents.

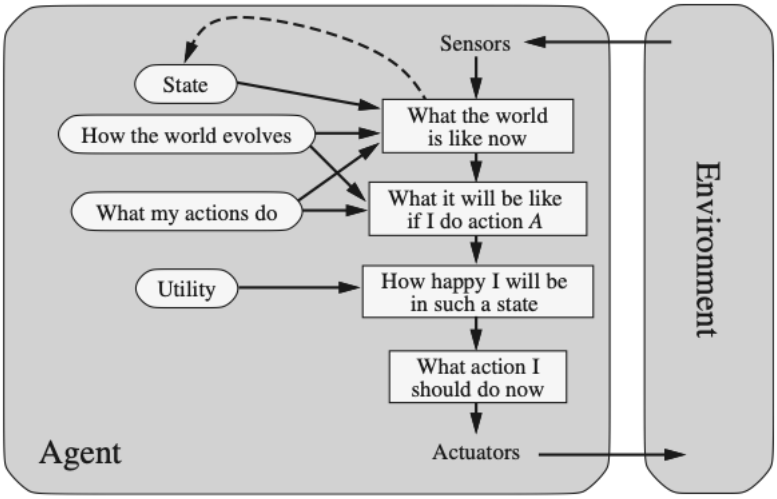


Figure 10. Architecture of utility-based agent

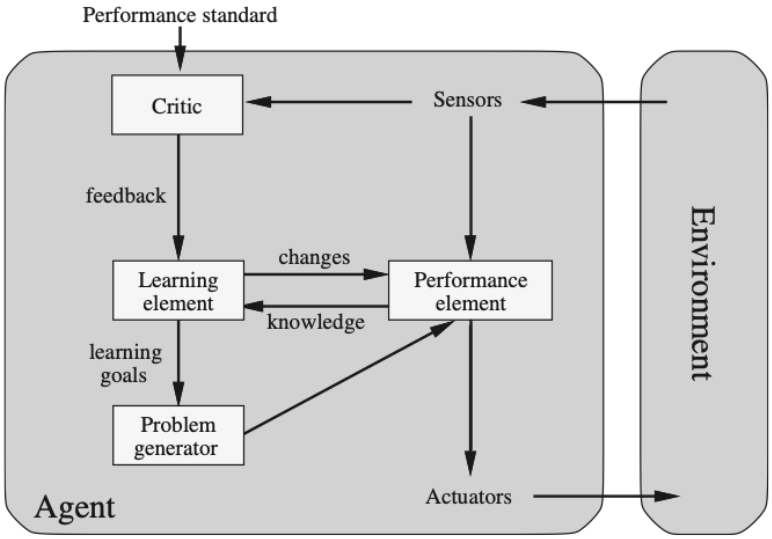


Figure 11. Architecture of general learning agent

4.1 PEER Modular Architecture – Overview

In the project we propose a modular architecture to describe the PEER Companion building blocks. The major motivation was to separate the components of the agent so that they can be developed independently and customized for particular use cases. The major difference from classical autonomous agents' architectures is that the PEER modular architecture

describes a human companion. Specifically, the PEER Companion does not have its own utility function but rather mimics the utility function of its user. This is reflected in the User Model. The PEER Companion may not only observe the environment directly, but it uses also input from the user. The PEER Companion does not execute the actions directly but rather suggests the actions to the human user who is the final authority to decide whether to execute the suggested action or to select another one. Finally, by observing human decisions, the PEER Companion is supposed to continually improve its own models such that its suggestions correspond better to human needs. The reason for using explicit models (of environment and user) is exploiting these models to generate explanations for suggestions of the PEER Companion. The quality of the agent can be measured by the ratio between accepted and declined suggestions. In the following, we briefly explain each component within the PEER modular architecture.

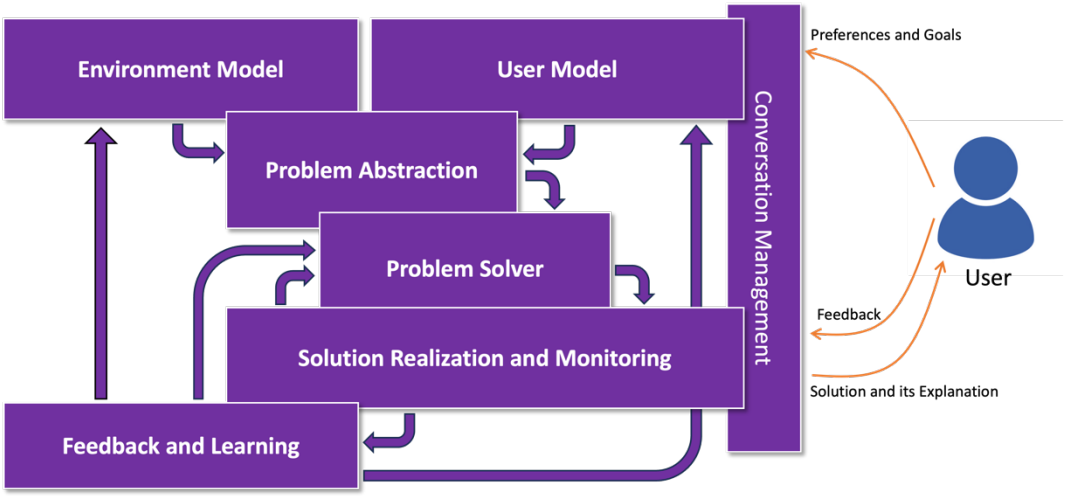


Figure 12. Architecture of the PEER Companion Agent.

4.2 Environment Model

The Environment Model is used by an agent to represent an environment in which the human user operates. In sequential decision making the agent typically needs to reason about future actions in order to achieve distant goals. The environment model describes how agent actions influence the environment and hence can be used for reasoning and planning. This model should contain all the details about the environment required to plan future actions of a human user. It is not static and should be updated based on information from plan execution (via the learning and feedback module) and other sources.From the classical agent's architecture (Figure 10), this module provides answers to questions What the world is like now (State) and What it will be like if I do action A (How the world evolves, What my actions do). It is part of the Performance element of the general learning agent (Figure 11) that is improved via learning.

4.3 User Model

The PEER Companion is not autonomous in the sense of maximizing its own utility function but rather mimics the utility function of its human user. This is realised via the User Model that describes capabilities of the human but also human's preferences and goals. This module represents the personality of the human user and allows customized suggestions of the PEER Companion. The user may directly express own goals and preferences (via the Conversation Management Module), but the PEER Companion may also learn tacit preferences based on observation of human actions (whether the human accepts the suggestion of the PEER Companion or rather uses/requires a different action).
From the classical agent's architecture (Figure 10), this module provides information about what the human agent can do (What my actions do), what is the current State of human user, and answers to question How happy I will be in such a state (Utility). It is also part of the performance element of the general learning agent (Figure 11) that is improved via learning.

4.4 Problem Abstraction Module

The role of the Problem Abstraction Module is to formulate the abstract problem to be solved by the PEER Companion. It combines information about the environment with the goals and preferences of the user to describe formally a (planning) problem to be solved. When formulating the problem, abstraction is used to remove details from the environment and user models not needed for problem solving. The Problem Abstraction Module may generate multiple abstract problems (for example by putting emphasis on particular preferences of the human user) to obtain alternative solutions. Hence, it plays the role of Problem generator from general learning agent (Figure 11). Later, based on selection of particular solution by the human user, the PEER Companion may learn the human user's preferences.

4.5 Problem Solver

The Problem Solver is an algorithmic module responsible for solving the abstract problem formulated by the Abstraction Module. Basically, it generates a plan to achieve the goal while assuming the preferences of the user. As the first step, this module generates one or more solutions (plans) for the abstract problem. Later, during execution, this module can be used to modify the solution based on the status of execution (for example, when the user deviates from the suggested solution) or to provide a completely new solution.
From the classical agent's architecture (Figure 10), this module answers the question What action I should do now.

4.6 Solution Realization and Monitoring Module

The Solution Realization and Monitoring Module is responsible for executing the plan and collecting feedback from the user. More precisely, this module suggests the actions from the plan to the user, who accepts/executes them or not, monitors execution of the actions (observes which action has actually been executed by the user), collects feedback from the user (both direct and indirect), and based on that, it modifies the plan (future suggestions) by asking the Problem Solver. This module also provides explanations to the user of why a given action has been suggested by the PEER Companion. The feedback from the user can be direct (accepting or rejecting an action suggested by the PEER Companion) or indirect (the user simply executes an action different from the suggested one). Part of the feedback from the user could be explanation from the human user of why the user deviates from the plan by using action different than suggested. This information will be used later by the Learning Module to update Environment and User models.

4.7 Feedback and Learning Module

This module is responsible for updating the Environment and User models based on information collected during execution. During execution, this module may also update the problem formulation to reflect the behaviour of the human. Primary information used by this module is whether the human user followed the suggestions of the PEER Companion or deviated from them. In case of deviation, the user may provide explanation for it that can be used to "understand" the reason for deviation and to update the models more accurately.
This module represents the Learning element from general learning agent (Figure 11).

4.8 Conversation Management Module

The Conversation Management Module is basically a bi-directional user interface to the PEER Companion. It is directly connected to two modules of the architecture. The connection to the User Model is used to directly express the goals and preferences of the human user. The connection to the Solution Realization and Monitoring module serves for multiple purposes. The PEER Companion communicates to the human user the (next) suggested action(s) and (by request) may also provide explanation why that action has been selected. Vice versa, information from the user is collected (directly or indirectly) about whether the suggested action has been accepted and successfully executed or not. In case of deviation from the suggested action, information about the reason of deviation (explanation) may be collected that will be used later by the Learning module.
At agent's architectures (Figure 9, Figure 10, Figure 11) this module roughly plays the roles of Sensors and Actuators.

5. Realization of PEER Modules in Use Cases

This section presents a mapping and instantiation of PEER modular architecture within four use cases. Although the PEER architecture provides generic building blocks for enabling human-AI and collaborative assistant, some modules have not been yet fully defined and therefore not reported for the first MVP.

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

As mentioned above, the PEER modular architecture provides a structured framework for developing AI-driven assistants that support human decision-making. Unlike fully autonomous systems, PEER Companions act as collaborative companions, learning from human users and refining their suggestions over time. This subsection maps SONAE's in-store picking use case to the PEER modular architecture, illustrating how each module contributes to assisting customers in navigating a store while following their personalized shopping preferences.

In this specific use case, PEER helps customers to optimize their in-store shopping journey. Customers build their shopping list in an app, specify constraints (such as picking frozen products last), and receive an initially optimized route. As they navigate the store, the system updates the suggested route whenever customers deviate from the current picking route, or new constraints arise. At the core of this solution is a simulation environment, structured as an OpenAI Gym environment, where customers will be able to interact with a playable prototype before being able to deploy the system in real-world stores.

To better understand customer behavior and preferences, the simulation environment captures every interaction, including movement choices, constraint modifications, and adherence to suggested routes. After completing a session, users provide explicit feedback, rating their experience and explaining their decisions in natural language. This data contributes to refining the system's recommendations and improving future iterations of the PEER Companion. The primary goal of this MVP is not just to optimize in-store picking but also to establish a flexible framework for solving other sequential decision-making problems in in-store picking applications.

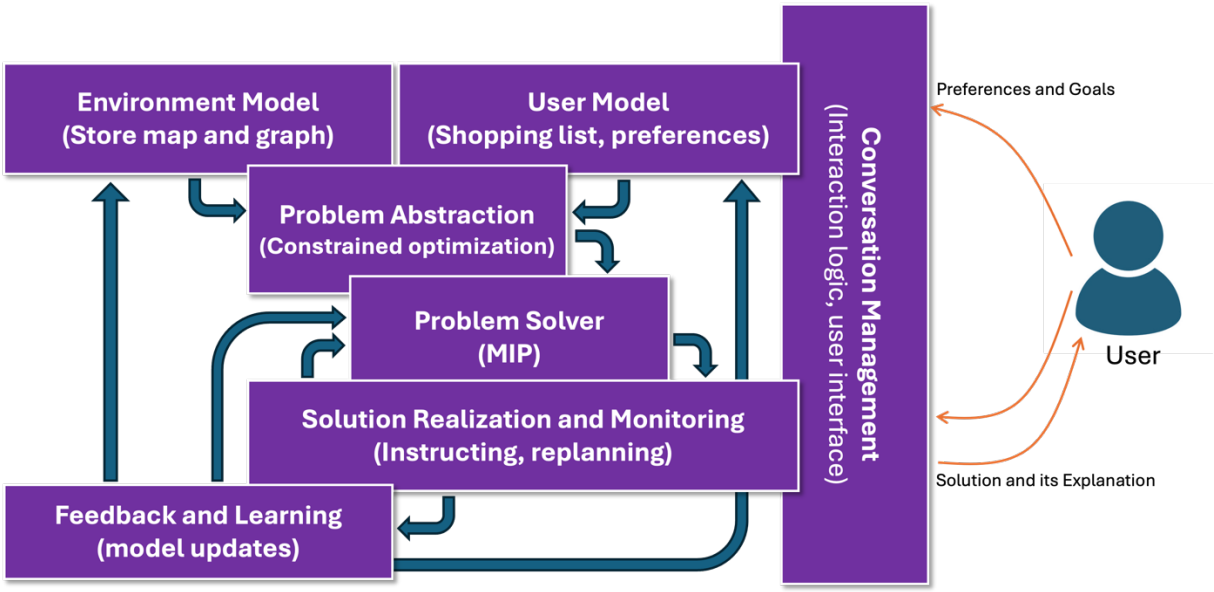


Figure 13. PEER building blocks mapping to personalized in-door shopping assistant use case

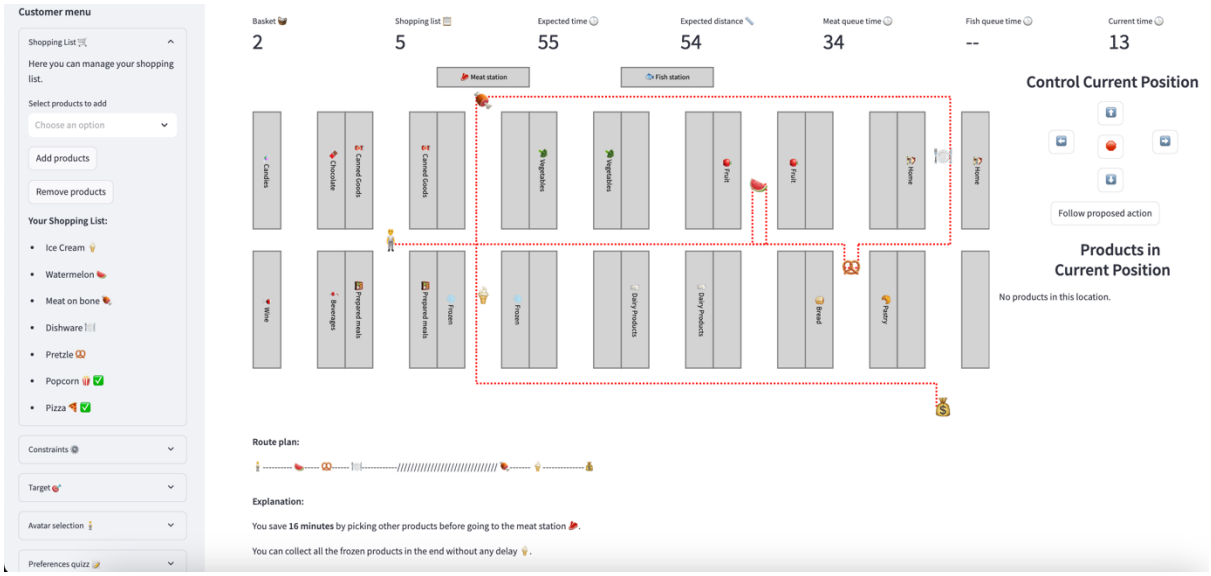


Figure 14. A screenshot of simulator and the interface of envisioned PEER shopping assistance (Sonae use case)

5.1.1 Environment Model

The Environment Model represents the retail store setting in which customers operate. It includes the store layout, shelf locations, product locations, and available paths that define how customers move through the space. This model is fundamental for planning in-store picking routes and adapting to customer preferences.

In the current implementation, the environment is simulated within an OpenAI Gym environment, which allows controlled training and testing of different decision policies. The simulation environment does not yet consider many external uncertainty sources, such as aisle congestion or product stockouts. Instead, it focuses on uncertainty introduced by the customer's behavior, specifically when they deviate from the suggested picking route.

This controlled environment allows for evaluating how different policies influence user adherence to recommended routes and how additional constraints impact overall shopping metrics. Over time, the Environment Model will evolve to incorporate additional dynamics, such as real-time store updates (e.g., aisle crowding, shelf restocking). However, for this initial MVP, the model primarily serves to structure a dynamic decision-making problem and support the development of optimized and preference-aware picking routes.

This simulation environment provides a safe testing ground where customers can interact with a playable prototype, testing the feasibility of the approach before real-world implementation.

5.1.2 User Model

The User Model is designed to capture and represent customer preferences, constraints, and behaviors during the in-store picking journey. Unlike fully autonomous systems, the PEER Companion does not optimize for its own objectives; rather, it seeks to mimic the customer's utility function, refining its recommendations based on explicit input and observed behavior.

At the beginning of the shopping journey, customers explicitly define preferences through the app, such as prioritizing fresh produce first or ensuring frozen items are picked last. These preferences are then translated into constraints that influence the picking route generated by the system.

Beyond explicit preferences, the User Model learns from observed behavior. The simulation environment captures customer decisions in real time, such as deviations from the suggested route, changes in item selection order, shopping list updates, or modifications to predefined constraints. By analyzing these interactions, the system can refine its understanding of individual preferences, improving future recommendations.

To further enhance personalization, customers provide direct feedback at the end of each session, including, for example, a numerical satisfaction rating (1 to 5) to measure overall experience and free-text feedback to capture insights on why they deviated from recommendations or how the system could improve.

Over time, this data enables the user model to evolve, ensuring that the PEER Companion better aligns with individual shopping habits and delivers increasingly relevant suggestions.

5.1.3 Problem Abstraction Module

The Problem Abstraction Module is responsible for formulating the SDM problem that underpins the in-store picking experience. It integrates information from both the Environment Model and the User Model, ensuring that the problem formulation reflects, for example, the customer's shopping list, stated preferences, and the store's spatial structure.

This module does not solve the problem itself; instead, it structures the necessary inputs for the Problem Solver, ensuring that only relevant details are considered. The problem formulated by this module involves path planning and constraint satisfaction under uncertainty, balancing route efficiency with user-defined preferences.

During the shopping journey, this module dynamically updates the problem definition based on customer actions. For example: (1) If the customer modifies constraints (e.g., adding a new preference mid-shopping), the problem formulation is adjusted accordingly. (2) If the customer deviates from the suggested route, the module updates the state representation, ensuring that the next recommendation aligns with the customer's new position and intent.

By structuring the problem in a way that accounts for both predefined constraints and real-time deviations, the Problem Abstraction Module enables the Problem Solver to generate feasible and customer-preferable solutions.

5.1.4 Problem Solver

The Problem Solver is responsible for computing optimized picking routes based on the problem formulation provided by the Abstraction Module.

In the current implementation, the Problem Solver relies on a Mixed-Integer Programming (MIP) model, which optimizes picking routes while respecting customer-defined constraints. The initial picking route is generated before the shopping journey begins. However, as the customer navigates the store, the Problem Solver can re-optimize the route in response to deviations or newly introduced constraints.

Relying on MIP-based solutions may work well in deterministic settings but this is not the case when the customer is faced with uncertainty (e.g., aisle congestion, stock fluctuations). To better handle uncertainties, future iterations of the Problem Solver will need to incorporate non-myopic, adaptive policies which consider immediate and future consequences of the actions made by the customer. After the first MVP, the research team will improve the initial approach by incorporating methods that are less myopic, such as, Value Function Approximations (VFAs), Direct Lookahead Approximations (DLAs), RL or Stochastic Optimization Methods.

5.1.5 Solution Realization and Monitoring Module

The Solution Realization and Monitoring Module is responsible for guiding the customer through the in-store picking process by suggesting actions, monitoring execution, and collecting feedback. Instead of executing actions autonomously, it provides recommendations that the customer may choose to follow or modify.

As the customer moves through the store, this module tracks whether it follows the proposed path or deviates from it. If deviations occur, the system updates future suggestions while

allowing the customer to provide reasons for their choices. Additionally, the module explains its recommendations, ensuring the customer understands why a particular route was suggested.

To improve future recommendations, the module gathers feedback both implicitly—by analyzing user decisions—and explicitly—by prompting the customer to rate their experience and provide comments. This data is essential for refining the Environment and User Models, ensuring that subsequent route suggestions better align with customer behavior.

Over time, the Solution Realization and Monitoring Module will evolve to offer more interactive explanations and refine real-time adjustments, making the system more adaptive and transparent in assisting customers with their in-store picking journey.

5.1.6 Feedback and Learning Module

The Feedback and Learning Module collects data on customer behavior and preferences to refine future recommendations. At this stage, the module primarily gathers implicit feedback by tracking route deviations and explicit feedback through customer ratings (see Figure 15) and comments at the end of the shopping session. While no advanced learning mechanisms are currently implemented, this data will serve as the foundation for future iterations, enabling the system to adapt and improve route suggestions based on long-term customer behavior.

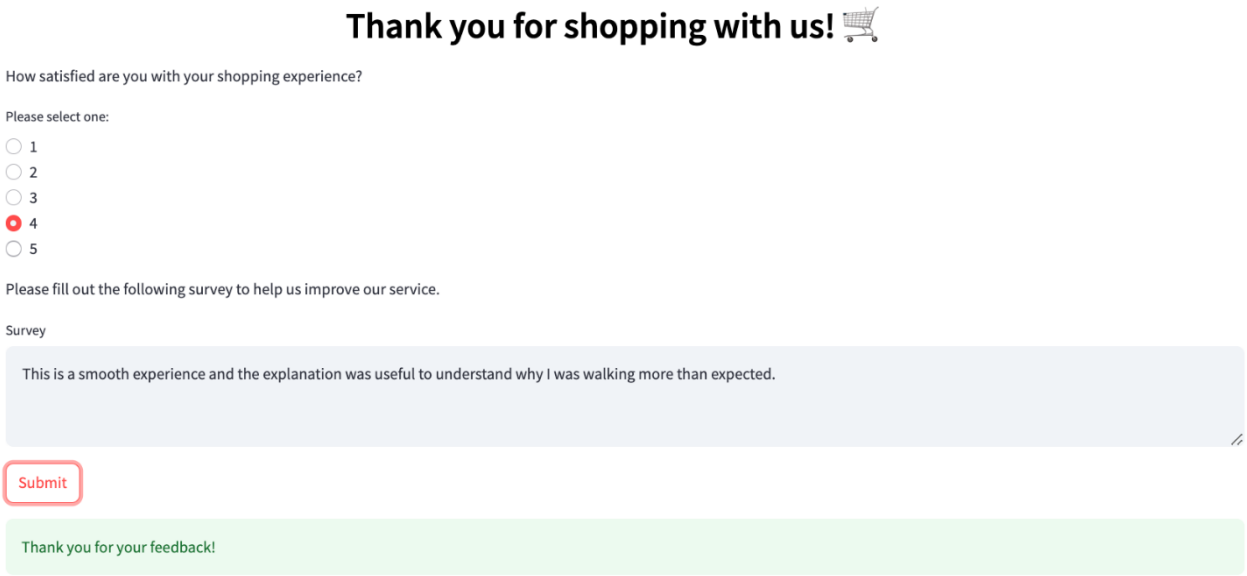


Figure 15. A screenshot of explicit user feedback form by which AI will learn to adapt for user needs

5.1.7 Conversation Management Module

The Conversation Management Module serves as the interface between the customer and the PEER system, allowing users to input their shopping list, define preferences, navigate the store, and interact with the suggested picking routes. In the current implementation, the interface is built using Streamlit package and is integrated into the simulation environment,

where customers manually navigate, receive route suggestions, and provide feedback. This is an alternative solution to allow for picking route execution without being in a real retail store (before developments are integrated into SIGA app).

At this stage, the Conversation Management Module is primarily focused on enabling structured user interactions and data collection (user feedback). Future iterations will enhance its capabilities by incorporating a more intuitive design, interactive explanations, and real-time adaptability to improve the overall user experience.

5.1.8 Next steps

In the MVP the focus shifted from in-store pickers to assisting customers in their shopping journey, which led the research team to focus its efforts on implementing a retail store environment to learn decision policies considering customer feedback. The next iteration will focus on solving this realistic SDM problem using customer feedback and state-of-the-art, non-myopic approaches, moving beyond the reliance on Mixed-Integer Programming (MIP) as a decision policy. Some key research directions we aim at exploring are the following:

- **Learning-Based Route Optimization:** Replace myopic MIP-based decision-making with Reinforcement Learning (RL), Value Function Approximations (VFAs), and Direct Lookahead Approximations (DLAs) to enable adaptive, forward-looking routing strategies.

- **Beyond Path Planning:** Extend recommendations beyond route optimization by suggesting relevant products, considering real-time store conditions, and optimizing the in-store experience for both efficiency and engagement.

- **Interactive and Explainable AI:** Integrate explainability and bi-directional communication, allowing customers to modify constraints and understand why specific routes or product suggestions are made.

- **Preference Modelling:** We plan to develop personalized recommendation system for products to be offered for users according to the historical shopping habits and objective function.

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

This subsection maps Amsterdam's personalized city navigation use case to the PEER Modular Architecture. It illustrates how each module contributes to helping wheelchair users navigate the city, aiming at a more inclusive and accessible Amsterdam.

In this use case, the PEER Companion should help wheelchair users find a route from their starting point to the destination within the city of Amsterdam. Project workshops (WP2) as well as prior work at the municipality of Amsterdam has shown that the users' requirements and needs for such city navigation can vary strongly, both from person to person as well as for the same person from day to day or navigation task to navigation task. This motivates the creation of a personalized routing system, which does not simply compute the shortest path but is able to suggest the best path given the user's requirements (e.g., constrained by a minimum width of $80cm$, a maximum step height of $2cm$, and 8% steepness; or an appropriate alternative route in case of unforeseen obstacles). The goal of PEER is to offer the user truly collaborative problem solving: trade-offs between the user's preferences for example can be explored with multi-objective decision-making techniques, and potentially changing conditions such as obstacles on sidewalks can be taken into account by accepting corrections from the user based on the dynamically changing local environment.

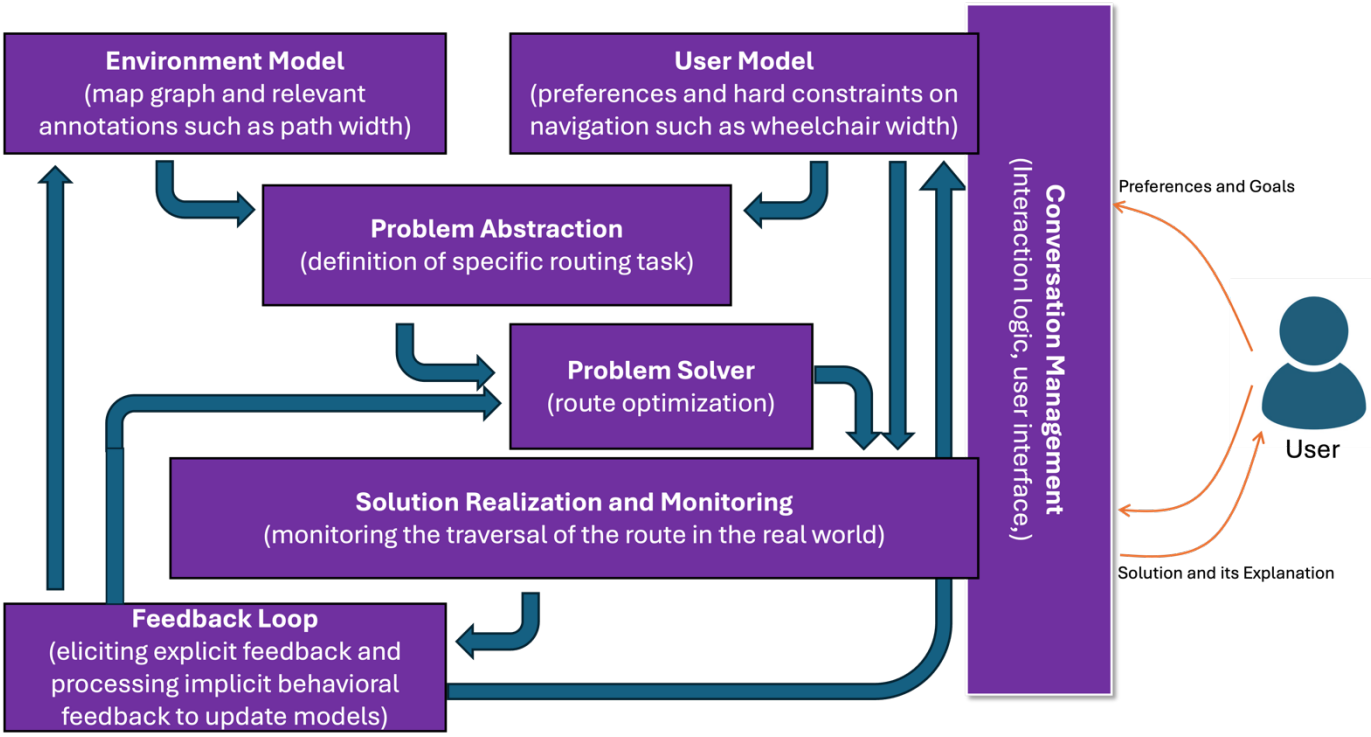


Figure 16. PEER building blocks mapping to explainable and personalized and route planner for wheelchair users

5.2.1 Environment Model

The Environment Model represents the city map, enriched with additional data that is relevant for city navigation according to the user requirements elicited in WP2's early workshops. The feasibility of these requirements often depends on the availability of related data; we can for example not promise to respect a maximum steepness of a planned route as long as data on the steepness of streets and sidewalks is not available. However, we already have a set of

several features available that can lead to improved solutions for users with different preferences: Every edge in the map graph (i.e. every single navigable stretch of bike path or sidewalk) is not only annotated with its length, but also its width, its maximum curb height, and whether it is a footpath or a bikepath. Example data and code are available on Github [2].

5.2.2 User Model

The User Model is meant to capture and represent individual preferences and constraints for city navigation. During the MVP and the further project, it will be studied how these preferences and constraint can be best learned by the AI system: either by being explicitly specified on the spot by the user (before starting a navigation task, or as feedback after the route has been traversed in real life), or by being implicitly learned over time by observing user behavior. We will also explore how to best communicate their meaning and their impact on the route suggestions of the PEER assistant. Our initially available set of user features, matching the data currently available in the Environment Model and taking into account features that were requested in user workshops, are 1) the desire for routes that are as short as possible; 2) the desire for routes that are as simple to navigate as possible, approximated for now with the number of intersections that have to be crossed; 3) a possible preference for bikepaths, or for footpaths; 4) a hard constraint on the minimum path width, based on the width of the used wheelchair; and 5) a hard constraint on the maximum curb height. We can represent users with different values for the hard constraints, and different priorities for the soft constraints/objectives. As the MVP will be tested with a Wizard of Oz approach (paper-based), the user model is currently manually specified by a team member.

5.2.3 Problem Abstraction Module

The Problem Abstraction Module formulates the concrete route planning problem at hand in a way that can be processed by the Problem Solver module, based the user's known preferences and constraints, and environment information relevant to it (e.g. data on footpaths vs. bikepaths does not need to be considered if the user does not have any preference for either). Later in the project, it is our goal to let this module dynamically update the definition of the problem based on explicit user feedback (such as reporting an obstacle on the sidewalk) or implicit user feedback (such as deviating from the recommended route). In the MVP, the execution phase is not considered yet, so this module is restricted to working with information that is available in the planning phase before a route is started. This can include a known user profile as well as answers to direct AI questions about preferences and constraints. As the MVP will be paper based, the problem formulation is so far done manually as well.

[2] https://github.com/Amsterdam-AI-Team/Accessible_Route_Planning

5.2.4 Problem Solver

The Problem Solver is the module that actually computes optimized city routes based on the problem formulation provided by the Abstraction Module. In the future, various pathfinding algorithms can be tested here, e.g. taking multi-objective aspects, uncertainty, and/or real-time constraints into account. For the MVP, we are using the simple and robust Dijkstra and sub-gradient algorithms [182]. Routes for different user preferences are prepared in advance of MVP testing.

5.2.5 Solution Realization and Monitoring Module

The Solution Realization and Monitoring Module will be responsible for guiding the user through the process of planning the route and actually navigating the city and following - or not following - the proposed route. It will handle all necessary explanations, such as comparisons of possible routes in terms of pros and cons (see Figure 17); it will monitor the execution of the plan, suggest and explain improved route options computed by the Problem Solver based on continuously refined User and Environment Models, and handle retrospective comparisons of planned and actually taken routes. Since the MVP will be based on indoors testing of Figma prototypes, the development of this module has so far been restricted to the initial planning phase before navigating the city. We will test different communication flows, explanations and feedback options for this phase. In the next deliverable, the phase of actually navigating the city will be considered as well.

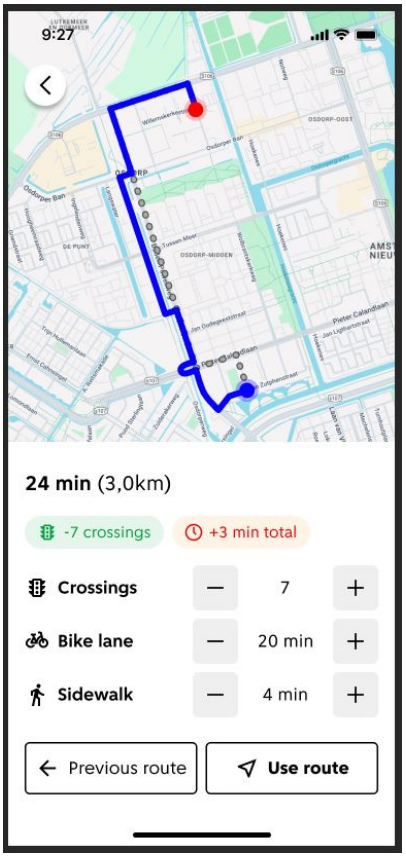


Figure 17. An example of an explanation of the differences between two routes in terms of objectives the user cares about (from the MVP).

5.2.6 Feedback and Learning Module

The Feedback and Learning Module collects and processes data on user behavior and preferences to refine future recommendations. In the MVP, we are already testing different explicit and implicit ways to obtain user feedback on the recommended routes (see Figures 18 and 19). In future implementations, this module will be responsible for learning user models over time, and keeping environment models are up-to-date.

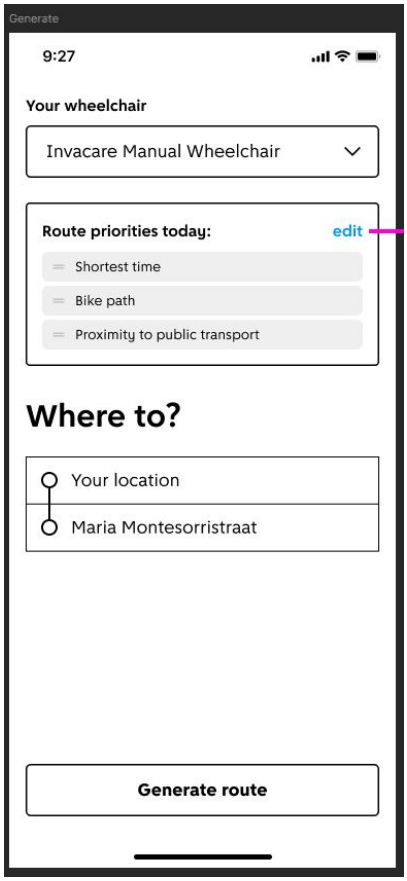


Figure 18. An example of explicit user feedback on user preferences, accepted before the trip (from the MVP).

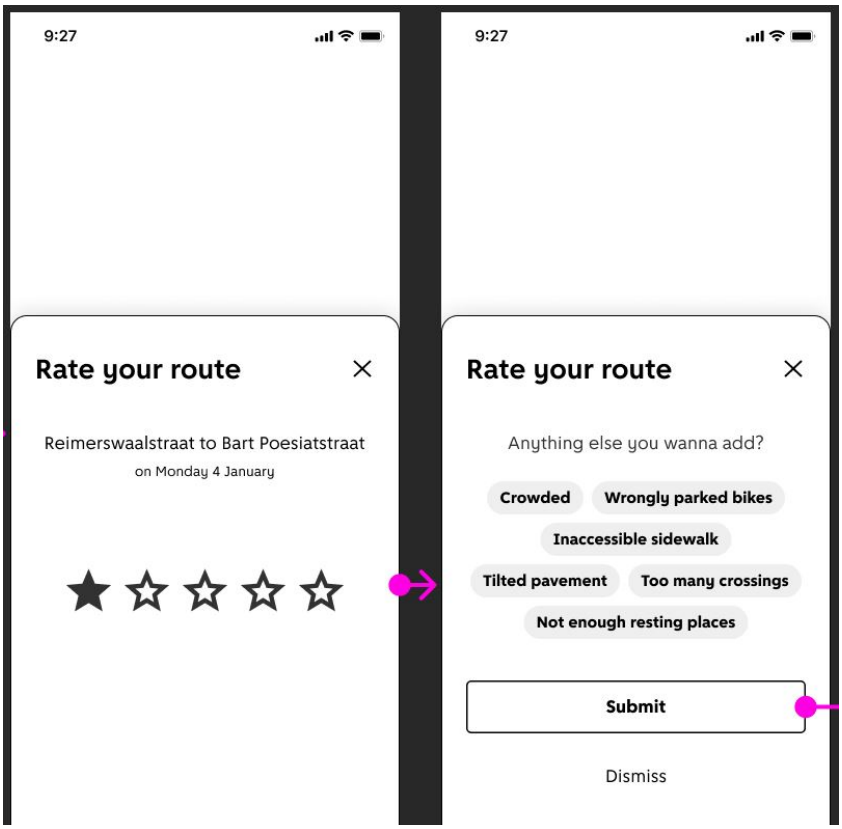


Figure 19. An example of explicit user feedback on solution quality, accepted after the trip (from the MVP).

5.2.7 Conversation Management Module

This module serves as the interface between the user and the PEER system, handling inputs and outputs of all user interactions. In the MVP, the PEER Companion will be presented to the users in paper form (see the Figma designs shown in this section for examples), and interactivity will be simulated by workshop leaders; in the next deliverable iterations, it will be an interactive mobile app.

5.2.8 Next steps

In line with the project's expectations for the second round of use case evaluations, the MVP is delivered in the form of Figma designs for Wizard of Oz style AI simulations, with manually prepared routing solutions and initial options for demonstrating AI explanations and user feedback. In the next phase of the project, we plan to move to code implementations of all PEER modules, allowing the users to directly interact with the system in the third round of use case evaluations.

In active collaboration with the municipality of Amsterdam, a number of next design ideas have already been prepared and realized in Figma. These will be adapted both to the results of the lab evaluation with potential end users, as well as to the hypotheses to be tested in future research in Tasks 3.1-3.3. Key research directions include:
In line with the project's expectations for the second round of use case evaluations, the MVP is delivered in the form of Figma designs for Wizard of Oz style AI simulations, with manually prepared routing solutions and initial options for demonstrating AI explanations and user feedback. In the next phase of the project, we plan to move to code implementations of all PEER modules, allowing the users to directly interact with the system in the third round of use case evaluations.

In active collaboration with the municipality of Amsterdam, a number of next design ideas have already been prepared and realized in Figma. These will be adapted both to the results of the lab evaluation with potential end users, as well as to the hypotheses to be tested in future research in Tasks 3.1-3.3. Key research directions include:

Multi-objective path planning and explanations that allow the user to hone in on the desired solutions with minimal cognitive effort.

Different user modelling techniques that focus on direct user feedback (as much as accepted by the users, and with the most acceptable timing – before, during, or after navigating the real-world route), and/or indirect user observation.

Route explanations that focus on the most relevant and most understandable aspects for the users; e.g. landmark-based vs. objective-based comparisons between routes.

We also aim to include additional datasets that would allow for taking into account additional user needs, such as (for example) how busy roads are expected to be, or how the lighting conditions are. Any additional dataset could lead to a new user objective to be integrated into route planning, route explanations, and user feedback.

- Multi-objective path planning and explanations that allow the user to hone in on the desired solutions with minimal cognitive effort.
- Different user modelling techniques that focus on direct user feedback (as much as accepted by the users, and with the most acceptable timing – before, during, or after navigating the real-world route), and/or indirect user observation.
- Route explanations that focus on the most relevant and most understandable aspects for the users; e.g. landmark-based vs. objective-based comparisons between routes.
- We also aim to include additional datasets that would allow for taking into account additional user needs, such as (for example) how busy roads are expected to be, or how the lighting conditions are. Any additional dataset could lead to a new user objective to be integrated into route planning, route explanations, and user feedback.

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

One of the many challenges of industrial applications of AI has been Anomaly Detection (AD), where a number of state-of-the-art methods are being used. AD, the task of identifying abnormal patterns in industrial Identify applicable funding agency here. Data (e.g., images), plays a critical role in applications like quality assessment and real-time monitoring. However, existing AD techniques, largely powered by AI and ML, face significant challenges that hinder their deployment in Industry 5.0 environments. AI-based AD systems serve different purposes, e.g., quality assurance. One such AD use case is product visual inspection [183]. The company we work with uses AD for tuning an industrial visual inspection system. The central challenge that gives us motivation for our work is related to the fact that practical tuning of AD algorithms is a complex task for non-data-scientists users that needs guidance and understanding of the data collection and labeling, anomaly detection method as well as the algorithm configuration process. This section overviews the different modules we envisioned for PEER Companion within the scope of Proditec (PDT) AD use case.

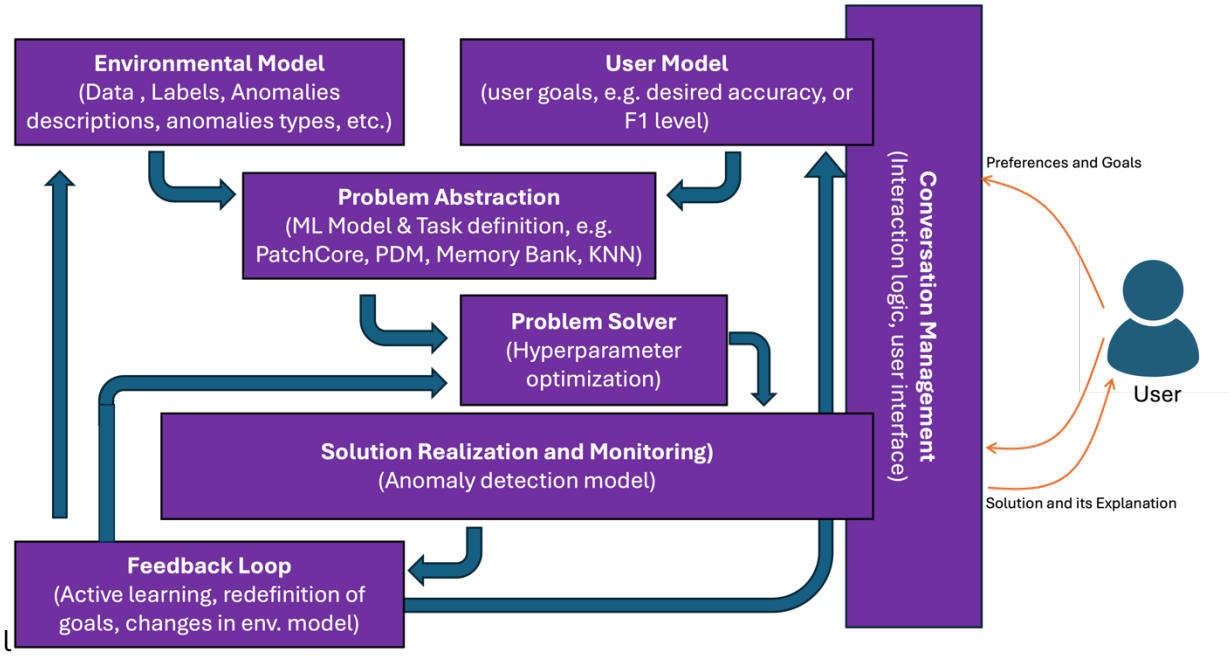


Figure 20. PEER building blocks mapping to explainable anomaly detection use case

5.3.1 Environmental Model

For PDT use case, our environmental and contextual data consists of an initial dataset required for training consisting of at least 600 unlabeled images. Experts review and label the provided images. A minimum of 500 good and 100 defective coins must be labeled to initiate training. While the dataset can include additional images, only the minimum labeled set is mandatory for starting the process.

Unlabeled images are also part of the environment module and are categorized in an iterative process if performance benchmarks are not met. This dataset size was determined through tests conducted by PDT and can be increased by the customer to achieve higher accuracy, especially for complex coin types. Images used in the dataset are directly captured by the machine's cameras, ensuring authenticity and quality. Factory settings for camera parameters, including lighting, zoom, and positioning, are pre-configured and remain unchanged during standard machine operation unless equipment replacement is required due to wear or breakdown.

5.3.2 User Model

One of the main challenges in designing such AI system lies in the fact that the end-users often have varying levels of technical expertise. These differences stem from both the internal policies of the company and the cultural context of the country and impact the effectiveness of the usage of the system. The hyperparameters of the AD model (e.g., threshold) also affect the performance metrics such as F1 score and False Positive (FP) rates. We consider that this would be highly dependent on the type of product and driven by the internal policy of the company.

5.3.3 Problem Abstraction Module

In industrial settings, anomaly detection often hinges on the ability to model and learn the representation of normal samples. This approach is motivated by the practical challenge that, in many industries, normal operational conditions are well-defined and relatively consistent, whereas anomalies are rare and varied. The State-of-the-art methods like PatchCore, PaDiM, etc. achieve high performance and efficiently use a distance metric to ensure a low inference time while decision making, however, to achieve such results, these methods require algorithmic configuration and hyperparameter tuning in the production. This requirement is beyond the expertise of PDT end users and abstraction module is responsible to formulate a hyperparameter search problem that needs to be solved using Problem Solver module.

5.3.4 Problem Solver

As explained in the abstraction module, the non-data-science user that are responsible for training the AD model, needs to be equipped with explainable hyperparameter optimization and guidance on the optimal steps to be taken to achieve desired performance of the system. This module addresses these gaps by solving the hyperparameter tuning problem more transparently using methods described in Section 3.2.1.8. It allows non-expert operators to guide the search in real-time, with interactive dashboards provided in the Communication module.

5.3.5 Solution Realization and Monitoring Module

Once the AD model is trained, this module provides detailed insights into which regions contributed to the detection of anomalies, making the system more interpretable and trustworthy. In critical fields like industry, such transparency ensures that users can confidently rely on the model's outputs, verify its accuracy, and address any potential errors.We propose SHAP [67] (SHapley Additive exPlanations) explainer as a candidate model for computing explanations for the anomaly detection model output. SHAP provides quantitative, feature-level attributions, explicitly showing how much each feature (pixel or region) contributes to the model's prediction. SHAP can also operate on feature representations that can be tailored to the desired level of detail. This explanation is fed to the conversation management module where the user interacts with the explanations provided by the AI. This explanation addresses both hyperparameter tunning and also prediction for unseen instances.

5.3.6 Feedback and Learning Module

Another aspect of enabling such a paradigm in visual inspection tasks is to incorporate human knowledge to improve model performance from human annotated samples. We argue that actively involving human operators within the anomaly inspection process not only decreases annotation effort but also increases model adaptability by considering operator feedback during AD validation and execution. We envision an AL setup in the feedback module; thus, we not only rely on the learning algorithm, but also reduce the cost of labeling the dataset. A potential candidate algorithm for incorporating AL mechanism for One-Class AD is to query the end-user based on uncertainty score in test phase and incorporate the feedback for both threshold tuning and similar sample annotation which would result in a better data representation.

5.3.7 Conversation Management Module

This module is responsible for the user interface and provides relevant explanation for AI decisions. The current version of the explainability layer relies on heatmaps generated directly by anomaly detection algorithms such as PatchCore [184], DFM [185], and PaDiM [186]. These algorithms compute anomaly scores, which are spatially mapped and visualized on the image to highlight regions exhibiting the most significant deviations from normal patterns. This type of visualization has some limitations that we plan to address in the next iteration. First of all, they are usually model-specific, which complicates their usage in industrial cases, where the standardized approach allows for easier comparison, and evaluation of different methods as well as their practical deployment in the system. Secondly, heatmaps show regions of high importance without providing detailed reasoning for why those regions contribute to the anomaly. As a result, the highlighted regions may lack context about the specific features (e.g., texture, color, or shape irregularities) that led to the anomaly detection,

leaving the explanation incomplete. Additional limitations in the context of anomaly explanations are the resolution and granularity issues, aggregating features into larger regions can lose fine-grained information, while pixel-level explanations may be too noisy to interpret. To overcome those limitations, we propose an explanation layer that unifies the way the explanations are generated, provides custom level of granularity of the explanations and opens possibility for research on more complex, human-centered explanations.

This component is also responsible for modeling, handling, and maintaining communication with users throughout successive iterations of the optimization process.

It not only manages user interactions but also integrates with all other system components, providing a human-centered interface and an intelligent controller that enhances and orchestrates the functionalities offered by the rest of the system.

5.3.8 Next steps

The first release of MVP implementation has been mainly done using Figma design for Wizard of OZ simulation purposes. We plan to implement the individual PEER backend components and extend the human-AI interaction in AD hyperparameter optimization process in finer granularity.
In particular we plan to provide a prototypical implementation of the following components from the architecture: Environmental Model, User Model and Problem Abstraction and parts of the bi-directional communication component.
For the remaining components we will investigate further enhancements to enhance their interactivity and compatibility with the PEER architecture.
This includes:

- Improvements in the Solution Realization and Monitoring module and Feedback module by utilizing unsupervised learning and active learning to limit the labor of data labelling
- Improvements in Problem Solver by expanding the hyperparameters space to process-level parameters, such as time, size of labelled data, different optimization criteria, etc.
- Improvements in the execution and communication modules by exploring the capabilities of multi-modal models that are capable of efficient handling text and image modalities in conversational manner.

5.4 Datacation. AI-assisted warehouse configuration optimization

Efficient allocation of items in warehouses is an important aspect of logistic chains, in particular when assuming various storage and allocation constraints and preferences. Logistics engineers need a tool to handle the increasing complexity of warehouse

environments and to suggest allocation of items from incoming orders as well as possible reconfiguration of warehouses in case of exceeding assigned capacities. This section describes how the architecture of PEER Companion applies to the Datacation (DAT) use case (Figure 21).

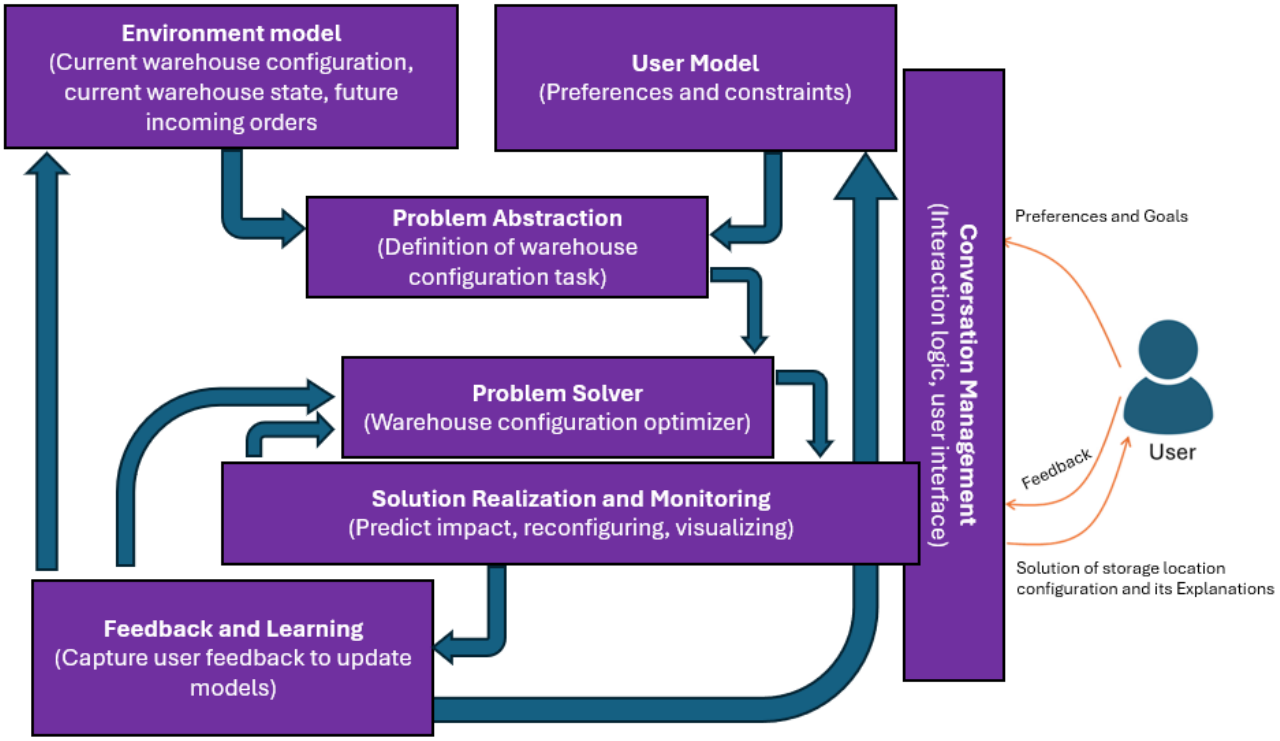


Figure 21. PEER building blocks mapping to smart warehouse configuration management

5.4.1 Environment Model

The Environment Model represents the warehouse, in particular its current status but also expected future incoming orders. A warehouse is basically a collection of spatially related locations used to store products. Locations have properties that restrict the type and quantity of products that can be stored there. The current occupation of storage space is also described here. In addition to a current snapshot of the warehouse, the environment model may also include descriptions of future incoming orders that can be used when allocating the current order to minimize future re-allocations.

5.4.2 User Model

The User Model represents individual preferences, constraints, and context information provided by logistics engineers. These may include information regarding the incoming order, its expected storage duration, parameters to prioritize such as the storage location's distance to the entrance, and constraints such as specific locations to include or exclude from the allocation.

5.4.3 Abstraction Module

The Abstraction Module formulates the optimization problem to solve. It combines information about the status of the warehouse, that is, available locations and existing allocation of items, with the task to allocate items from the new order and inputs of logistic engineers that are encoded as objectives or constraints. The main task of this module is the precise formalization of the problem, which can for example be seen as matching items with locations, or optimizing a flow of items to and from locations (see *Figure 22*). Abstraction means eliminating details not necessary for solving the problem. This could be omitting locations not related to the current order, or grouping individual locations into meta-locations.

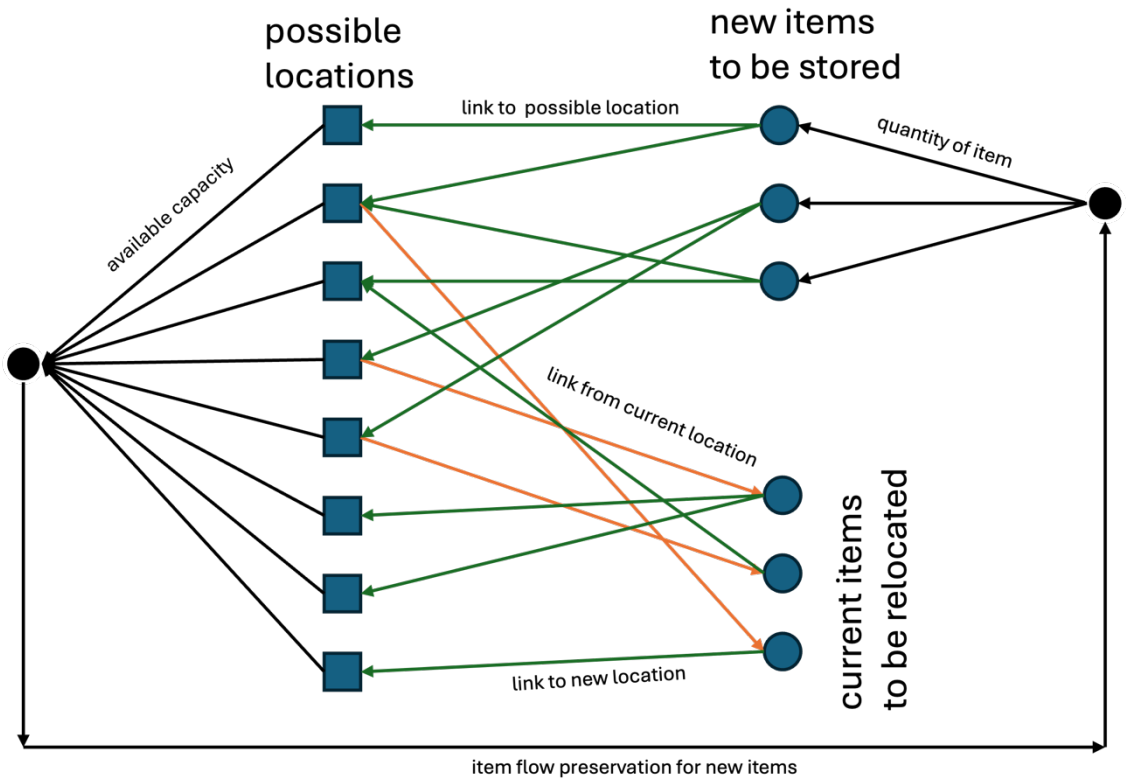


Figure 22. Possible abstract model for warehouse configuration management using network flows

5.4.4 Problem Solver

The Problem Solver is the module that generates warehouse (optimal) layout design, that is, allocation of items to locations. The solving process is supposed to be interactive, done in co-operation with the logistics engineer who can guide the solving process by providing options to be explored, such as which items might be re-located or where the configuration of the warehouse can be modified. A possible approach to solve the problem is using network flow algorithms with additional constraints reflecting properties of valid or preferred assignments.

5.4.5 Solution Realization and Monitoring Module

The Solution Realization and Monitoring Module will be responsible for guiding the logistics engineer through the process of item allocation and possible warehouse layout modification.

In cooperation with the Abstraction Module and Problem Solver, it will provide alternative solutions highlighting the changes with respect to the current status and generate explanations of the proposed solution, collect feedback from the logistics engineer, and possibly initiate re-solving the updated formulation of the optimization problem.

5.4.6 Conversation Management Module

The Conversation Management Module is responsible for commuting the proposed solutions to the user in a visual form and collecting instructions from logistics engineers specifying the task. A chat communication interface is also assumed to convert explanations from the system to natural language and accept instructions from logistics engineers that are easier expressed in language.

5.4.7 Next Steps

The MVP is delivered in the form of Figma designs for Wizard of Oz style AI simulations, with manually prepared warehouse allocation and reconfiguration solutions and initial options for demonstrating AI explanations and accepting user feedback. In the next phase of the project, we plan to move to code implementations of all PEER modules, allowing the users to directly interact with the system in the third round of use case evaluations.

In collaboration with Datacation, a number of design ideas have already been prepared and realized in Figma. These will be adapted both to the results of the lab evaluation with potential end users. Key research directions include:

- Solving the challenging underlying combinatorial optimization problem effectively and efficiently.
- Different (verbal and visual) types of explanations for solutions and the differences between them, to help the user hone in on their preferred solution as quickly as possible.
- Methods to integrate user feedback into the solving process interactively, for example as re-weighted objectives or as additional or relaxed constraints, or in the form of (partial) solutions proposed by the engineers themselves.

6. Lessons Learned

During the project, several challenges emerged in different use cases that we will tackle in the next deliverable iterations. Workshops (WP2) played a crucial role in identifying certain gaps between what can at the moment be realized in PEER and what is expected by the users in terms of the project's end products. We use this as an opportunity to inform the research and potential solutions within this work package, as well as manage expectations.

Sonae

The goal is to develop an AI assistant to generate a personalized plan and help customers navigate an indoor environment for product picking. However, a key limitation of the application is the unavailability of both product and user location data due to technological constraints and privacy concerns. Discussions in workshops helped clarify these challenges, emphasizing the need for alternative approaches to enhance navigation support or to resolve the technological issues in the next deliverables. Due to our Figma approach, this is however not limiting the delivery and evaluation of the MVP.

City of Amsterdam

Workshops were instrumental in gathering user requirements for a personalized route planner tailored to individuals with disabilities. However, the AI algorithms generating these routes rely on datasets that lack certain context features (e.g. weather conditions, real-time public transport information), potentially leading to a misalignment between user expectations and the generated plans which cannot integrate such information. Moving forward, it is essential to identify the best strategies for the municipality of Amsterdam to collect or integrate the missing data, and to manage the expectations of the users with regard to the currently possible scope of the project.

Proditec

The initial user requirements and studies for the Proditec system evolved during MVP development. While these changes did not significantly impact AI development, they provided deeper insights into the real-world implementation of anomaly detection systems. This helped a better understanding of user interactions and requirements, which will ensure better alignment with evolving system needs.

All in all, workshops played a crucial role in identifying these challenges and guiding necessary adjustments. By addressing these limitations, we can better align our AI-driven solutions built in WP3 with real-world expectations and constraints.

Datacation

While the Datacation use case is facilitated by the fact that the intended end users are experts - logistics engineers - who are expected to have a high affinity for technology and a high level of skill in interacting with it, there are also challenges. These include the difficulty of the underlying warehouse allocation problem to be solved, including its mathematical complexity, but also the required domain knowledge of many relevant technical, legal, and business constraints. Moving forward, we are planning to tackle these challenges i) by organizing additional workshops with the logistics engineers, in which we develop an increased mutual understanding of the domain and the possibilities of AI support on the side of the researchers and the engineers, respectively; and ii) by working in parallel on a digital twin of a real warehouse, able to process realistic orders (Datacation), and on simplified and abstracted warehousing problems, which allow us to tackle the fundamental underlying challenges already before having developed deep domain knowledge (research partners).

7. Deviations occurred and corrective actions taken

Continental has been replaced by Datacation; the Datacation use case is fast-tracked as much as possible to catch up with the project within the next year. During the consortium meeting in Porto (September 2024) discussions with Datacation have taken place to have a better understanding of the use case and the stakeholders, and plan WP3 activities. A plan has been set up for that period to catch up with activities through a regular meetings schedule and an additional targeted meetings schedule (for eliciting users' requirements, getting access to data, and planning visits to the warehouse). This deliverable updated with MVP's first release of Datacation in end of June 2025.

Conclusions

A literature review is currently underway as part of Tasks 3.1 to Task 3.3. Along with this literature review, regular discussions with stakeholders across all use cases are helping us to identify key deficiencies in current AI solutions. These findings will guide the development of our collaborative AI assistant, ensuring that it addresses real-world challenges effectively. Key deficiencies include, for example:

- Search and planning methods that are inflexible, that don't adapt well to changing circumstances, and that can't effectively trade off multiple user objectives; recommender systems that are not transparent and insufficiently context-aware; AI agents whose individual decisions as well as overall policies are hard to scrutinize or to intervene on.
- XAI methods that lack interactivity that are not based on human-understandable concepts and on the user's mental model of the task and that are not easily tailored to end-users' changing requirements and level of understanding.
- Conversation management methods that cannot smoothly handle intent recognition, context awareness, and personalization; don’t understand arguments and counterarguments for and against given task solutions; and are inflexible in the level of abstraction used in conversation.

A PEER modular architecture has been developed, aiming at a separation of the agent's components so that they can be developed independently and customized for particular use cases. These components include 1) an Environment Model, 2) a User Model, 3) an Abstraction Module, 4) a Problem Solver, 5) a Solution Realization and Monitoring Module, 6) a Feedback and Learning Module, and 7) a Conversation Management Module.

Initial instantiations of the PEER modular architecture have been developed for four PEER use cases: Sonae (a personalized in-store navigation assistant for customers), the city of Amsterdam (a personalized navigation app for wheelchair users), Proditec (a human-in-the-loop and explainable anomaly detection trainer), and Datacation (AI-assisted warehouse configuration optimization).

As part of these modules, candidate methods and algorithms have been prepared to be simulated in Figma (for Amsterdam, Proditec, and Datacation) or in a simplified simulation environment (for Sonae), respectively. These are ready to be studied with potential end users (via Wizard of Oz approaches where needed), further connecting WP3 tightly to the user workshops done in WP2, the evaluation methodologies developed in WP4, and the implementation work and management of the evaluation process in WP5.

Next steps have been identified both in terms of fundamental research to be done, as well as in terms of additional features to be developed for the use case applications. These include:

- For Sonae, moving beyond myopic solutions based on mixed-integer programming towards more flexible and robust decision-making, e.g. based on reinforcement learning.
- For Amsterdam and Proditec, moving from Figma designs (for Wizard of Oz testing) to interactive prototypes implemented in code.
- For all use cases, the exploration of (additional) explainability and user modelling techniques, informed by the results of the Wizard of Oz evaluations and ongoing research on the promising directions outlined in Section 3.

Finally, a new use case (Datacation) has been onboarded and is progressing well, quickly catching up with the other use cases thanks to an accelerated schedule.

So far, the objectives of WP3 have been fulfilled, and the challenges we have encountered (in particular the change of one use case) are not foreseen to pose any significant risk to the project's goals.

References

[1] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. pearson, 2016.

[2] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[3] R. S. Sutton, A. G. Barto, and others, *Reinforcement learning: An introduction*, vol. 1, no. 1. MIT press Cambridge, 1998.

[4] D. L. Poole and A. K. Mackworth, *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[5] J. Von Neumann and O. Morgenstern, "Theory of games and economic behavior: 60th anniversary commemorative edition," in *Theory of games and economic behavior*, Princeton university press, 2007.

[6] D. Kahneman, "Prospect theory: An analysis of decisions under risk," *Econometrica*, vol. 47, p. 278, 1979.

[7] K. Deb, "Multi-objective optimisation using evolutionary algorithms: an introduction," in *Multi-objective evolutionary optimisation for product design and manufacturing*, Springer, 2011, pp. 3–34.

[8] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[9] B. Friedman and D. G. Hendry, *Value sensitive design: Shaping technology with moral imagination*. Mit Press, 2019.

[10] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[11] R. Bellman, "Dynamic programming," *Science (1979)*, vol. 153, no. 3731, pp. 34–37, 1966.

[12] E. Karpas and D. Magazzeni, "Automated planning for robotics," *Annu Rev Control Robot Auton Syst*, vol. 3, no. 1, pp. 417–439, 2020.

[13] C. Aeronautiques *et al.*, "Pddl| the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[14] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif Intell*, vol. 2, no. 3–4, pp. 189–208, 1971.

[15] E. P. D. Pednault, "ADL: exploring the middle ground between STRIPS and the situation calculus.," *Kr*, vol. 89, pp. 324–332, 1989.

[16] S. Izquierdo-Badiola, G. Canal, C. Rizzo, and G. Alenyà, "Improved task planning through failure anticipation in human-robot collaboration," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7875–7880.

[17] S. Izquierdo-Badiola, G. Canal, G. Alenyà, C. Rizzo, and A. Coles, "Planning for Human-Robot Collaboration Scenarios with Heterogeneous Costs and Durations," in *ECAI 2024*, 2024.

[18] D. Klein and C. D. Manning, "A* parsing: Fast exact Viterbi parse selection," in *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003, pp. 119–126.

[19] M. L. Hetland, *Python Algorithms: mastering basic algorithms in the Python Language*. Apress, 2014.

[20] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, 2005, pp. 117–122.

[21] P. Bagad, P. Dahatonde, R. Gagare, I. Athare, and G. Ghule, "Optimizing Pathfinding in Dynamic Environments: A Comparative Study of A* and D-Lite Algorithms," in *2024 IEEE Pune Section International Conference (PuneCon)*, 2024, pp. 1–11.

[22] G. M. J. Chaslot, M. H. M. Winands, H. J. van den Herik, J. W. H. M. Uiterwijk, and B. Bouzy, "Progressive strategies for Monte-Carlo tree search," *New Mathematics and Natural Computation*, vol. 4, no. 03, pp. 343–357, 2008.

[23] T. Miller, "Monte-Carlo Tree Search (MCTS)," 2016. [Online]. Available: https://gibberblot.github.io/rl-notes/single-agent/mcts.html

[24] A. Pettet *et al.*, "Decision making in non-stationary environments with policy-augmented search," *arXiv preprint arXiv:2401.03197*, 2024.

[25] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, “Anytime dynamic A*: An anytime, replanning algorithm.,” in *ICAPS*, 2005, pp. 262–271.

[26] A. Oyola, D. G. Romero, and B. X. Vintimilla, “A Dijkstra-based algorithm for selecting the Shortest-Safe Evacuation Routes in dynamic environments (SSER),” in *Advances in Artificial Intelligence: From Theory to Practice: 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, Arras, France, June 27-30, 2017, Proceedings, Part I 30*, 2017, pp. 131–135.

[27] J. Achiam, “Spinning Up documentation,” 2018. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html

[28] L. Begnardi, B. Meijenfeldt, H. Baier, W. Jaarsveld, and Y. Zhang, “Algorithm Configuration in Sequential Decision-Making,” in *Proceedings of the 22nd International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2025)*, 2025.

[29] E. Schede *et al.*, “A survey of methods for automated algorithm configuration,” *Journal of Artificial Intelligence Research*, vol. 75, pp. 425–487, 2022.

[30] S. Adriaensen *et al.*, “Automated dynamic algorithm configuration,” *Journal of Artificial Intelligence Research*, vol. 75, pp. 1633–1699, 2022.

[31] D. Silver *et al.*, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[32] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for decision making*. MIT press, 2022.

[33] R. Ortner, P. Gajane, and P. Auer, “Variational regret bounds for reinforcement learning,” in *Uncertainty in Artificial Intelligence*, 2020, pp. 81–90.

[34] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *European conference on machine learning*, 2006, pp. 282–293.

[35] G. Pettet, A. Mukhopadhyay, M. J. Kochenderfer, and A. Dubey, “Hierarchical planning for dynamic resource allocation in smart and connected communities,” *ACM Transactions on Cyber-Physical Systems*, vol. 6, no. 4, pp. 1–26, 2022.

[36] C. F. Hayes *et al.*, “A practical guide to multi-objective reinforcement learning and planning,” *Auton Agent Multi Agent Syst*, vol. 36, no. 1, p. 26, 2022.

[37] T. Anthony, Z. Tian, and D. Barber, “Thinking fast and slow with deep learning and tree search,” *Adv Neural Inf Process Syst*, vol. 30, 2017.

[38] W. Röpke, D. M. Roijers, A. Nowé, R. Rădulescu, and H. Baier, “Model-Based Reinforcement Learning in Multi-Objective Environments with a Distributional Critic,” in *Multi-objective Decision Making Workshop at ECAI 2024*, 2024.

[39] M. G. Bellemare, W. Dabney, and M. Rowland, *Distributional reinforcement learning*. MIT Press, 2023.

[40] S. Gronauer and K. Diepold, “Multi-agent deep reinforcement learning: a survey,” *Artif Intell Rev*, vol. 55, no. 2, pp. 895–943, 2022.

[41] A. Patterson, S. Neumann, M. White, and A. White, “Empirical design in reinforcement learning,” *Journal of Machine Learning Research*, vol. 25, no. 318, pp. 1–63, 2024.

[42] M. Towers *et al.*, “Gymnasium: A standard interface for reinforcement learning environments,” *arXiv preprint arXiv:2407.17032*, 2024.

[43] J. Terry *et al.*, “Pettingzoo: Gym for multi-agent reinforcement learning,” *Adv Neural Inf Process Syst*, vol. 34, pp. 15032–15043, 2021.

[44] L. N. Alegre *et al.*, “MO-Gym: A library of multi-objective reinforcement learning environments,” in *Proceedings of the 34th Benelux Conference on Artificial Intelligence BNAIC/Benelearn*, 2022, p. 2.

[45] V. Tkachuk, S. A. Bakhtiari, J. Kirschner, M. Jusup, I. Bogunovic, and C. Szepesvári, “Efficient planning in combinatorial action spaces with applications to cooperative multi-agent reinforcement learning,” in *International Conference on Artificial Intelligence and Statistics*, 2023, pp. 6342–6370.

[46] H. Xiao and P. Wang, “Llm a*: Human in the loop large language models enabled a* search for robotics,” *arXiv preprint arXiv:2312.01797*, 2023.

[47] S. Izquierdo-Badiola, G. Canal, C. Rizzo, and G. Alenyà, "Plancollabnl: Leveraging large language models for adaptive plan generation in human-robot collaboration," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 17344–17350.

[48] T. Chakraborti *et al.*, "Planning for serendipity," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5300–5306.

[49] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "Human-aware planning revisited: A tale of three models," in *IJCAI-ECAI XAI/ICAPS XAIP Workshops*, 2018.

[50] S. Shekhar, A. Favier, and R. Alami, "An Epistemic Human-Aware Task Planner which Anticipates Human Beliefs and Decisions," *arXiv preprint arXiv:2409.18545*, 2024.

[51] G. Buisan, A. Favier, A. Mayima, and R. Alami, "HATP/EHDA: A robot task planner anticipating and eliciting human decisions and actions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2818–2824.

[52] G. Canal, G. Alenyà, and C. Torras, "Adapting robot task planning to user preferences: an assistive shoe dressing example," *Auton Robots*, vol. 43, no. 6, pp. 1343–1356, 2019.

[53] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robot Autom Lett*, vol. 6, no. 2, pp. 1136–1143, 2021.

[54] G. Buisan and R. Alami, "A human-aware task planner explicitly reasoning about human and robot decision, action and reaction," in *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 544–548.

[55] V. V Unhelkar, S. Li, and J. A. Shah, "Decision-making for bidirectional communication in sequential human-robot collaborative tasks," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 329–341.

[56] B. Zhang and H. Soh, "Large language models as zero-shot human models for human-robot interaction," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 7961–7968.

[57] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023, pp. 1–22.

[58] J. Wu *et al.*, "Tidybot: Personalized robot assistance with large language models," *Auton Robots*, vol. 47, no. 8, pp. 1087–1102, 2023.

[59] D. Han, T. McInroe, A. Jelley, S. V Albrecht, P. Bell, and A. Storkey, "LLM-Personalize: Aligning LLM planners with human preferences via reinforced self-training for housekeeping robots," *arXiv preprint arXiv:2404.14285*, 2024.

[60] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[61] B. Goodman and S. Flaxman, "European Union regulations on algorithmic decision-making and a 'right to explanation,'" *AI Mag*, vol. 38, no. 3, pp. 50–57, 2017.

[62] P. Hacker and J.-H. Passoth, "Varieties of AI explanations under the law. From the GDPR to the AIA, and beyond," in *International workshop on extending explainable AI beyond deep models and classifiers*, 2020, pp. 343–373.

[63] A. Bertrand, T. Viard, R. Belloum, J. R. Eagan, and W. Maxwell, "On selective, mutable and dialogic XAI: A review of what users say about different types of interactive explanations," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.

[64] G. Bansal *et al.*, "Does the whole exceed its parts? the effect of ai explanations on complementary team performance," in *Proceedings of the 2021 CHI conference on human factors in computing systems*, 2021, pp. 1–16.

[65] C. Gomez, S. M. Cho, S. Ke, C.-M. Huang, and M. Unberath, "Human-AI collaboration is not very collaborative yet: A taxonomy of interaction patterns in AI-assisted decision making from a systematic review," *Front Comput Sci*, vol. 6, p. 1521066, 2025.

[66] V. Lai, C. Chen, A. Smith-Renner, Q. V. Liao, and C. Tan, "Towards a science of human-AI decision making: An overview of design space in empirical human-subject studies," in *Proceedings of the 2023 ACM conference on fairness, accountability, and transparency*, 2023, pp. 1369–1385.

[67] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Adv Neural Inf Process Syst*, vol. 30, 2017.

[68] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘ Why should i trust you?’ Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[69] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[70] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, “Gradient-based attribution methods,” *Explainable AI: Interpreting, explaining and visualizing deep learning*, pp. 169–191, 2019.

[71] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, “This looks like that: deep learning for interpretable image recognition,” *Adv Neural Inf Process Syst*, vol. 32, 2019.

[72] D. Rymarczyk, Ł. Struski, J. Tabor, and B. Zieliński, “Protopshare: Prototypical parts sharing for similarity discovery in interpretable image classification,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1420–1430.

[73] D. Rymarczyk, Ł. Struski, M. Górszczak, K. Lewandowska, J. Tabor, and B. Zieliński, “Interpretable image classification with differentiable prototypes assignment,” in *European Conference on Computer Vision*, 2022, pp. 351–368.

[74] M. Nauta, R. Van Bree, and C. Seifert, “Neural prototype trees for interpretable fine-grained image recognition,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14933–14943.

[75] M. Espinosa Zarlenga *et al.*, “Concept embedding models: Beyond the accuracy-explainability trade-off,” *Adv Neural Inf Process Syst*, vol. 35, pp. 21400–21413, 2022.

[76] T. Chiaburu, F. Haußer, and F. Bießmann, “Copronn: Concept-based prototypical nearest neighbors for explaining vision models,” in *World Conference on Explainable Artificial Intelligence*, 2024, pp. 69–91.

[77] P. W. Koh *et al.*, “Concept bottleneck models,” in *International conference on machine learning*, 2020, pp. 5338–5348.

[78] B. Wang, L. Li, Y. Nakashima, and H. Nagahara, “Learning bottleneck concepts in image classification,” in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2023, pp. 10962–10971.

[79] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” *Artif Intell*, vol. 267, pp. 1–38, 2019.

[80] S. Bobek *et al.*, “User-centric evaluation of explainability of AI with and for humans: a comprehensive empirical study,” *arXiv preprint arXiv:2410.15952*, 2024.

[81] M. Ghassemi, L. Oakden-Rayner, and A. L. Beam, “The false hope of current approaches to explainable artificial intelligence in health care,” *Lancet Digit Health*, vol. 3, no. 11, pp. e745–e750, 2021.

[82] S. Verma, A. Lahiri, J. P. Dickerson, and S.-I. Lee, “Pitfalls of explainable ML: an industry perspective,” *arXiv preprint arXiv:2106.07758*, 2021.

[83] J. Roski, E. J. Maier, K. Vigilante, E. A. Kane, and M. E. Matheny, “Enhancing trust in AI through industry self-governance,” *Journal of the American Medical Informatics Association*, vol. 28, no. 7, pp. 1582–1590, 2021.

[84] T. Evans *et al.*, “The explainability paradox: Challenges for xAI in digital pathology,” *Future Generation Computer Systems*, vol. 133, pp. 281–296, 2022.

[85] U. Ehsan and M. O. Riedl, “Explainability pitfalls: Beyond dark patterns in explainable AI,” *Patterns*, vol. 5, no. 6, 2024.

[86] C. Molnar *et al.*, “General pitfalls of model-agnostic interpretation methods for machine learning models,” in *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, 2020, pp. 39–68.

[87] F. Mujkanovic, V. Doskoč, M. Schirneck, P. Schäfer, and T. Friedrich, “timeXplain–A Framework for Explaining the Predictions of Time Series Classifiers,” *arXiv preprint arXiv:2007.07606*, 2020.

[88] D. Collaris and J. J. van Wijk, “ExplainExplore: Visual exploration of machine learning explanations,” in *2020 IEEE Pacific Visualization Symposium (PacificVis)*, 2020, pp. 26–35.
[89] M. Fuji, H. Morita, K. Goto, K. Maruhashi, H. Anai, and N. Igata, “Explainable AI through combination of deep tensor and knowledge graph,” *Fujitsu Sci. Tech. J*, vol. 55, no. 2, pp. 58–64, 2019.
[90] H. Baniecki, D. Parzych, and P. Biecek, “The grammar of interactive explanatory model analysis,” *Data Min Knowl Discov*, vol. 38, no. 5, pp. 2596–2632, 2024.
[91] G. Guo, D. Arendt, and A. Endert, “Explainability in JupyterLab and Beyond: Interactive XAI Systems for Integrated and Collaborative Workflows,” *arXiv preprint arXiv:2404.02081*, 2024.
[92] M. Nazar, M. M. Alam, E. Yafi, and M. M. Su’ud, “A systematic review of human–computer interaction and explainable artificial intelligence in healthcare with artificial intelligence techniques,” *IEEE Access*, vol. 9, pp. 153316–153348, 2021.
[93] Y. Dang *et al.*, “Explainable and interpretable multimodal large language models: A comprehensive survey,” *arXiv preprint arXiv:2412.02104*, 2024.
[94] H. Baier and M. Kaisers, “Towards explainable MCTS,” in *2021 AAAI Workshop on Explainable Agency in AI*, 2021, p. 178.
[95] B. Hayes and J. A. Shah, “Improving robot controller transparency through autonomous policy explanation,” in *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction*, 2017, pp. 303–312.
[96] O. Khan, P. Poupart, and J. Black, “Minimal sufficient explanations for factored markov decision processes,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2009, pp. 194–200.
[97] S. Milani, N. Topin, M. Veloso, and F. Fang, “Explainable reinforcement learning: A survey and comparative review,” *ACM Comput Surv*, vol. 56, no. 7, pp. 1–36, 2024.
[98] D. Claes, F. Oliehoek, H. Baier, and K. Tuyls, “Decentralised online planning for multi-robot warehouse commissioning,” in *AAMAS’17: PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS*, 2017, pp. 492–500.
[99] Z. An, H. Baier, A. Dubey, A. Mukhopadhyay, and M. Ma, “Enabling mcts explainability for sequential planning through computation tree logic,” in *ECAI 2024*, IOS Press, 2024, pp. 4068–4075.
[100] S. Shah, M. Lowalekar, and P. Varakantham, “Neural approximate dynamic programming for on-demand ride-pooling,” in *Proceedings of the AAAI conference on artificial intelligence*, 2020, pp. 507–515.
[101] A. Ziyan *et al.*, “Combining LLMs with a Logic-Based Framework to Explain MCTS,” in *24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025)*, May 2025.
[102] F. Cruz, R. Dazeley, and P. Vamplew, “Memory-based explainable reinforcement learning,” in *AI 2019: Advances in Artificial Intelligence: 32nd Australasian Joint Conference, Adelaide, SA, Australia, December 2–5, 2019, Proceedings 32*, 2019, pp. 66–77.
[103] J. van der Waa, J. van Diggelen, K. van den Bosch, and M. Neerincx, “Contrastive explanations for reinforcement learning in terms of expected consequences,” *arXiv preprint arXiv:1807.08706*, 2018.
[104] Z. Ye, S. Milani, G. J. Gordon, and F. Fang, “Concept-based interpretable reinforcement learning with limited to no human labels,” *arXiv preprint arXiv:2407.15786*, 2024.
[105] G. Shafer and V. Vovk, “A tutorial on conformal prediction.,” *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.
[106] M. Yuksekgonul, M. Wang, and J. Zou, “Post-hoc concept bottleneck models,” *arXiv preprint arXiv:2205.15480*, 2022.
[107] B. Kim *et al.*, “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav),” in *International conference on machine learning*, 2018, pp. 2668–2677.
[108] S. Yamaguchi and K. Nishida, “Explanation Bottleneck Models,” *arXiv preprint arXiv:2409.17663*, 2024.
[109] B. Bonet, H. Palacios, and H. Geffner, “Automatic derivation of finite-state machines for behavior control,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010, pp. 1656–1659.
[110] J. P. Inala, O. Bastani, Z. Tavares, and A. Solar-Lezama, “Synthesizing programmatic policies that inductively generalize,” in *8th International Conference on Learning Representations*, 2020.

[111] A. Verma, V. Murali, R. Singh, P. Kohli, and S. Chaudhuri, “Programmatically interpretable reinforcement learning,” in *International conference on machine learning*, 2018, pp. 5045–5054.
[112] Q. A. Sadmine, H. Baier, and L. Lelis, “Language models speed up local search for finding programmatic policies,” *Transactions on Machine Learning Research*, vol. 20, no. X, 2024.
[113] M.-A. Zöller, W. Titov, T. Schlegel, and M. F. Huber, “XAutoML: a visual analytics tool for understanding and validating automated machine learning,” *ACM Trans Interact Intell Syst*, vol. 13, no. 4, pp. 1–39, 2023.
[114] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.
[115] P. Moritz *et al.*, “Ray: A distributed framework for emerging {AI} applications,” in *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, 2018, pp. 561–577.
[116] C. Wang, Q. Wu, M. Weimer, and E. Zhu, “Flaml: A fast and lightweight automl library,” *Proceedings of Machine Learning and Systems*, vol. 3, pp. 434–447, 2021.
[117] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Adv Neural Inf Process Syst*, vol. 24, 2011.
[118] S. Falkner, A. Klein, and F. Hutter, “BOHB: Robust and efficient hyperparameter optimization at scale,” in *International conference on machine learning*, 2018, pp. 1437–1446.
[119] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms,” in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 847–855.
[120] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *Learning and intelligent optimization: 5th international conference, LION 5, rome, Italy, January 17-21, 2011. selected papers 5*, 2011, pp. 507–523.
[121] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Adv Neural Inf Process Syst*, vol. 25, 2012.
[122] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. pearson, 2016.
[123] P. Viappiani, “Preference Modeling and Preference Elicitation: An Overview.,” *DMRS*, pp. 19–24, 2014.
[124] M. Balabanović and Y. Shoham, “Fab: content-based, collaborative recommendation,” *Commun ACM*, vol. 40, no. 3, pp. 66–72, 1997.
[125] J. Ben Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” in *The adaptive web: methods and strategies of web personalization*, Springer, 2007, pp. 291–324.
[126] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in artificial intelligence*, vol. 2009, no. 1, p. 421425, 2009.
[127] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “BPR: Bayesian personalized ranking from implicit feedback,” *arXiv preprint arXiv:1205.2618*, 2012.
[128] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM computing surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.
[129] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.
[130] B. Hidasi and A. Karatzoglou, “Recurrent neural networks with top-k gains for session-based recommendations,” in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 843–852.
[131] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer (Long Beach Calif)*, vol. 42, no. 8, pp. 30–37, 2009.
[132] S. Rendle, “Factorization machines,” in *2010 IEEE International conference on data mining*, 2010, pp. 995–1000.
[133] Q. Du, N. Li, S. Yang, D. Sun, and W. Liu, “Integrating KNN and gradient boosting decision tree for recommendation,” in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2021, pp. 2042–2049.

[134] D. Malitesta, C. Pomo, V. W. Anelli, A. C. M. Mancino, E. Di Sciascio, and T. Di Noia, “A topology-aware analysis of graph collaborative filtering,” *arXiv preprint arXiv:2308.10778*, 2023.
[135] T. Hekmatfar, S. Haratizadeh, P. Razban, and S. Goliaei, “Attention-based recommendation on graphs,” *arXiv preprint arXiv:2201.05499*, 2022.
[136] C. Singh, J. P. Inala, M. Galley, R. Caruana, and J. Gao, “Rethinking interpretability in the era of large language models,” *arXiv preprint arXiv:2402.01761*, 2024.
[137] J. Wei *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Adv Neural Inf Process Syst*, vol. 35, pp. 24824–24837, 2022.
[138] S. Raza, “Bias reduction news recommendation system,” *Digital*, vol. 4, no. 1, pp. 92–103, 2023.
[139] A. Y. Ng, S. Russell, and others, “Algorithms for inverse reinforcement learning.,” in *Icml*, 2000, p. 2.
[140] M. Barnes *et al.*, “Massively scalable inverse reinforcement learning in google maps,” *arXiv preprint arXiv:2305.11290*, 2023.
[141] B. Settles, “Active learning literature survey,” 2009.
[142] J. M. Rožanec *et al.*, “Human in the AI loop via xAI and active learning for visual inspection,” *Artificial Intelligence in Manufacturing*, p. 381, 2024.
[143] C. C. Bonwell and J. A. Eison, *Active learning: Creating excitement in the classroom. 1991 ASHE-ERIC higher education reports*. ERIC, 1991.
[144] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-R\'\ios, J. Bobes-Bascarán, and Á. Fernández-Leal, “Human-in-the-loop machine learning: a state of the art,” *Artif Intell Rev*, vol. 56, no. 4, pp. 3005–3054, 2023.
[145] R. M. Monarch, *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster, 2021.
[146] T. T. Hills, P. M. Todd, D. Lazer, A. D. Redish, and I. D. Couzin, “Exploration versus exploitation in space, mind, and society,” *Trends Cogn Sci*, vol. 19, no. 1, pp. 46–54, 2015.
[147] M. Tkalčič, B. De Carolis, M. De Gemmis, A. Odić, and A. Košir, “Emotions and personality in personalized services,” *Human-Computer Interaction Series. Elsevier*, 2016.
[148] A. Calero Valdez and M. Ziefle, “Human factors in the age of algorithms. understanding the human-in-the-loop using agent-based modeling,” in *Social Computing and Social Media. Technologies and Analytics: 10th International Conference, SCSM 2018, Held as Part of HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part II 10*, 2018, pp. 357–371.
[149] D. S. Nunes, P. Zhang, and J. S. Silva, “A survey on human-in-the-loop applications towards an internet of all,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 944–965, 2015.
[150] K. Kutt, M. Kutt, B. Kawa, and G. J. Nalepa, “Human-in-the-loop for personality dynamics: proposal of a new research approach,” in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, 2024, pp. 455–464.
[151] R. Lara-Cabrera and D. Camacho, “A taxonomy and state of the art revision on affective games,” *Future Generation Computer Systems*, vol. 92, pp. 516–525, 2019.
[152] A. Pyae, “The Human-AI Handshake Framework: A Bidirectional Approach to Human-AI Collaboration,” *arXiv preprint arXiv:2502.01493*, 2025.
[153] D. Jannach, A. Manzoor, W. Cai, and L. Chen, “A survey on conversational recommender systems,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.
[154] L. Tudor Car *et al.*, “Conversational agents in health care: scoping review and conceptual analysis,” *J Med Internet Res*, vol. 22, no. 8, p. e17158, 2020.
[155] I. J. Akpan, Y. M. Kobara, J. Owolabi, A. A. Akpan, and O. F. Offodile, “Conversational and generative artificial intelligence and human–chatbot interaction in education and research,” *International Transactions in Operational Research*, vol. 32, no. 3, pp. 1251–1281, 2025.
[156] Y. Cheng *et al.*, “AutoPal: Autonomous Adaptation to Users for Personal AI Companionship,” *arXiv preprint arXiv:2406.13960*, 2024.
[157] T. Ait Baha, M. El Hajji, Y. Es-Saady, and H. Fadili, “The power of personalization: A systematic review of personality-adaptive chatbots,” *SN Comput Sci*, vol. 4, no. 5, p. 661, 2023.

[158] R. Ahmad, D. Siemon, U. Gnewuch, and S. Robra-Bissantz, “Designing personality-adaptive conversational agents for mental health care,” *Information Systems Frontiers*, vol. 24, no. 3, pp. 923–943, 2022.
[159] L. Longo *et al.*, “Explainable Artificial Intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions,” *Information Fusion*, vol. 106, p. 102301, 2024.
[160] G. A. Santos, G. G. De Andrade, G. R. S. Silva, F. C. M. Duarte, J. P. J. Da Costa, and R. T. de Sousa, “A conversation-driven approach for chatbot management,” *IEEE Access*, vol. 10, pp. 8474–8486, 2022.
[161] D. A. Dhinagaran *et al.*, “Designing, developing, evaluating, and implementing a smartphone-delivered, rule-based conversational agent (DISCOVER): development of a conceptual framework,” *JMIR Mhealth Uhealth*, vol. 10, no. 10, p. e38740, 2022.
[162] M. McTear, *Conversational ai: Dialogue systems, conversational agents, and chatbots*. Springer Nature, 2022.
[163] J. Quan *et al.*, “Integrating pre-trained model into rule-based dialogue management,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 16097–16099.
[164] M. Free, A. Langworthy, M. Dimitropoulaki, and S. Thompson, “Towards Goal-Oriented Agents for Evolving Problems Observed via Conversation,” in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2023, pp. 142–155.
[165] S. Verma, J. Fu, M. Yang, and S. Levine, “Chai: A chatbot ai for task-oriented dialogue with offline reinforcement learning,” *arXiv preprint arXiv:2204.08426*, 2022.
[166] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.
[167] A. H. Marani, U. Schnaithmann, Y. Son, A. Iyer, M. Paldhe, and A. Raghuvanshi, “Graph Integrated Language Transformers for Next Action Prediction in Complex Phone Calls,” *arXiv preprint arXiv:2404.08155*, 2024.
[168] D. H. Vicci, “Emotional intelligence in artificial intelligence: A review and evaluation study,” *Available at SSRN 4818285*, 2024.
[169] N. A. Helal, A. Hassan, N. L. Badr, and Y. M. Afify, “A contextual-based approach for sarcasm detection,” *Sci Rep*, vol. 14, no. 1, p. 15415, 2024.
[170] A. Kalloniatis and P. Adamidis, “Computational humor recognition: a systematic literature review,” *Artif Intell Rev*, vol. 58, no. 2, p. 43, 2024.
[171] K. Čyras, A. Rago, E. Albini, P. Baroni, and F. Toni, “Argumentative XAI: a survey,” *arXiv preprint arXiv:2105.11266*, 2021.
[172] A. Shih, A. Choi, and A. Darwiche, “A symbolic approach to explaining bayesian network classifiers,” *arXiv preprint arXiv:1805.03364*, 2018.
[173] S. Sreedharan, T. Chakraborti, and S. Kambhampati, “The emerging landscape of explainable automated planning & decision making,” in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 4803–4811.
[174] S. Modgil *et al.*, “The added value of argumentation,” *Agreement technologies*, pp. 357–403, 2013.
[175] P. M. Dung, “An argumentation-theoretic foundation for logic programming,” *The Journal of logic programming*, vol. 22, no. 2, pp. 151–177, 1995.
[176] S. Kraus, K. Sycara, and A. Evenchik, “Reaching agreements through argumentation: a logical model and implementation,” *Artif Intell*, vol. 104, no. 1–2, pp. 1–69, 1998.
[177] D. Lewis, “Causation as influence,” *J Philos*, vol. 97, no. 4, pp. 182–197, 2000.
[178] K. Sokol and P. Flach, “Explainability fact sheets: A framework for systematic assessment of explainable approaches,” in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 56–67.
[179] W. Pedrycz and W. Pedrycz, “Information Granules and Granular Computing,” *An Introduction to Computing with Fuzzy Sets: Analysis, Design, and Applications*, pp. 1–11, 2021.
[180] C. P. Bonini, “Simulation of information and decision systems in the firm,” *(No Title)*, 1967.

[181] G. A. Gorry and M. S. Scott Morton, “A framework for management information systems,” 1971.
[182] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and others, *Network flows: theory, algorithms, and applications*, vol. 1. Prentice hall Englewood Cliffs, NJ, 1993.
[183] Y. Cui, Z. Liu, and S. Lian, “A survey on unsupervised anomaly detection algorithms for industrial images,” *IEEE Access*, vol. 11, pp. 55297–55315, 2023.
[184] 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.
[185] N. A. Ahuja, I. Ndiour, T. Kalyanpur, and O. Tickoo, “Probabilistic modeling of deep features for out-of-distribution and adversarial detection. arXiv 2019,” *arXiv preprint arXiv:1909.11786*.
[186] T. Defard, A. Setkov, A. Loesch, and R. Audigier, “Padim: a patch distribution modeling framework for anomaly detection and localization,” in *International conference on pattern recognition*, 2021, pp. 475–489.

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.

