Unpacking Human-AI interactions
From Interaction Primitives to a Design Space

Tsiakas, Konstantinos; Murray-Rust, Dave

DOI
10.1145/3664522

Publication date
2024

Document Version
Final published version

Published in
ACM Transactions on Interactive Intelligent Systems

Unpacking Human-AI interactions: From Interaction Primitives to a Design Space

KONSTANTINOS TSIAKAS and DAVE MURRAY-RUST, Delft University of Technology, Delft, The Netherlands

This article aims to develop a semi-formal representation for Human-AI (HAI) interactions, by building a set of interaction primitives which can specify the information exchanges between users and AI systems during their interaction. We show how these primitives can be combined into a set of interaction patterns which can capture common interactions between humans and AI/ML models. The motivation behind this is twofold: firstly, to provide a compact generalization of existing practices for the design and implementation of HAI interactions; and secondly, to support the creation of new interactions by extending the design space of HAI interactions. Taking into consideration frameworks, guidelines, and taxonomies related to human-centered design and implementation of AI systems, we define a vocabulary for describing information exchanges based on the model's characteristics and interactional capabilities. Based on this vocabulary, a message passing model for interactions between humans and models is presented, which we demonstrate can account for existing HAI interaction systems and approaches. Finally, we build this into design patterns which can describe common interactions between users and models, and we discuss how this approach can be used toward a design space for HAI interactions that creates new possibilities for designs as well as keeping track of implementation issues and concerns.

CCS Concepts: • **Human-centered computing** → **Interaction design**; *Interaction paradigms*; • **Computing methodologies** → *Artificial intelligence*;

Additional Key Words and Phrases: Human-AI interaction, interaction patterns, explainable AI, human-in-the-loop, hybrid intelligence

ACM Reference format:
Konstantinos Tsiakas and Dave Murray-Rust. 2024. Unpacking Human-AI interactions: From Interaction Primitives to a Design Space. *ACM Trans. Interact. Intell. Syst.* 14, 3, Article 18 (August 2024), 51 pages.
https://doi.org/10.1145/3664522

1 Introduction

Artificial Intelligence (AI) and **Machine Learning (ML)** models are in the midst of a transition from use in back-end data science systems, operated and interacted with by experts, to end-user-focused applications that prioritize ease of use and interactional fluidity [104]. This transition to **Human-Centered AI (HCAI)** is driven by several factors—the increasing power of the models and systems; the need for more engagement with real-world data; the enlisting of users as participants in model decision-making and development; and the corresponding need to make sure that the models

Authors' Contact Information: Konstantinos Tsiakas, Delft University of Technology, Delft, The Netherlands; e-mail: tsiakas.konstantinos@gmail.com; Dave Murray-Rust (Corresponding author), Delft University of Technology, Delft, The Netherlands; e-mail: D.S.Murray-Rust@tudelft.nl.

are operating correctly or adapt them for particular tasks. This has given rise to a growing set of HCAI methods, in particular **Human-in-the-Loop (HITL)** [18, 69], **Explainable AI (XAI)** [55, 83], and **Hybrid Intelligence (HI)** methods [25, 102]. One of the main challenges is the seamless integration of human and technological capabilities into a socio-technical system [12] which can involve different types of human and artificial agents with different goals and intentions. The wide range of *human-centered design and implementation methods* coupled with the need to understand how systems grow and change over time [36] and affect diverse stakeholders [16] extends the design space and thus leads to a need for new ways to design **Human-AI (HAI)** *interactions* [105].

Despite the computational advances and capabilities of AI/ML models, designing interactions between users and models remains a challenging task. Designers often have a broad understanding of the possibilities of AI/ML, but lack specifics [30]. This is related to two AI attributes: *capability uncertainty*, which indicates the uncertainty of what the possibilities of an ML model are, and *output complexity*, which covers the general difficulty of working with multidimensional, rich outputs from large systems [108]. Envisioning new AI-based solutions for a given **User Experience (UX)** problem should consider that HAI interactions need to adjust to different users and evolve over time [36]. Going beyond one-to-one interactions with systems, there is a need to investigate the effects of algorithmic systems on the different groups and types of users involved in or affected by AI decisions [16], such as in organizational decision-making [12]. Human oversight is an important aspect which enables users to maintain human agency and accountability, by participating in the decision-making process. From such a *socio-technical* perspective, hybrid decision-making can involve both decision-makers and decision-subjects toward interactions with contestable AI models [7]. Designing for hybrid decision-making can be challenging while considering the legal, social, technical, and organizational issues [32]. Moreover, designing AI-based systems that facilitate *meaningful human control* requires the implementation of decision-making methods in order to address responsibility gaps related to hybrid decisions [17].

Responses to this complexity and the need for human-centered design of AI systems takes several forms, including design frameworks, guidelines, and practices for HAI interactions. Design frameworks and guidelines shape the way that people approach designing HAI interaction systems and can support practitioners through a set of design methods, practices, and examples for designing AI systems. Shneiderman's HCAI framework looks to create reliable, safe, and trustworthy HAI interactions through active participation, with the intent to achieve a high level of human control, while maintaining a high level of computer automation [85]. Xu's HAI framework sets out three key components for system design: *technology enhancement*, *ethically aligned design*, and *human factors design*, highlighting the need to design responsible and reliable AI-based solutions [104]. Yurrita et al.'s multi-stakeholder framework looks at how human values connect to the properties of AI/ML systems [110]. Design frameworks are also used to address challenges related to *Responsible AI* by integrating ethical analysis into engineering practice [74, 112]. **Research through Design (RtD)** has been proposed as an approach to ensure that the role of AI in a system is legible to the end users [57]. RtD activities include sketching and prototyping of interactions to support the ideation and design of HAI interactions. At a higher conceptual level, metaphors can affect expectations of system's performance [48], help designers to understand key concepts [29, 67], or lead to envisioning new kinds of relations between humans and technology [58]. From a technical perspective, AI practitioners often need to translate human-understandable concepts into functional blocks toward the development and deployment of ethical and accountable models [53].

The motivation of this article is the need to bridge this gap between engineering and design practices for HCAI system design (Figure 1). To address this, we propose *a semi-formal representation for HAI interactions* which describes interactions based on the low-level information exchanges between users and models. Our proposed approach aims to be more pragmatic than

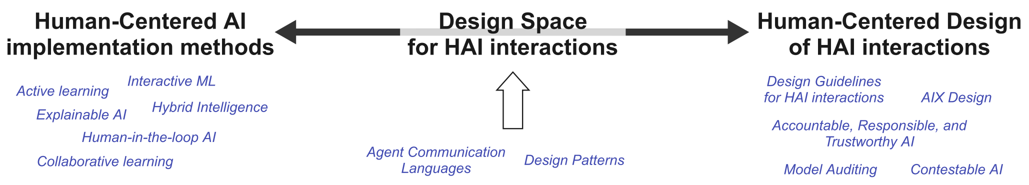

Human-Centered AI implementation methods
Design Space for HAI interactions
Human-Centered Design of HAI interactions
Active learning
Interactive ML
Explainable AI
Hybrid Intelligence
Human-in-the-loop AI
Collaborative learning
Agent Communication Languages
Design Patterns
Design Guidelines for HAI interactions
AIX Design
Accountable, Responsible, and Trustworthy AI
Model Auditing
Contestable AI

Fig. 1. Our proposed design space for HAI interactions as a link between implementation and design methods for HCAI.

high-level frameworks, as it works from describing interactions based on user-model information exchanges, providing a link from guidelines and design practices to the actual implementation and prototyping of existing and new types of interaction. It attempts to give designers and developers more direct facility to engage with the interactional capabilities of the models by providing a palette of possibilities that map understandable human concepts to specific exchanges of information. We draw inspiration from **Agent Communication Languages (ACLs)** [e.g., 71, 34] which use communicative acts and messages to enable representation and exchange of knowledge between agents [2, 3]. Working from this, the intuition behind this article is that a small set of *interaction primitives* can be combined into a collection of *interaction patterns* used to design more complex HAI interactions and systems. Information exchanges can then be translated in human-understandable concepts, including providing training examples to a model, validating a model's prediction or requesting explanations for a given decision. Such interaction primitives can be combined into patterns of interaction describing higher-level interactions, such as requesting explanations to validate a model's decision or negotiate with a model until an agreement is met. Such a specification would help to imagine richer interactions with models and could allow a broader range of people to take part in the design of HAI interactions.

Our approach aims to identify the possible types of information exchanges between users and models and formalize them as modeling elements for HAI interactions. Toward this, we review frameworks and guidelines for HAI interactions, as well as implementation methods and examples of HAI interactions, toward identifying a common set of communicative acts between users and AI models. Following an unpacking approach, we decompose existing HAI interactions into a set of descriptive actions and patterns, based on which we define our formalization for HAI interaction primitives. Our motivation for defining a design space based on HAI interaction primitives is to (a) provide a space in which existing interaction concepts and paradigms can be represented, bridging the gap between human and machine understanding of what an interaction entails, (b) allow for the potential invention of new kinds of interactions through exploring the space of communicative actions and interaction patterns, and (c) provide a specification which carries the required information needed for the implementation of HAI interactions.

The key contributions of the article are as follows:

—A semi-formal representation of HAI interactions which describes the information exchanges between users and models using communicative acts and
—A collection of interaction patterns which can (a) reflect existing practices and create new forms of HAI interactions and (b) characterize interactions in terms of their design and implementation aspects.

2 Background and Related Work

The aim of this research is to develop a semi-formal modeling language for HAI interactions. In order to do so, we need to identify how to formally represent aspects related to the design and implementation of HAI interactions. We begin with an overview of existing *HAI interaction paradigms*, focusing on the information exchanges between users and models. Following these paradigms, we searched articles related to (a) design aspects and (b) implementation approaches for HAI interactions. For design methods, we provide an overview of existing guidelines and practices, based on which we extract a subset of design considerations for transparency and user control. Then, we explore how computational methods can be used to implement HAI interactions, in order to identify the low-level information exchanges that take place during such interactions. Drawing inspiration from agent-oriented software engineering practices, we aim to formalize the communication between users and AI models as an exchange of messages and develop a semi-formal representation for HAI interactions, considering a set of requirements for language properties.

2.1 HAI Interactions: Definitions, Frameworks, and Taxomonies

2.1.1 HAI Interaction Paradigms. *HAI interaction* is defined as *"the completion of a user's task with the help of AI support, which may manifest itself in non-intermittent scenarios"* [98]. Following this definition, there are three main HAI interaction paradigms, *intermittent*, *continuous*, and *proactive*, which take into consideration that *"differences in initiation and control result in diverging user needs."* Intermittent HAI interactions are user-initiated and turn-taking interactions where the system provides a response for an explicit and predefined cue. Continuous HAI interactions utilize user's implicit input, as part of a continuous input stream, and provide a response which can either be accepted or ignored by the user. Finally, proactive HAI interactions are AI-initiated and triggered by predefined changes in the system. Since such interaction paradigms can exist in parallel, there is an increasing need to consider the open challenges while designing interactions in terms of how users and AI systems can interact with each other and to develop new categories for the types of interactions between people and AI-enhanced technologies [43].

One of the key concerns while designing HAI interactions is the black box nature of AI models, where decisions and underlying operations are not visible or explained to humans. Moreover, human intentions are not always clear or they differ from the actual communicated user action. This problem is known as the *"two black boxes problem,"* based on which both human cognition (cognitive intelligence) and AI are considered to be black boxes. In order to address this problem, a symmetric and collaborative model for HAI interactions has been proposed, highlighting the need for explainability for both communication channels [101]. Based on this framework, **Semantic Interaction (SI)** is used as a design philosophy for symmetric and collaborative interactions between humans and models, where XAI serves as a method to communicate information from AI to human users, and *explainable cognitive intelligence* is used to provide information from humans to AI systems. A main challenge while considering both communication channels for the design of HAI is the *semantic gap*. In order to address this, human high-level decisions and actions need to be translated to a set of model-understandable parameters and operations, while AI models must communicate its internal processes and decisions in a human-understandable way [33].

HAI interactions should enable both systems and users to efficiently communicate to each other and make decisions in a collaborative manner to augment both human and AI, toward HI interactions. Akata et al. [4] unpack the research challenge of building HI systems into four research themes: *Collaborative*, *Adaptive*, *Responsible*, and *Explainable HI*. An open challenge toward designing interactions with HI systems is to develop methods for designing negotiation, agreements, planning, and delegation interactions in hybrid teams, considering the needs and role of the team members. *Usable* and *Useful AI* are two terms which describe how interactions with AI models can be designed to provide usable solutions that are easy to understand and apply in order to satisfy user needs [104]. Based on the Teaching-Learning interaction paradigm [50], AI systems could also enable human users to guide the model's learning process, by enhancing their perception of what the system can learn, through interactivity and transparency. Given the wide range of such interaction paradigms, there is a need for the development of methods which can enable human-centered design features, such as explainability and user control.

2.1.2 HCAI Methods and Frameworks. HCAI refers to a set of practices for building, evaluating, deploying, and critiquing AI systems that balances technical performance with an focus on human and social concerns [19]. While HCAI is a broad term with multiple definitions [80], the goal of these methods is to augment and enhance human performance, while establishing a reliable, safe, and trustworthy interaction with an AI system [85]. One of the main challenges while developing HCAI systems is following human-centered design principles while considering the technical and implementation challenges of the methods used [72]. The integration of *explainability* and *user control* features creates new types of interactions between users and models and also requires appropriate implementation mechanisms. Designing interactions with explainable models should follow specific design principles, related to fairness, transparency, and privacy aspects [9]. In terms of *user control and feedback*, HITL and **interactive ML (iML)** techniques can enable users to interact with AI/ML models in order to guide, steer, and facilitate the learning process. Human users can be involved in the different stages of the ML pipeline: data extraction, integration and cleaning, iterative labeling, as well as model training and inference [18, 103]. Such approaches highlight the need for designing new types of interactions, leading to the development of design frameworks and guidelines for HAI interactions.

Taxonomies and frameworks for XAI can support the design and implementation of interactions with *explainable and transparent* systems. A collection of XAI-based questions has been proposed as a design space for XAI interactions, considering diverging user characteristics, e.g., user needs, role, expertise, and experience [54, 55]. Other taxonomies focus on the different aspects of the use case, e.g., problem definition, explanator properties, and evaluation metrics [83] or the relations between the use case, the AI system, and the explanation algorithm [1]. Design frameworks can provide guidelines on how to design and evaluate XAI-based interactions by mapping design goals to evaluation methods based on the target population [64]. Focusing on user evaluation for XAI interactions, Chromik et al. [23] classify evaluation methods based on task-related, participant-related, and study design-related dimensions and can support the selection of appropriate XAI methods for a given interaction concept. Sperrle et al. [87] proposed a dependency model for XAI processes as a design space for human-XAI interactions, considering the potential bias that can be propagated during the interactions between different stakeholders and end-users. Wang et al. [99] followed a theory-driven approach to link explanation features to user reasoning goals, resulting in a conceptual framework which can inform the selection of appropriate explanations toward mitigating possible cognitive biases. Designing XAI interfaces should also consider specific design principles based on the interaction concept and explanatory goals [22]. Human users may interact with XAI interfaces to understand the underlying AI behavior *(information transmission)* or establish a user-driven communication through user queries and AI responses/explanations *(dialogue)*. For both concepts, the AI behavior does not change in contrast to XAI-interaction as *control*, where the user is included in the loop and adjusts the model's behavior until a desired behavior is reached.

HITL methods and approaches can enable human users to participate in the decision-making and model training process. Design principles for user control and feedback interfaces include interactivity to promote rich interactions, providing explicit and clear task goals, supporting user understanding and engagement, and capturing user's intent based on their input [31, 69]. Moreover, designing interactions with iML interfaces requires appropriate learning methods, considering the different forms of relationships between humans and ML algorithms [24, 65]. The design and implementation of HITL interfaces depend on the type of user feedback and can affect both UX and system/model performance. Michael et al. [62] discuss how the different types of cognitive feedback can be integrated to the learning mechanism, i.e., *self-reporting*, *implicit feedback*, and *modeled feedback*, through different feedback mechanisms. For example, domain-expert feedback can be communicated to the system and translated to model updates, i.e., modify dataset or model parameters [20]. Such approaches consider the effects of the interaction on user's engagement and quality of their feedback. Since explanations can be used to enhance user's understanding of the model's performance, they can play an important role in ensuring a high quality of user feedback. Explanations can be combined with interactive capabilities enabling users to train ML models from scratch, resulting in a closed loop of XAI- and HITL-basedbreak interactions [93].

HI methods can combine human capabilities with AI, allowing both parts to efficiently communicate and exchange information in order to mutually make decisions, inform and learn from each other [4, 111]. Such interactions can also enable human users to challenge a model's decisions and negotiate with it until reaching a final decision [45]. A proposed framework for contestable AI by design [7] highlights a set of socio-technical features and practices for model contestation and hybrid decision-making, including interactive controls, explanations, and intervention requests. The development of HCAI systems usually combines different implementation methods, making them not trivial to document and evaluate [86]. Our aim is to explore how models and users communicate through the different methods; XAI-based interactions require the communication of explainable model predictions and/or parameters, while HITL interactions require human feedback in a specific format to integrate it to the learning or decision making processes. HI systems can utilize both user's and AI's communicated information to augment both agents' perception of their common task and environment. These types of information, i.e., model inputs, outputs, explanation, and user feedback, can be combined into more complex patterns of interaction, including intervention requests, negotiation-based interactions, shared decision making, **Active Learning (AL)**, and so on. Next, we provide an overview of design methods and frameworks which aim to support the design of HAI interaction systems.

2.2 Design Methods for HAI Interactions

Design guidelines and frameworks aim to support designers on how to apply specific design principles for interactions with AI systems. Microsoft Research has proposed a set of guidelines for designing HAI interactions [8]. These guidelines provide design choices and examples for interactions between users and AI systems, considering aspects of *system transparency and explainability*, and *user feedback and control*. Designing interactions which support such features requires human users to exchange low-level information with the system. That means that designers should take into consideration the computational and interactional capabilities of AI model, as well as the underlying design and technical challenges, while prototyping such interactions. A process model for co-creation of AI experiences describes how designers can be familiarized with AI models and their functionalities in order to include them in the design process as design materials [90]. However, there is a lack of design innovation in envisioning how ML/AI might improve and create UX value to users. Yang et al. [107] highlight the need to support UX practitioners by creating design patterns to describe specific aspects of the interactions between users and models. They propose four channels based on which AI/ML capabilities can create UX value to users, namely *self*, *context*, *optimal*, and *utility-capability*, considering what users can infer about these aspects through their interaction with models.

Design patterns can be used as a repeatable solution to a software engineering problem, enabling system designers to re-use existing solution templates for a given objective or design goal [81]. In the context of interaction design, design patterns can simplify the design of complex systems and interactions and make them transparent in terms of their design and implementation requirements [73]. Focusing on co-creation applications with Generative Adversarial Networks, Grabe et al. provide a list of HAI interaction patterns which categorize interactions based on the AI's co-creativity support level [38]. These interaction patterns are defined as sequences of actions which describe a specific activity, e.g., initialize, create, adapt model, and can be combined to design interactions with different co-creativity support levels. A co-creative framework for interaction design analyzes the interactions between users, AI models, and the shared product [77], resulting in a set of design choices, which can define the behavior of the AI, as a generative, improvisational, or advisor agent. In the context of AI-based game design, AI can serve different roles based on the selection of the interaction pattern [94]. For example, AI can act as *role-model*, where users need to imitate an AI agent to complete a game, or as an (AI trainee) which enables the user to teach the agent to do something in the game.

Design patterns have also been used for collaborative learning interactions to describe interactions which enable team members to learn from each other [82]. HI and collaborative learning systems can combine *HITL* with *computer-in-the-loop* interactions in order to enable both users and models to effectively communicate [102]. More specifically, users and computers can communicate through a set of interaction patterns: (a) *decision support*, (b) *exploration*, and (c) *integration*. Each pattern refers to a different type of interactions between human users and computers as an exchange of inputs, outputs, and feedback/explanations. However, even prototyping such interactions would require a lower-level specification of how users and models communicate and exchange this low-level information. To this end, design patterns have been used to specify and categorize the interactions with data-driven and reasoning systems based on the model processes and data operations [97]. These patterns can specify the interactions in terms of model operations, including training, inference, and transformation, and can be extended to capture the concept of different types of human actors included in interactions with hybrid AI systems. A formal representation of model and data processes can provide developers with information about the implementation aspects of an interaction. Toward this, we review applications of HAI systems focusing on the implementation aspects of HCAI methods.

2.3 Implementation of HCAI Methods

Advances in computational and learning methods, as well as the plethora of human-generated data, have recently led to innovative AI systems which can be used by non-expert users for complex tasks. More specifically, OpenAI has released two types of deep learning models which generate new content based on user's input (prompts); DALL·E is an AI system which generates images based on a textual description [76] and chatGPT is a Large Language Model which generates structured textual content based on user's prompts and questions.[1] Despite their complex architecture and advanced learning methods, interactions with such models are straightforward; the user provides a prompt and the model returns a generated response, offering only one interaction paradigm, i.e., the user guides the model output through their different input prompts. However, there are more interactions that can take place around this model, expanding the design space of HAI interactions [21, 27]. For example, the training process of chatGPT requires AI methods which can utilize human feedback to facilitate model learning. The training process utilizes an HITL method to involve human users to the learning process. More specifically, **Reinforcement Learning (RL)** from Human Feedback is used to integrate human expertise to the learning process [41], based on which, human users can provide feedback during different learning process steps. During model initialization, human users demonstrate the desired optimal behavior of the model (response to

[1]https://openai.com/blog/chatgpt/

a prompt). For model optimization, a reward model is trained based on user ranking of possible responses (outputs) for a given prompt (input). Such an approach demonstrates the different implementation aspects, i.e., selection of learning/update mechanisms, that need to be considered for the implementation of HAI interactions between multiple users and models with different roles. Interactions can become complex when designing for explainability and user feedback, even with less advanced or pre-trained models. Toward this, we provide an overview of HAI interaction use cases which demonstrate how HCAI methods can be implemented to achieve (a) *transparency and explainability* and (b) *user feedback and control*. While we do not provide a comprehensive review of methods and use cases, we explore what types of information need to be communicated between users and models for the development of HCAI methods.

2.3.1 Transparency and Explainability. Explainable and transparent interfaces can communicate information about the model's performance and functionality and can serve different purposes. Model transparency has shown to improve user's trust during their interaction with a virtual conversational agent [100]. More specifically, human participants interacted with a virtual agent and an underlying speech recognition system. The virtual agent was used as a visualization means for model transparency to enhance user's perception of the model outputs and trust to the system. During these interactions, the user provides a model input (utterance) and the model communicates both its output and additional feedback (prediction/explanation). Model explainability can also enhance the collaboration between users and models for a given task. In the context of AL, model explanations aim to support human annotators during a data labeling task. An interactive image classification system communicates model confidence by visualizing model predictions with low confidence [44]. Explanations have also been used to enable both users and models to justify their decisions in a collaborative game context [42]. The model communicates its prediction for the game outcome along with rule-based explanations and the user can modify both the prediction and the explanations. Explainable interfaces can also enhance user's understanding of how the model makes predictions, thus enabling them to personalize their interaction with the model. An explainable music recommendation system visualizes the user preferences based on which it makes recommendations and adjusts its behavior based on user actions, through accepting or rejecting recommendations, or modifying their own preferences [60]. These examples highlight the need to follow specific practices for explainability and transparency. Moreover, the design space of XAI-based interactions can be broadened, considering the possibility or explanations at different stages of the system, i.e., development or deployment, and the potential stakeholders involved [26].

2.3.2 User Feedback and Control. HAI interactions can involve multiple users with different roles, expertise, and intentions. In such interactions, an AI model should be able to behave in a different way based on the user's characteristics. This requires the implementation of a learning mechanism which considers different types of human feedback. An interactive RL-based framework for personalized robot-based cognitive training involves two different types of users [96]; a player (primary user) interacts with a robot (RL agent) through a game-based interaction where the robot sets the game difficulty, and a supervisor (secondary user) can remotely supervise and control the interactions. In terms of communicated information, the player provides implicit feedback to the model through task performance and engagement, while a supervisor can monitor the model's decisions through an informative **User Interface (UI)** and modify the robot's decisions, when needed. Both communication channels contribute to the model's learning updates and the decision-making process in a different way. Similarly, the Human Experience Transfer Model [59] combines two interaction loops; one for a human expert (trainer) who guides the model learning process and one for the human learner (trainee) who can provide feedback during the learning interaction. The trainer loop aims to facilitate the model training while the learner loop aims to

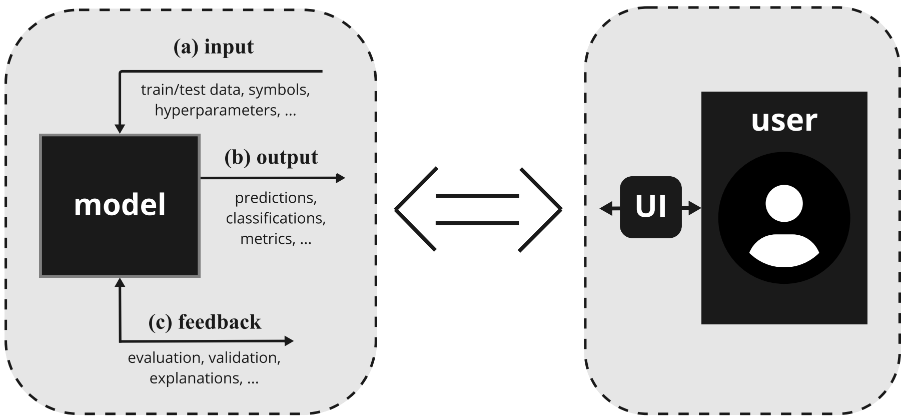

(a) input
train/test data, symbols, hyperparameters, ...
model
(b) output
predictions, classifications, metrics, ...
(c) feedback
evaluation, validation, explanations, ...
UI
user

Fig. 2. Communication types during human–model interactions. AI models provide/receive information in a model-specific format: (a) input (test/train data, relations, hyperparameters, etc.), (b) output (labels, predictions, estimated parameters, projections, etc.), and (c) feedback (explanations, validation, requests, etc.). Users send and receive information through the UI which translates user actions to a model-understandable format and vice versa. Our approach aims to specify such communications using interaction primitives.

personalize the learner's interaction. These loops specify two different interactions with different communication channels and model update methods. HITL-based interactions can allow for user feedback and control within an interaction in various ways [24]. The type of human feedback (as well as amount, frequency, granularity, etc.) should be inline both with the user's characteristics (e.g., role, expertise, preferences) and the model specifications (model architecture, learning/update rules, etc.). Human feedback can facilitate the model training process by enabling the user to adjust the model's behavior through modifying its predictions or by evaluating the model's performance or predictions [49, 56].

Such implementation approaches of HAI interactions highlight the need to specify the information exchanges between users and models. Our work moves toward a semi-formal representation which can encode these low-level communications between users and models and combine them into a set of design patterns for HAI interactions. Following the taxonomy for hybrid AI systems [97], model-specific information can be in different formats, including train/test data, learning and estimated parameters, symbols, rules, labels, and others. Moreover, user feedback and explanations can be communicated in a model-specific format. Based on the SI paradigm [101], human and artificial agents should be able to exchange information in a mutually understandable way and ensure meaningful communication from both sides. In order to address the *semantic gap*, both user and model actions should be translated into a mutually-understandable format [33]. Based on the model specifications, user actions can be translated through the UI into machine-readable information. Similarly, model actions should be effectively communicated and understandable to human users. To this end, our aim is to represent HAI interactions as an exchange of information with respect to the model's interactional capabilities and characteristics (Figure 2). In order to define an interaction in terms of exchanges of model inputs, outputs and additional feedback, we need to provide a formalization of such HAI communication exchanges. Toward this, we explore formal languages and representations for (a) multi-agent communication systems and ACLs and (b) processes and pipelines for AI/ML systems, in order to identify ways to model HAI interactions in terms of information exchanges.

2.4 Formal Models and Representations for HAI Interactions

2.4.1 ACLs. ACLs aim to provide a means of exchanging information between agents following a specific set of rules and protocols for a specific service. ACLs are high-level communication methods and use lower-level communication protocols for transferring information. Research in ACLs is inspired by the Speech Act theory [10], based on which speech acts or *performative* verbs can represent the state and intention of communicating agents. There are many types of speech acts, including acts to provide (representatives) or to request (directives) information. Examples of speech acts include: "*request,*" "*inform,*" "*suggest,*" and "*query.*" The **Knowledge Query and Manipulation Language (KQML)** and the **Foundation for Intelligent Physical Agents ACL (FIPA-ACL)** are two major developments in message exchange interaction protocols between agents [34, 71]. KQML is a high-level, message-oriented communication language and consists of the *message*, *content*, and *communication* layers. KQML uses performatives to describe the intention of an agent to send a message with a specific content, including basic queries ("ask-if," "ask-about") and informatives ("tell," "deny"). Similarly, FIPA-ACL uses performatives which denote the type of the communicative act, including an *assertive inform* act, based on which an agent provides a message, and a *directive request* act, which describes a message request from an agent to another. Such primitive communicative acts can be combined into composite acts to describe more complex communications. Our work aims to define HAI communicative acts based on which we describe the intent and the type of information exchanges information through a small set of performatives. Following the concepts and notations from existing ACLs, our goal is to define the *message*, *content* and *communication* layers, in the context of HAI interactions.

ACLs can also be used to formalize the communication between multiple and distributed agents. Process calculus provides a tool for a high-level description of interactions, communications, and synchronizations between agents or processes. The Complete Agent Modeling Language is a comprehensive modeling framework for **multi-agent systems (MAS)** which can formalize the structural organization and reasoning abilities of the interacting agents [2]. Taking into consideration multi-agent coordination and social norms, **Lightweight Social Calculus (LSC)** can be used to specify the behaviors required of agents interacting in a given social context [78], as a form of "electronic institution" [28]. LSC provides a framework for describing patterns of communication and has been extended to model coordination and communication between multiple interacting actors in social and domain-specific contexts [68]. **Domain-Specific Modeling Languages (DSMLs)** are modeling languages of limited expressiveness which aim to formally represent structure, behavior and requirements of a particular domain [47]. The development of DSMLs requires the definition of the domain processes and communication exchanges between the available agents in a given context [3, 75]. DSMLs can provide a high-level abstraction of rules and concepts, which enable non-technical domain experts to employ high-level concepts and rules for a given context, and develop complicated designs without handling technical complexities [11, 61, 92]. DSMLs have been developed to support practitioners while dealing with high-level aspects of ML systems, including social concerns and composition of ML datasets [37], and ethical practices for transparency, fairness, and accountability in ML pipelines [113].

Despite the wide range of ACLs, their development and evaluation follow specific criteria and requirements for agent-oriented software engineering systems [84, 88]. The quality of a language can be determined based on a set of requirements which correspond to the desired characteristics of the language. The AgentDSM-Eval framework [6] uses six major quality characteristics for the development and evaluation of MAS: *functional suitability*, *usability*, *reliability*, *expressiveness*, *compatibility, and MAS development.* Based on a similar set of quality characteristics, the **Agent Modeling Language (AML)** is developed as a *theoretically sound*, *comprehensive*, *consistent*, *easy to use*, *extensible*, *generic*, and *automatable* modeling language [95]. Our approach is inspired by such agent-based modeling and evaluation approaches; our goal is to model HAI interactions considering both the low-level model functionality and high-level aspects of HAI interactions, aiming to bridge the gap between engineering and design practices. In order to do so, we propose to develop a *semi-formal modeling language for HAI interactions*, based on which a small set of modeling elements (interaction primitives) can be used to describe complex HAI interaction systems. Following the quality criteria and language requirements for AML [95], our goal is to develop a language which is:

— *Generic:* it is independent of any particular theory, paradigm, technology, or development process of AI systems;
— *Theoretically sound:* it is well reasoned based on the theory for agent-oriented software engineering systems;
— *Well specified and documented:* it includes a detailed and comprehensive specification of its notations, syntax and semantics, and use;
— *Comprehensive:* it includes the required modeling structures and elements to represent a wide range of current and emerging HAI interaction paradigms;
— *Consistent:* its concepts, notations, and syntax and semantics are consistently specified and do not contradict with each other;
— *Extensible:* it can be used to introduce new modeling elements and specify them in terms of design and implementation aspects of HAI interactions;
— *Easy to use:* it takes minimal effort for a wide range of people to learn and apply the language and its embedded concepts to describe existing and new types of HAI interactions;
— *Automatable:* it includes computational and executable representations to simulate, test, and evaluate interactions between users and models.

2.4.2 Representations of AI Processes and ML Pipelines. HCAI approaches have led to the development of interaction techniques to facilitate end-user interaction with ML models. Marcelle is an architectural model for the design of human-ML interactions [35]. Its architecture includes a modular collection of interactive components which enables users to interactively train their own models, by providing training data, correcting misclassifications, and testing model parameters. Lara is a declarative domain-specific language which can formalize the preprocessing and training processes in terms of data and model processes [52]. Other formalization approaches focus on the fairness and accountability aspects of ML pipelines, including methods for model auditing and technical bias mitigation in order to incorporate ethics and legal compliance into machine-assisted decision-making [40, 79, 91, 106].

Such approaches can provide a graph representation of the dataflow from a preprocessing pipeline which can be used to formally describe the data operations [39]. The instance taxonomy for hybrid AI systems [97] can specify a dataflow representation of a system in terms of model operations, including training, inference, and transformation, and can be extended to capture the concept of different types of human and model actors. In order to support AI practitioners to develop ML models for human-centered applications, model sketching provides a technical framework for authoring ML models based on high-level human-understandable concepts [53]. To do so, a low-level representation of ML pipelines is used to translate high-level concepts into functional building blocks. Such an approach can help practitioners to consider how low-level implementation aspects can be used to meet high-level requirements for human-centered development of ML models. Our proposed approach aims to develop a semi-formal representation of low-level information exchanges which can describe HAI interactions in terms of implementation and design aspects. In the next section, we present how unpacking an HAI interaction into these low-level information exchanges, can result to a set of interaction primitives and patterns which can describe an HAI interaction.

Based on the literature review, our goal is to develop a formalization which can adequately represent HAI interactions and characterize them in terms of their design considerations (Section 2.2) and implementation aspects (Section 2.3), which can support the development of HCAI systems. Toward this, we propose a semi-formal modeling language [78, 95] which can be used to define and characterize HAI interaction patterns. Figure 1 visualizes how our approach is informed by:

— Human-centered design of AI systems: Our approach is informed by existing design frameworks and guidelines for HCAI systems, including responsible AI, contestable AI, and model auditing.
— Implementation of HCAI methods: The development of our language considers a range of HAI interaction paradigms and methods, including XAI, iML, HITL, AL, HI, and collaborative learning.

3 Unpacking HAI Interactions into Interaction Primitives and Patterns

3.1 Unpacking HAI Interactions to Patterns and Primitives

Our approach aims to unpack HAI interactions into *interaction primitives* which can specify the type and intent of the communication during a single interaction step. Such interaction steps can be defined as user-model information exchanges, e.g., user provides an input, user requests an output, model provides an output, and so forth. Our motivation is that unpacking HAI interactions can provide us with insights about the underlying design and implementation aspects of HAI interactions. In order to unpack HAI interactions into interaction primitives, we identify the interaction patterns between users and models and we specify the intent and type of the communicated information. The goal of this approach is to identify a set of communicative acts from different HAI interaction scenarios, toward a formalization for HAI interaction primitives. We demonstrate our approach using an interactive robot learning system for multimodal emotion recognition as a running example [109]. More use cases were selected for unpacking in order to cover different types of interactions, including XAI, HITL, and HI interactions (see Appendix). We provide a description of the HAI interactions in the form of patterns and actions, and we highlight the design and implementation aspects of the interactions (Figure 3).

3.1.1 Description of the HAI Interactions. The goal of this system is to collect and annotate human-generated data for an emotion classification model, through the human–robot interaction. In terms of interaction design, the interaction starts with an emotion elicitation session (user watches a clip which invokes a specific emotion). After the session, the user is asked to select their current emotional state from a list of emotions. This emotional state is provided as a model output (class) and it is used by the robot for the interaction as the ground truth emotion. The robot asks the user to walk toward and stand in front of it, demonstrating the selected emotion. The robot uses the gait/thermal data to predict the user's emotional state. If user's response (ground truth) is different from the predicted emotion, the robot asks the user about their current emotional state, annotating the collected gait/thermal data which are used to retrain the model.

3.1.2 Interaction Patterns. We unpack the interaction into three patterns of interactions between the user and the robot (model): (a) *class selection:* model asks user to select an emotion from a list (class)—user selects an emotion from the list, (b) *new class sample:* robot asks user to provide an input by demonstrating the selected emotion—user responds by walking and standing in front of the robot, and (c) *annotate sample:* model makes a prediction and asks the user for labeling, if prediction is different from ground truth—user provides the correct label by responding to the

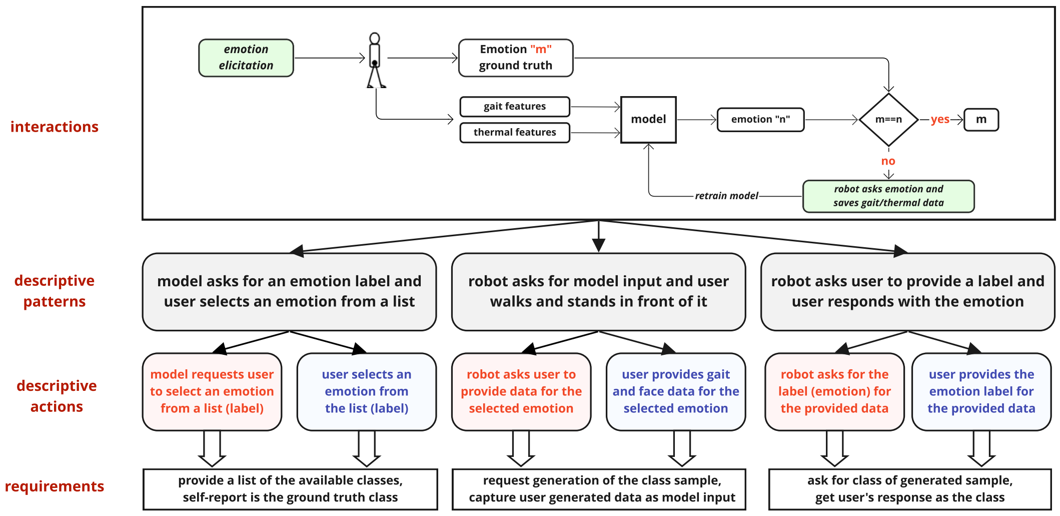

interactions
emotion elicitation
Emotion "m" ground truth
gait features
thermal features
model
emotion "n"
m==n
yes
m
no
robot asks emotion and saves gait/thermal data
retrain model
descriptive patterns
model asks for an emotion label and user selects an emotion from a list
robot asks for model input and user walks and stands in front of it
robot asks user to provide a label and user responds with the emotion
descriptive actions
model requests user to select an emotion from a list (label)
user selects an emotion from the list (label)
robot asks user to provide data for the selected emotion
user provides gait and face data for the selected emotion
robot asks for the label (emotion) for the provided data
user provides the emotion label for the provided data
requirements
provide a list of the available classes, self-report is the ground truth class
request generation of the class sample, capture user generated data as model input
ask for class of generated sample, get user's response as the class

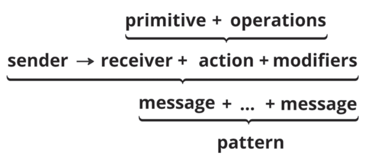
Fig. 3. Interactive robot learning [109] using descriptions of interaction patterns and primitives. The image shows the original interaction schema as a box and arrow diagram, followed by a high-level description of the HAI interaction patterns, the unpacking into actions involving the exchange of information between the human and the system, and a description of the interaction requirements.

robot's question. During these interactions, user and robot exchange information in the form of *model input* (gait/thermal data) and *model output* (emotion from list, model prediction, robot question, user response). Model input is communicated as a set of raw data generated and captured during user's activity (walking and standing). Model output is communicated (a) through user's selection from a list, (b) implicitly through the robot's (model) prediction, and (c) as a user response to the robot's request during their interaction.

3.1.3 Design and Implementation Aspects. The robot runs an underlying emotion classification model which uses walking (gait data) and facial expressions (thermal data) as model inputs to predict the user's emotional state. The system utilizes the interaction with the user in order to dynamically improve the classification model by retraining on new (labeled) data. The design of the interactions enables the user to participate in the interactive learning process in an implicit way. The prediction model consists of two models (gait and thermal models) and the final prediction is estimated through a modified confusion matrix. After each interaction with the user, the model is retrained including the new data from the user as an input–output pair. The robot initiates the interactions to request for user's input (emotion label, input data, and user response) and uses the responses to make interaction decisions; if model's prediction is different from user's input, the model implicitly requests the correct label from the user without communicating its own prediction. These implementation and design choices aim to support model's improvement, i.e., when the model fails to predict the user's emotion, while engaging the user in the interaction by communicating the "ground-truth" emotion. A possible limitation of this implementation is that it highly depends on the quality of user's input. User may provide inaccurate information both during the selection (label) and the demonstration (sample) of the emotion.

Based on the unpacking of a set of use cases for HAI interactions, we can describe interactions using these to provide and request messages for a specific model data type, i.e., model input and output data. Our approach aims to address the semantic gap by defining such information exchanges in a mutually understandable and human-readable format.

Table 1. Definitions and Visual Representation for Interaction Primitives, Types, and Actions

Definitions		
primitive	`P: provide(X:type+,Y:type*)`	 primitive + operations sender → receiver + action + modifiers message + ... + message pattern
	`R: request(X:type+,Y:type*)`	
type	`input`: raw_data | fvector | state | query | mparams | ...	
	`output`: class | regression | action | clusters | items | ...	
	`feedback`: validate | evaluate | XAI.features | XAI.rules | ...	
action	`P | R → op*`	
	`op`: create(X) | select(X,Y) | modify(X,Y) | map(X,Y)	
message	`<sender→receiver,action,[mod*]>`	
	`mod`:<key: *var*, value: *any*>	
pattern	message+	

An action is defined as a primitive specified by a set of operations. Actions are communicated through messages, by specifying the interacting agents and the modifiers of the message. Actions are reusable and can be used by different messages. Sequences of messages can form patterns of interaction.

3.2 Defining Interaction Primitives and Patterns for HAI Interactions

We propose a semi-formal modeling language for HAI interaction which uses a message-passing protocol to describe the information exchanges between users and models. We define such information exchanges using a set of performatives from ACLs which describe the intent of communicating a message to provide or request a specific type of information. Considering the different types of communication between users and models (Figure 2), we define a set of interaction primitives which specify user/model actions in terms of the *intent*—provide or request information—and the *type* of the communicated information—content. More specifically, we define the *provide* and *request* performatives which describe a user/model providing or asking for a message with a specific content. The message's content depends on the type of the information, e.g., model-specific data types. Our motivation is that a simple set of performatives and types can sufficiently describe the possible information exchanges in a compact manner. We provide a description of the definitions for our proposed formalization (Table 1), as well as examples to demonstrate our approach.

3.2.1 Interaction Primitives and Types. We define an *interaction primitive* as a low-level communicative act which specifies the type and intent of the communicated information. Considering the intent during an interaction step, the communicating agents can either *provide* or *request* information. We define a *provide primitive* as `P: provide (X:type+,Y:type*)` to describe the act of providing information (provide type `X` referencing type `Y`). Similarly, a *request primitive* is defined as `R: request (X:type+,Y:type*)` to describe the act of requesting information (request type `X` referencing type `Y`). For both primitives, the optional argument (`Y:type`) is used to reference an additional type. Considering the format of the information, human users and models can provide and request the following types of information: `input`, `output`, and `feedback`. The definition of the types (and subtypes) can inform the design and implementation of the specific interaction, since they characterize how users and models exchange information. We provide an overview of the types (and their possible subtypes) with examples.

— `input` is information to be fed into a model. The *subtype* depends on the model type and architecture, including numeric vectors, world state information, images, video frames or sequences, user preferences, audio, text, and so on. Model-specific parameters (hyperparameters) are also used as model input, e.g., learning rate, number of clusters, model sensitivity, and so forth.

Table 2. Examples and Descriptions of Interaction Primitives and Types

Examples of interaction primitives and types		
`provide(X:input)`	provide an input	upload/capture an image
`provide(X:output,Y:input)`	provide an output for a given input	detect a face in the image
`provide([X:input,Y:output])`	provide an input–output pair	show an image and the detected face
`request(X:output,Y:input)`	request an output for an input	ask if there is a face in a given image
`request([Y:input,X:output])`	request an input–output pair	ask for an image with a detected face (if any)
`provide(Z:output,[Y:input,X:output])`	provide output for an input–output pair	modify an existing bounding box on an image
`request(F:feedback,Y:output)`	request feedback for the given output	ask for confirmation about a bounding box
`request(X:[input])`	request a set of inputs	ask for a set of input images

— `output` is information coming out of a model—class labels, feature estimations, lists of recommendations, selected actions, generated images, projections of the input space, cluster labels, lower dimensional data, and so on. This can also include learned parameters such as model weights, confidence values, loss, and so forth.
— `feedback` refers to the additional information that can be provided both from users and models, including (requests for) evaluation, validation, explanations, and so forth. Explanations can be provided in different modalities, e.g., salient maps or natural language. Users can provide explanations to justify their decisions, as well as feedback for a model's decision. Similarly with model input/output, feedback is model-specific.

The proposed interaction primitives and types can describe different *actions* between users and AI models. Primitives capture the intent of the communication (provide/request) and types describe the format of the communicated information. For example, actions `u1: provide (X:input)` and `u2: request (Y:output, X:input)` can both describe a user communicating a model input type. However, the first action describes a user providing `X:input`, while the second one describes a user request for `Y:output` given the provided input. These actions are similar in terms of the provided input (`X:input`) but they differ on the type of the action (provide input vs. request output).

The selection of the types (and subtypes) depends on the model specifications and can inform design and implementation choices. For example, given that `provide (X:input.raw_data)` can describe a user who communicates a model input in the form of raw data, the interaction design (e.g., interface) should enable the user to interact with raw data (e.g., images). Primitives and types can also provide information about the requirements related to the goal of the interaction. Let us consider a face recognition model which takes as an input an image (raw data or extracted features) and detects faces (if any) in the form of bounding boxes. For this case, both of the defined actions `request (Y:output;X:input)` and `request (Y:output, [X:input, Z:output])` can describe a request for an output (detected face image). The first one is an output request for a given input image and can describe a face detection interaction. The second action is an output request given an input–output pair (image—bounding box) and can describe a modification request for the model's detection. Table 2 provides examples of interaction primitives and types along with a description in the context of face detection.

3.2.2 Actions, Operations/Performatives, and Modifiers. In order to specify an interaction between a user and a model for a given interaction context, we define a message as `msg: < sender→receiver, action, [mod*] >` to describe the communication of an action from a sender to a receiver. An action is defined as `P | R ← operations` and specifies a primitive action by adding a description of how the arguments need to be communicated, through a set of `operations`. The operations contextualize an action by specifying the preconditions for its communication in a given interaction context. More specifically, the operations define how the argument is being created and describe the relations between multiple arguments. A list of operations includes, but is not limited to

— `select(A,B)`: argument `A` is selected (from a given set `B`—optional argument)—this operation can describe the selection of an item from a list or set of choices, i.e., recommendations, labels, samples, and so forth.
— `map(A,B)`: argument `A` is mapped to argument `B`—this operation can be used to describe a model prediction (e.g., classification, regression, clustering), human labeling, evaluation, and so forth.
— `modify(A,B)`: modify argument `A` to argument `B`—this operation can describe a modification of a sample (modify input image), an alternate decision (change label), and so forth.
— `create(A)`: create new argument `A`—this operation can describe data acquisition or generation (e.g., image, sound), a human annotation (e.g., new label), and so forth.

Finally, a set of `modifiers` (mod: < key: *type*, val=*any* >) can be used to further characterize the message in terms of interaction requirements, as a free-form annotation feature. Modifiers can provide information about the interaction modality, interface elements (e.g., buttons, forms), type of communication (e.g., explicit vs. implicit), and so forth. Based on the above, the definition of an action includes the primitive (provide/request), the type (input, output, and feedback), the operations needed to communicate the types, as well as additional information for the interaction through the modifiers, and describes the communication of an action as a single message from a sender to a receiver. A sequence of actions for a given interaction context is defined as an *interaction pattern*.

The proposed formalization allows for a custom definition of an action and its specification as a message for a given interaction context. For example, we can define the action **`req-new_sample(M)`**≡ `request (M:input.raw_data)⟵create(M)` to describe a request for a generated input in the form of raw data. Sequences or exchanges of messages can be used to define interactions between users and models for a given context. For interactions with a face recognition model, we can define a message `msg: < model⟶user,` **`req-new_sample(M)`**`,[M:image] >` to describe the model's request for an image from the user. As a response to this request message, the user could respond with either

— `msg1: < user⟶model,` **`generate-sample(M)`**`,[M:image] >`,
where: `generate-sample(M)≡provide(M:input.raw_data)⟵create(M)` or
— `msg2: < user⟶model,` **`req-gsample_class(M,L)`**`,[M:image] >`,
where: `req-gsample_class(M,L)≡request(L:output.label,M:input.raw_data)`
`⟵[create(M),map(M,L)]`

Based on the first message, the user responds by providing the requested input (create/capture image), while with the second message, a request is made to the model for face detection given the generated input (map/assign the captured image to a label). The same action can be communicated by different messages. A message specifies the communication of an action from a sender to a receiver in a given interaction context. For example, a speech recognition model can communicate its request for user-generated input (speech) using `msg': < model⟶user,` **`req-new_sample(M)`**, `[M:speech] >`. From this, a set of actions can be defined as a vocabulary which can be used in different HAI interactions. Such actions and messages can be used to design interaction patterns as sequences of messages between users and models. Given the message definitions above, we can define `query_input≡[msg,msg1]` as a query pattern to describe the request and communication of a model input, while `query-label_input≡[msg,msg2]` could describe an interaction where the user awaits for the model's decision based on a new sample. These patterns serve different

Table 3. Messages and Action Definitions for the Interactive Robot Learning Interactions

Message		Action definition
A1	`<model⟶user,`**`req-class_selection(Y,L)`**`, [Y:reqSelfReport;L:listEmotions]>`	`req-class_selection(Y,L) ≡ request(Y:output.label, L:[output.label]) ⟵ select(Y,L)`
A2	`<user⟶model,`**`select-class(Y,L)`**`, [Y:SelfReport;L:listEmotions]>`	`select-class(Y,L) ≡ provide(Y:output.label,L:[output.label]) ⟵ select(Y,L)`
A3	`<model⟶user,`**`req-new_class_sample(X,Y)`**`, [X:reqWalkStand,Y:SelfReport]>`	`req-new_class_sample(X,Y) ≡ request(X:input.raw_data,Y:output.label) ⟵create(X), map(X,Y)`
A4	`<user⟶model,`**`generate-class_sample(X,Y)`**`, [X:WalkStand,Y:SelfReport]>`	`generate-class_sample(X,Y) ≡ provide(X:input.raw_data,Y:output.label) ⟵create(X), map(X,Y)`
A5	`<model⟶user,`**`req-sample_class(X,Y)`**`, [X:WalkStand;Y:reqSelfReport]>`	`req-sample_class(X,Y) ≡ request(Y:output.label,X:input.raw_data) ⟵ map(X,Y)`
A6	`<model⟶user,`**`annotate-sample(X,Y)`**`, [X:WalkStand;Y:SelfReport]>`	`annotate-sample(X,Y)≡ provide(Y:output.label, X:input.raw_data) ⟵map(X,Y)`

interaction goals and also require different implementation. Our motivation to develop an *extensible* language is to enable developers to refine or create the language elements at all levels, from primitives to high-level specification of actions and patterns. In the next section, we provide a set of action definitions and interaction patterns from existing HAI interactions and frameworks.

3.3 Representing HAI Interactions Using Primitives and Patterns

In this section, we describe how the proposed primitives can be used to define actions and interaction patterns. Such patterns serve a given goal during the interaction and can be applied for the design of other interactions. For the *interactive robot learning for emotion recognition* (Section 3.1), we define the following messages and actions (Table 3):

Interaction Patterns. The defined messages and actions can semi-formally specify (a) the intent of the sender's act (primitive), (b) the type of the communicated information (types and operations), and (c) how the information is communicated (modifiers). Based on the unpacking of the interactions, we define the following interaction patterns: `class_selection ≡ [A1,A2]`, `new_class_sample ≡ [A3,A4]` and `sample_annotation ≡ [A5,A6]` (Table 4). All patterns describe a query-response interaction initiated by the robot (model). The goal of the first pattern is to set the target class, without requiring any information about the model input. The class represents the user's emotional state selected from a predefined list of emotions (after an emotion elicitation activity). The goal of the second pattern is to receive a model input for the target class (emotion), resulting to a training example (input–output pair). The robot asks the user to demonstrate the selected emotion (model input—waling and standing). For the third pattern, the model captures the gait/thermal data and makes a prediction for the user's emotion. This prediction is not communicated to the user but it is used for the design of the interaction; if the prediction is inaccurate, the robot interacts with the user and uses the response to retrain its model. These patterns can be used to design the interactions between the human user and the model (Figure 4).

Design and Implementation Choices. The selection of the patterns and the actions that define them can provide insights about the design and implementation aspects. For example, designing

Table 4. Interaction Patterns for the Interactive Robot Learning Interactions

Pattern	Actions	Message description
class-selection	`req-class_selection(Y,L)`	model asks user for a class from list
	`select-class(Y,L)`	user selects a class from a list
new_class_sample	`req-new_class_sample(X,Y)`	model asks user to provide a sample for the given class
	`generate-class_sample(X,Y)`	user provides a sample for the given class
sample-annotation	`req-sample_class(X,Y)`	model asks user to annotate sample
	`annotate-sample(S,M)`	user provides a label for the sample

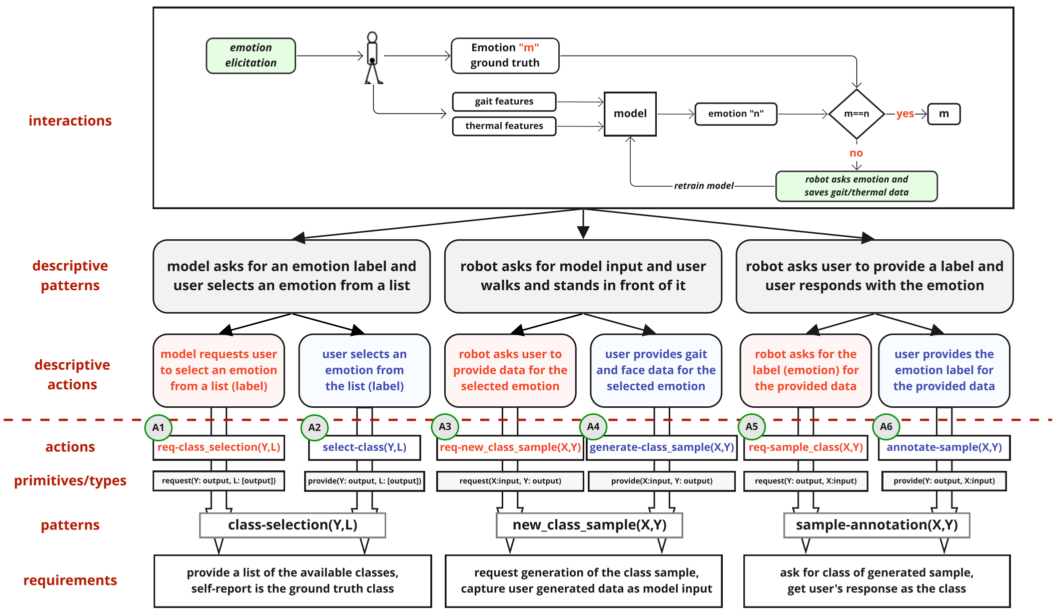

interactions
emotion elicitation
Emotion "m" ground truth
gait features
thermal features
model
emotion "n"
m==n
yes
m
no
retrain model
robot asks emotion and saves gait/thermal data
descriptive patterns
model asks for an emotion label and user selects an emotion from a list
robot asks for model input and user walks and stands in front of it
robot asks user to provide a label and user responds with the emotion
descriptive actions
model requests user to select an emotion from a list (label)
user selects an emotion from the list (label)
robot asks user to provide data for the selected emotion
user provides gait and face data for the selected emotion
robot asks for the label (emotion) for the provided data
user provides the emotion label for the provided data
actions
A1 req-class_selection(Y,L)
A2 select-class(Y,L)
A3 req-new_class_sample(X,Y)
A4 generate-class_sample(X,Y)
A5 req-sample_class(X,Y)
A6 annotate-sample(X,Y)
primitives/types
request(Y: output, L: [output])
provide(Y: output, L: [output])
request(X:input, Y: output)
provide(X:input, Y: output)
request(Y: output, X:input)
provide(Y: output, X:input)
patterns
class-selection(Y,L)
new_class_sample(X,Y)
sample-annotation(X,Y)
requirements
provide a list of the available classes, self-report is the ground truth class
request generation of the class sample, capture user generated data as model input
ask for class of generated sample, get user's response as the class

Fig. 4. Description of the HAI interactions using definitions of interaction primitives and patterns. The diagram illustrates our unpacking approach and the process of describing interaction primitives and patterns in terms of the interaction requirements.

the `class-selection` interaction requires an interface for the user to choose an emotion so it can be communicated to the model (robot). For the `new_class_sample` pattern, the model needs to capture the gait/thermal data and process them for a model prediction (feature extraction). In terms of technical aspects, the implementation does not consider user's input uncertainty, making the assumption that the user provides the correct data (model input and output).

Such choices can affect both the performance of the model and its interaction with the user. Different actions or patterns can lead to different types of interactions between users and models, e.g., negotiation, which may require different methods for implementation, e.g., shared decision-making. Based on the unpacking process of HAI interactions to primitives and patterns on this running example, we compiled a list of three patterns, and the actions that define them, which can describe the interactions between the user and the robot (Table 4). Following the proposed formalization for actions and patterns, this approach aims to (a) characterize existing interactions and (b) modify or design new interactions. Our motivation is to compile a list of commonly used actions and patterns, toward a design space for HAI interactions.

4 From Interaction Primitives to a Design Space for HAI Interactions

In this section, we discuss how the proposed formalization can be used toward the definition of a design space for HAI interactions. We demonstrate how the proposed interaction primitives and patterns can be used as design materials to prototype HAI interactions. We highlight how differences in patterns can serve different interaction goals and concepts. The goal of the proposed design space for HAI interactions is to support AI designers and practitioners by providing appropriate design and implementation choices for a given interaction concept. Based on the literature review and the unpacking of existing HAI interactions, we provide an overview of how interaction patterns can be designed for given interaction paradigms, aiming to bridge the gap between high-level guidelines and implementation requirements.

4.1 A Collection of Actions and Interaction Patterns

Based on the unpacking of HAI interaction use cases and frameworks (Section 2), we compiled a list of actions, messages and interaction patterns. We provide a set of action definitions with their description (Table 5).[2] Such actions can be further specified as messages and used for a given interaction context. For example, `annotate-sample(X,Y)` describes the annotation of a sample during a classification task, as the mapping of input sample `X` to output label `Y`. A similar action, `show-policy(S,A)`, describes an RL policy as the mapping of input state `S` to output action `A`. This list is not exhaustive since it does not cover all possible actions and patterns, but rather provides examples from a range of HAI interaction scenarios. More actions can be defined following the proposed formalization, resulting to an extendable library of actions. While there are different ways to define these actions and messages for the selected use cases, the goal of this approach is to specify the intent and type of the communicated information considering existing interaction concepts and HAI interaction paradigms.

Following the proposed definitions, we provide a collection of interaction patterns (Table 6). Each pattern is described as a sequence of messages based on the action definitions and the unpacking process of the selected HAI interactions. The selection of the messages and actions plays an important role to the definition of an interaction pattern. For example, `req-class_selection` and `req-sample_class` can both describe a model's request for an output. However, the first action specifies the selection of a class from a list, while the second action requires a class given an input sample. The selection of an action for a given pattern depends on the goal of the interaction, as well as the design and technical requirements.

Combining different actions can lead to patterns with different interaction goals. For example, model queries for informative or evaluative advice aim to improve model's performance, while XAI-based interaction patterns can be used to support the user by providing justifications about model predictions. This collection captures a range of interaction patterns in terms of the interaction concept and requirements for the selected use cases. Combinations of interaction patterns can serve multiple goals and concepts. For example, providing explanations to users (XAI) can enhance the quality of human feedback (HITL) resulting in collaborative learning and HI systems. The long-term goal of this research is the formalization of a new design space for HAI interactions which will support designers to explore, modify, and apply interaction patterns by providing design and implementation choices toward the prototyping of new types of interactions between human users and AI models. These collections of actions and patterns are extracted through the unpacking of existing systems with HAI interactions. We demonstrate that our formalization can describe a wide range of interactions and patterns based on a set of interaction primitives. However, these collections do not provide a comprehensive set of modeling materials for HAI interactions, and may

[2]A detailed description of the action definitions is included in Appendix.

Table 5. A List of Action Definitions and Their Description, Extracted by Unpacking Existing HAI Interactions into Interaction Primitives

Action definition	Description
`req-class_selection(Y,L)` ≡ `request(Y:output.label,L[output.label])` → `select(Y,L)`	request the selection of a class Y from list L
`select-class(Y,L)` ≡ `provide(Y:output.label, L:[output.label])` → `select(Y,L)`	select a class Y from a list L
`req-new_class_sample(X,Y)` ≡ `request(X:input.raw_data|fvector, Y:output.label)` → `create(X),map(X,Y)`	ask for a new sample X for a given class Y
`req-class_sample(X,Y)` ≡ `provide(X:input.raw_data|fvector, Y:output.label)` → `select(X), map(X,Y)`	select a sample X of a given class Y
`req-sample_class(X,Y)` ≡ `request(Y:output.label, X:input.raw_data|fvector)` → `map(X,Y)`	ask for the class Y of a given sample X
`req-gsample_class(X,Y)` ≡ `request(Y:output.label, X:input.raw_data|fvector)` → `create(X),map(X,Y)`	request the annotation of a generated sample
`req-sel_sample_class(X,Y)` ≡ `request(Y:output.label, X:input.raw_data|fvector)` → `select(X),map(X,Y)`	request the annotation of a selected sample
`annotate-sample(X,Y)` ≡ `provide(Y:output.label, X:input.raw_data|fvector)` → `map(X,Y)`	annotate a given sample
`show-policy(S,A)` ≡ `provide(Y:output.action, X:input.state)` → `map(S,A)`	show selected action A for state S
`give-evaluative_advice(S,A,R)` ≡ `provide(R:feedback.eval, [X:input.state,Y:output.action])` → `select(R), map(S,A,R)`	select a reward R for the mapping from S to A
`modify-prediction(X,Y,Z)` ≡ `provide(Z:output.label,[X:input.raw_data|fvector, Y:label])` → `modify(Y,Z), map(X,Z)`	modify prediction of X from Y to Z
`show-candidate_samples(CS,S)` ≡ `provide(CS:[input.raw_data], S:[input.raw_data]])` → `select(CS,S)`	show a list of candidate samples CS based on sample S
`select-sample(X,CS)` ≡ `provide(X: input.raw_data, CS:[input.raw_data]])` → `select(X,CS)`	select sample X from a list of samples CS
`modify-sample(X,M)` ≡ `provide(M:input.raw_data,X:input.raw_data)` → `modify(X,M)`	modify a sample from X to M
`generate-sample(X)` ≡ `provide(X:input.raw_data)` → `create(X)`	modify a sample from X to M
`modify-mparams(P,M)` ≡ `provide(M:input.mparams,X:input.mparams)` → `modify(X,M)`	modify model parameter from P to M
`modify-features(X,M)` ≡ `provide(M:input.fvector,X:input.fvector)` → `modify(X,M)`	modify a feature vector from X to M
`req-prediction_evaluation(X,Y,F)` ≡ `request(F:feedback.eval,[X:input.raw_data, Y:output.label])` → `select(F), map(X,Y,F)`	ask for feedback F to evaluate the mapping of X to Y
`evaluate-prediction(X,Y,F)` ≡ `provide(F:feedback.eval,[X:input.raw_data, Y:output.label])` → `select(F), map(X,Y,F)`	select feedback F to evaluate the mapping of X to Y
`show-prediction_XAI(X,Y,F)` ≡ `provide(F:feedback.XAI,[X:input.raw_data, Y:output.label])` → `map(F,X,Y)`	provide explanation for the mapping of X to Y

not cover all existing types of interactions. Our proposed formalization allows for the extension of these collections, by defining new types of actions and interaction patterns as modeling elements.

4.2 Interaction Patterns as Design Materials

HCAI approaches for explainability, transparency, and human control provide opportunities to design new types of interactions, e.g., model auditing and contestation, negotiation, shared decision

Table 6. A List of Interaction Patterns as Sequences of the Defined Actions/Messages

Patterns	Actions	Description
class-selection	`req-class_selection(Y,L)`	request for a class from a list
	`select-class(Y,L)`	select a class from a list
new_sample	`req-new_sample(X)`	ask for a new input sample
	`generate-sample(X)`	generate an input sample
new_class_sample	`req-new_class_sample(X,Y)`	request a new sample for a given class
	`generate-class_sample(X,Y)`	generate a sample for a given class
sample-annotation	`req-sample_class(X,Y)`	ask for the class of a given sample
	`annotate-sample(X,Y)`	select a class for a given sample
new_sample-annotation	`req-new_sample(X)`	ask for a new input sample
	`req-gsample-class(X,Y)`	ask for the class of a new sample
candidate_samples	`req-candidate_samples(CS,S)`	ask for a set of candidate input samples
	`show-candidate_samples(CS,S)`	show a set of candidate input samples
sample-modification	`req-modified_sample(X,M)`	ask for a modified input sample
	`modify-sample(X,M)`	modify an input sample
feature-modification	`req-modified_feature(X,M)`	ask for a modified feature vector
	`modify-feature(X,M)`	provide a modified feature vector
parameter-modification	`req-mparam-modification(P,M)`	ask for a modified model parameter
	`modify-mparam(X,M)`	modify a model input parameter
prediction-modification	`annotate-sample(X,Y)`	provide a label for a sample
	`modify-prediction(X,Y,M)`	modify a prediction of a given sample
policy-visualization	`show-policy(S,A)`	show selected action for current state
informative_advice	`req-informative_advice(S,A,B)`	ask for informative advice based on state-action
	`give-informative_advice(S,A,B)`	modify (or not) the selected action
evaluative_advice	`req-evaluative_advice(S,A,B)`	ask for evaluative feedback for state-action
	`give-evaluative_advice(S,A,B)`	evaluate the state-action pair
prediction-based_XAI	`req-prediction_XAI(X,Y,F)`	ask for explanations for a given input–output
	`show-prediction_XAI(X,Y,F)`	show explanation for a given input–output pair
outcome-evaluation	`req-outcome_evaluation(Y,F)`	request evaluative feedback for a given outcome
	`evaluate-outcome(Y,F)`	provide evaluative feedback for a given outcome
prediction_parameters	`req-prediction_params(X,Y,P)`	request predictions with model output parameters
	`show-prediction_params(X,Y,P)`	show predictions with model output parameters
turn_taking-evaluation	`generate-and-turn(X,Y)`	request a modified sample based on a generated input
	`capture-and-generate(X,Y)`	provide a modified sample based on input
	`evaluate-outcome(Y)`	provide evaluative feedback based on outcome
prediction-with-XAI	`select-sample(X)`	select a sample
	`show-prediction_XAI(F,X,Y)`	show explanation for the sample prediction
	`modify-annotation(X,Y,M)`	modify the outcome based on the input–output pair
recommendations	`req-recommendations(M,R)`	modify a model input parameter
	`show-recommendations(M,R)`	show recommended items for a given model input
	`evaluate-recommendation(S,V)`	evaluate a selected recommended item (accept/reject)

making, and others. HAI interactions can utilize the available information provided by the models, apart from decisions and predictions. For example, model uncertainty can be measured, communicated, and used as a design material through transparency [14]. AI designers and practitioners need to consider interactions with AI as a design material, based on its capabilities, limitations, and the challenges that may arise while designing for transparency, unpredictability, learning, and shared control [46]. Our proposed formalization for interaction primitives and patterns aims to be *accessible* in order to enable AI designers and practitioners to explore appropriate patterns for a given set of design and technical requirements.

The following example illustrates two alternatives of a query pattern (Figure 5), where the user queries the model for a prediction using a model input—sample (`req-sample_class(X,Y)`), receives the model's prediction, and provides a modified prediction (`modify-prediction(X,Y,Z)`). Using the same user actions, we can define two query patterns based on two different model actions.

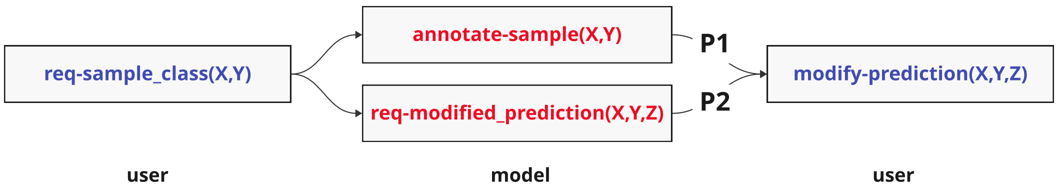


Fig. 5. Query patterns. The user asks for a model prediction for a given input. Based on pattern P1, the model provides the prediction and the user chooses to modify the prediction. Based on pattern P2, the model makes an explicit request for a modified prediction.

The first model action describes the annotation of a provided sample, while the second one describes the modification of a label for a provided sample-label pair. More specificallly, we define

—**annotate-sample (X, Y)** ≡ `provide (Y:output,X:input)←...`, and
—**req-modified_prediction (X, Y, Z)** ≡ `request (Z:output,[X:input,Y:output])←...`

In the first case (P1), the model provides the user with a prediction on user's input and the user contests the prediction by providing an alternate output. The second case (P2) describes a model which queries the user to provide an alternative output given its prediction on the user's input. In terms of design and implementation choices, the first query alternative may require a mechanism to decide who makes the decision based on the quality of user/model predictions and how to update the model based on user's prediction (hybrid decision making). For the second query, the model is designed to explicitly ask the user for an alternative output and could be an example of AL, where the model requests human annotation for (uncertain) predictions. In that case, human's decision should have a larger effect on the model updates, compared to the first version. Such patterns can describe the interactive control feature for the design of *contestable AI* interactions [7], which enable the users to intervene and modify a decision made by the system. Through these two versions of a query, we show how interaction primitives can be used as design materials and generate appropriate patterns for a given interaction goal.

4.3 Prototyping HAI Interactions

A key aspect of the proposed design space is the ability to create and explore alternatives of interactions by selecting different patterns or actions. The motivation is to support practitioners while sketching and prototyping interactions with models by providing insights about the human-centered design aspects of model development and deployment toward an *accessible* and *useful* modeling approach [53]. We demonstrate how different sequences of patterns can result to different types of interactions. We consider a set of alternative designs for the interactive robot learning scenario [109], using the defined interaction primitives and patterns as design materials. For each design alternative, we provide a description of the design and technical aspects, as well as the interaction concept (Figure 6). The first design (D1) represents the original interaction design (Figure 3); a robot-initiated interaction where the user provides an annotated sample in a query-response manner. The second design (D2) describes an interaction where the robot communicates its prediction for the user's emotion, followed by a request for user's validation. This is achieved by replacing the last pattern (`annotate-sample`) with a pattern for evaluating the model's prediction (`evaluate-prediction`). The third design (D3) can describe a system where the user asks the robot to make a prediction which they can modify. This design introduces two patterns to allow the user to request model's prediction for the generated sample (`new_sample_and_class`) and modify it (`modify-prediction`). The fourth design (D4) includes an additional XAI-based interaction

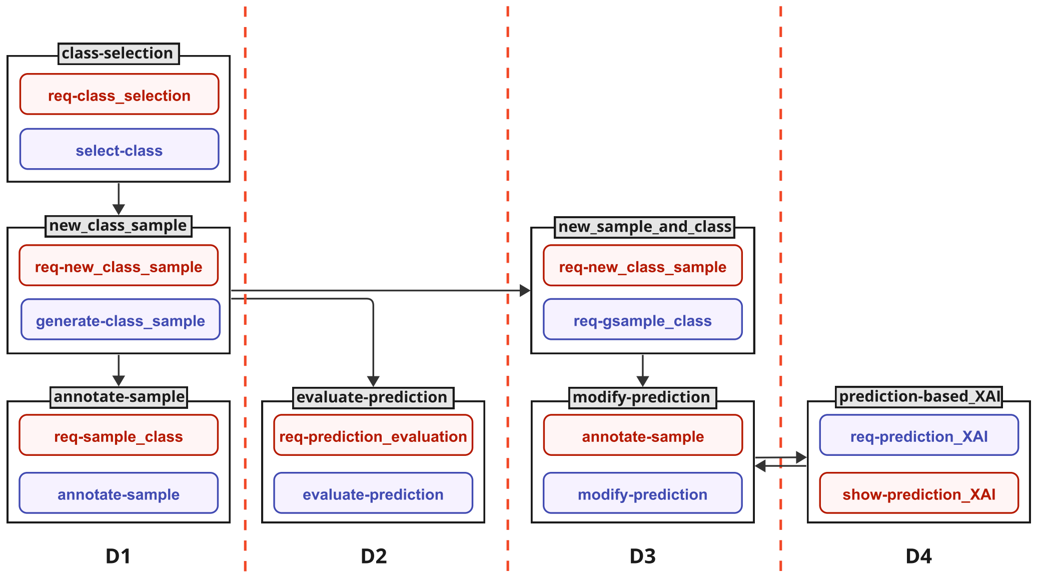


Fig. 6. Design alternatives for the interactive robot learning interactions [109]. Different sequences of patterns result in a set of design alternatives for different interaction concepts. Each alternative is related to specific design/implementation aspects and requirements. D1 describes the original design. D2 describes a robot-initiated interaction for user feedback. Based on D3, user asks the robot for its prediction based on the generated sample which can be modified. D4 adds model explanations for the robot's prediction.

pattern, where the model provides explanations about its prediction to the user (`prediction-based_XAI`). These alternatives can characterize different types of interactions with respect to the goal of the interaction. The original design is proposed as an interactive robot learning approach where the goal is to evaluate and improve the model's predictions through its robot-initiated interactions with the user. The second design requires the user to be more active in the interaction through an interaction pattern for model's prediction and user's feedback (evaluation). Based on the third design, the user initiates the interaction for the model's prediction. Finally, adding the XAI interaction pattern enables the user to further interact with the robot for explanations.

These designs are also related to specific design and implementation aspects. The original design assumes that user's provided sample and self-reported emotion are accurate. The second one requires an interaction where the robot communicates the prediction and asks for user's evaluation. Such an interaction requires an appropriate learning mechanism which can integrate user's feedback to the learning process. The XAI-based interaction requires the implementation of a specific explanation method. These alternatives can also be related to different roles of users involved in the interaction. The first design (D1) has been proposed for an end-user who implicitly participates in the model training process. The fourth alternative (D4) could be a design for an interaction between the model and the developer for model evaluation, where XAI is used to enhance the user's understanding about the system capabilities, e.g., identify "hard-to-predict" emotions.

Considering multi-user HAI interactions, different types of users can participate in the interaction for different goals. For example, the robot-based game interactions [96] are divided into two interaction loops: player-AI and supervisor-AI interaction. Each loop serves a specific interaction goal. The player-AI interaction goal is to elicit feedback from the player implicitly in order to personalize the model's decisions, while the supervisor-AI loop aims to enhance safety through the

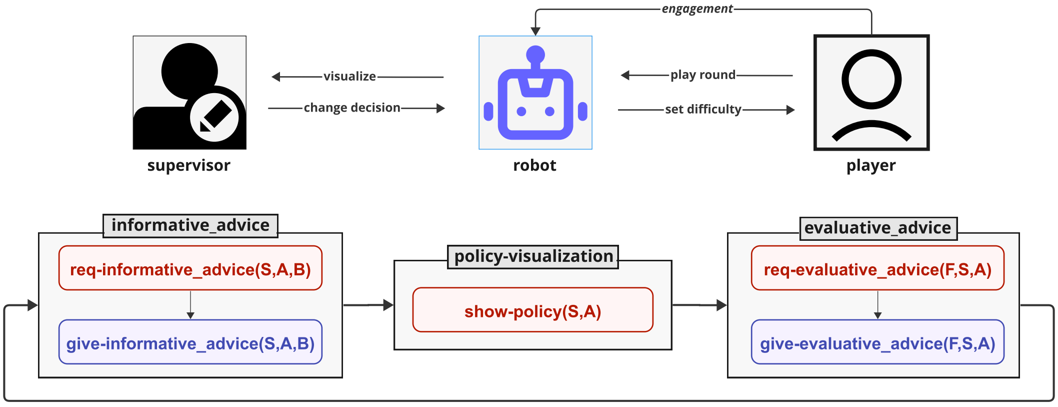


Fig. 7. Interaction patterns for the robot-based multi-user interaction [96]. The robot visualizes its policy to the supervisor (`policy-visualization`) who can alter it ((`informative_advice`)). The robot receives implicit feedback from user's engagement through EEG (`evaluative_advice`). Both types of feedback are integrated to model learning updates. EEG, Electroencephalography.

interventions of a supervisor using a transparent interface. Considering these, different interaction patterns should be used to serve each goal (Figure 7).

The patterns describe the interactions between the robot, the player, and the supervisor. The robot adjusts its policy (difficulty selection) based on user's performance and engagement, and the supervisor's interventions. Two main design aspects of the proposed system are (a) transparency and explainability and (b) user feedback and control. Considering the player-AI interactions, the model communicates the selected difficulty through the robot's announcement and the player provides implicit feedback through task performance and engagement. Task performance is measured based on the user's response, while engagement is measured through an **Electroencephalography (EEG)** headset. For the supervisor-AI interaction, model's transparency through the UI aims to enhance user's decision making by providing appropriate interventions. The supervisor can provide explicit feedback to the model by either accepting the robot's decision or proposing an alternate one. Considering both types of interactions and their patterns, a learning mechanism is required to learn from both types of users in an online way so the robot can dynamically improve its policy and select the appropriate levels of difficulty. The goal of the interaction is to include both types of users in the personalization process.

Designing multi-user interactions may require combining interaction patterns for different goals. Considering the framework for contestable AI [7], we provide a description of possible interactions during model contestation, highlighting the different interaction concepts between users and the AI model (Figure 8). The framework follows the paradigm of mixed-initiative interaction, based on which human controllers and decision subjects can both participate in the decision-making process. Decision subjects can interact with the system to negotiate a decision which affects them. Such decisions may be the outcome of decision support interactions between the model and the human controller (semi-automated decisions). In order to support such types of interactions, the proposed framework provides the following features: *interactive controls*, *explanations*, and *intervention requests*. Interactive controls enable both types of users to provide feedback to the system for different purposes. Explanations are used to provide justification for the model's decisions and aim to support the semi-automated decision making process. Intervention requests enable the decision subject to initiate a model auditing process. XAI-based interaction patterns can be used to (a) make a user aware of a decision made by the system, (b) inform the user about how to contest a

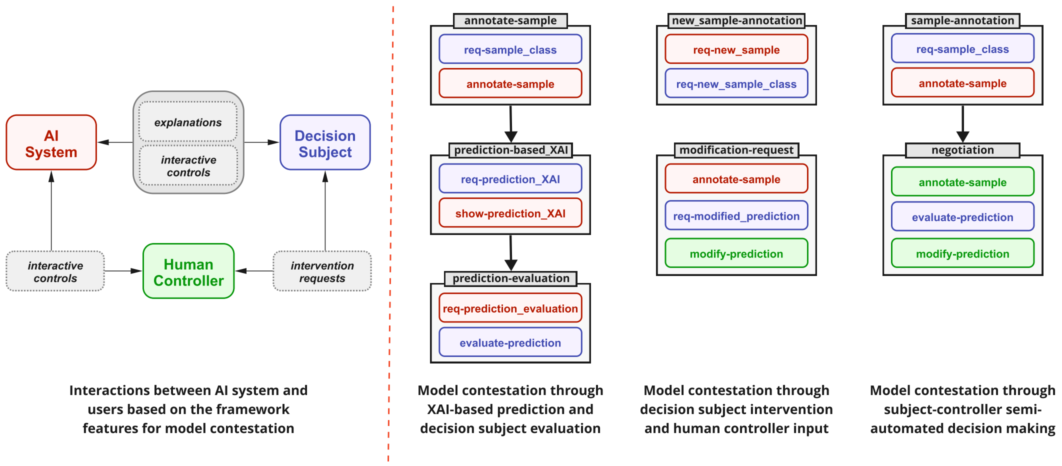


Fig. 8. Examples of interaction patterns for contestable AI interactions [7]. We describe three examples of interactions based on the proposed framework for mixed-initiative interactions between the AI system, the decision subject, and the human controller. Each interaction describes a different contestation aspect using interactive controls, explanations, and intervention requests.

model decision, and (c) provide an explanation to justify the decision of the system. HITL-based interaction patterns can provide both types of users with the ability to negotiate and even override AI decisions (interactive controls). Moreover, collaborative learning interactions can enable both users and system to learn from their interactions and augment their decision making toward HI.

Considering the above, we provide examples of interaction based on our defined actions and patterns (Figure 8). The first example is a combination of three interaction patterns and describes the communication between AI and the decision subject. During the interaction, the user asks for the model's prediction for a sample (`sample-annotation`) and for explanations for this prediction (`prediction-based_XAI`). The subject can utilize the explanations to evaluate the outcome. if needed (`prediction-evaluation`). The second example is a multi-user interaction, where the decision subject needs to generate a new sample (e.g., submit a form) and ask for a decision (`new_sample-annotation`). The AI provides its decision to subject who makes a request to the human controller for the modification of the decision (`modification-request`). Finally, the third example describes a semi-automated decision making process, where the human controller can provide a decision (considering the AI's output) and modify it based on the decision subject's evaluative feedback for the model's prediction (`negotiation`).

We demonstrated how the proposed formalization can be used toward prototyping HAI interactions using patterns. In order to design and implement such interactions, we need to consider the interaction goals that each pattern can serve. The motivation for an *extensible* modeling language is to enable the specification of interaction patterns in terms of design and implementation aspects. Toward this, based on our literature review (Section 2) and the extracted patterns and actions (see Tables 5 and 6) from the unpacking process of HAI interaction use cases, we provide an overview of how the identified patterns can be used in the context of different concepts for XAI-based, HITL-based, and HI-based interactions. The goal of the overview is to identify possible relations between interaction patterns and requirements.[3]

[3]An extended description of the use cases is included in Appendix.

4.3.1 XAI-Based Interactions. Human-XAI interactions can be designed for several interaction concepts and goals (e.g., debugging, persuasion, decision support). Designing explainable and transparent models is not trivial, especially while considering the various parameters that can affect the interaction, e.g., user's expertise, perception and understanding, cognitive load, preferences, and so forth. From the unpacking process, we identified the following XAI-based patterns and interaction goals:

- *XAI-based interactions can manage user's expectations about the AI model's behavior.* A Meeting Scheduling Assistant [51] uses explanations to calibrate user's trust and expectations about its model predictions. The model predicts if an email is a meeting request to help the user make a decision. The `prediction-with-XAI` pattern enables the model to be transparent by communicating its accuracy rate along with the prediction. The system visualizes the model's accuracy rate, as well as a set of prediction/explanation examples with different levels of uncertainty to enhance user's understanding.
- *XAI-based interactions can enhance user's perception about the model's performance.* In the context of AL for emotion recognition [44], the proposed NOVA systems aims to facilitate the selection of appropriate samples for model refinement. Based on the unpacking process of the proposed system, we identify two patterns with different goals: the `prediction-parameters` pattern aims to support the selection of appropriate samples for annotation by visualizing the confidence of predictions, and the `prediction-based_XAI` pattern is used to support the annotation process by providing additional information about the model's prediction for a given sample through visual explanations.
- *Model transparency can support human trainers while providing feedback to iML models.* Policy visualization [15] serves as a transparency method to engage the user to provide feedback to the model during task performance. The model utilizes user feedback to facilitate its learning process, i.e., faster convergence to the optimal policy. Based on our unpacking approach, the `policy-visualization` pattern is used to communicate the model's current policy, by visualizing the current state (input) and the selected action (output).
- *Explanations can justify model's prediction based on user preferences.* For an music recommendation system [60], explainable user (preference) models are used to enhance user's perception about their own preferences and how these affect model's recommendations. Considering this, the `prediction-based_XAI` pattern is used to provide predictions and justify them through explanations based on user's preferences. In terms of design aspects, different visualizations (and explanation methods) are required considering the individual characteristics of users, e.g., need for cognition.

4.3.2 HITL-Based Interactions. The goal of HITL methods is to efficiently integrate the human user to the learning and decision making process of an AI system. Human users can participate in the model's development and deployment phases. The selection of appropriate iML/HITL methods and approaches depends on several aspects of the interaction, including user role and expertise. We present a set of HITL-based patterns extracted from the unpacking process, considering the different interaction goals.

- *HITL methods can be used for interactive data collection and labeling.* For the interactive robot learning scenario for emotion recognition (Figure 3), the model utilizes human–robot interaction data in order to evaluate its predictions and collect training data. Model re-trains without making the user aware of their participation in the data collection, annotation and training processes. Based on our unpacking, `sample-annotation` defines a human labeling action given a generated sample.

— *HITL methods can be used for model improvement by relabeling uncertain predictions.* In the context of AL, the NOVA system asks the user to select uncertain samples and modify their predictions. In terms of the interaction design, the `prediction-modification` pattern enables the user to validate the model's prediction or provide a new label. Manual corrections are used to update the model. Since the user makes the final decision (label), XAI methods are used to enhance user's perception and, thus, the quality of feedback.
— *Feedback interfaces should be user-friendly and intuitive in order to ensure human feedback quality.* Training an RL agent through human advice requires an appropriate learning methods to integrate human feedback. The interaction supports two types of feedback. The `informative-advice` pattern is used to receive human advice in the form of a corrective action and the `evaluative-advice` pattern describes the evaluation of the model's decision in the form of binary feedback. These types of feedback are integrated through different feedback interfaces and learning methods (policy/reward shaping).
— *Users can control model predictions and parameters.* For the Meeting Scheduling Assistant, HITL methods enable the user to (a) provide feedback for a prediction by accepting or rejecting it and (b) control the model's sensitivity parameters through a UI slider, affecting the model's predictions. The `modify-prediction` action enables the user to validate (or not) a prediction and the `modify-mparams` action allows the user to adjust the model's sensitivity parameters until they are satisfied with the model predictions.

4.3.3 Collaborative Learning and HI. The goal of collaborative learning and HI interactions is to enable both humans and machines (AI systems) to learn from each other in a collaborative manner. Collaborative learning interactions can be designed by combining XAI-based and HITL-based interactions, enabling both users and AI models to exchange information while solving a task. Based on our unpacking process, we identify and discuss interaction patterns used in collaborative learning and HI interactions.

— *User can control and evaluate the collaboration with a model.* In the context of a robot-based collaborative sketching interaction [56], both user and robot work together during the co-ideation process in a turn-taking interaction. Both agents generate ideas building on their partner's generated sketch. The model uses the captured image to generate a variation of the user's sketch—to provide alternative ideas. During the interaction, the user can control what the robot will capture as an input by moving the robot and also provide feedback for the generated outcome. The `turn_taking-evaluation` pattern is repeated until the user is satisfied. The model uses image classification for the user's input and generates a variation of this. The goal of such interactions is for the user to explore and identify new insights, rather than to identify a correct solution.
— *Model can support the user through semi-automated decision making.* An interactive sound segmentation system enables the user and the model to work together in order to complete an audio segmentation and annotation task [49]. The model supports the user by providing a list of candidate samples which can be selected, edited and annotated by the user, toward a collaborative interaction. The `candidate-samples` pattern requires a model mechanism to identify the candidate samples and a proper visualization to highlight the segments. Based on this visualization, users provide feedback to adjust the model's predictions.
— *Explanations can be provided both by users and models to justify their predictions.* In a game-based scenario [42], XAI and iML methods are used to enhance user's trust about the model's decisions and allow for corrections. The `turn-taking_XAI` pattern describes an interaction where both user and model communicate their prediction justification in the form of rule-based XAI. In terms of the design and implementation of this interaction, an appropriate

visualization of the rules is needed to enable the user to understand the reasoning of the model in order to provide appropriate modifications and justification.

Based on this overview, we can observe that each pattern can serve a specific goal within the interaction. For example, considering the XAI-based patterns, the `prediction-with-XAI` pattern is used to calibrate end-user's trust, while `prediction_XAI` is used to enhance user's understanding about a prediction. For HITL-based patterns, `annotate-sample` enables a user to improve a model by providing annotations, while `modify-mparams` is used as a control pattern which enables the user to alter the model's predictions until the user is satisfied with the decision. Considering possible commonalities and differences between patterns and goals, we envision a design space which can provide suggestions for interaction patterns based on a given interaction concept. This would allow for fast prototyping of interactions using patterns, considering the interaction goals and requirements.

5 Discussion

5.1 Evaluation and Limitations

In this article, we introduce a semi-formal representation for HAI interactions which uses low-level communicative acts to describe the information exchanges between users and models. We provide a definition of our proposed language based on a set of interaction primitives, which can enable users and models to exchange information between each other through a message passing protocol. Based on a set of use cases and frameworks for HAI interactions, we demonstrated how we can build patterns of HAI interactions and characterize them in terms of design and implementation aspects. Following existing frameworks for the development and evaluation of ACLs [95], we provide an evaluation of our current formalization based on the desired language properties. Based on these properties, we discuss the limitations of our current formalization and how these can be addressed.

- *Generic:* One of the goals of our proposed language is to support the modeling of a wide range of interactions between multiple human and AI actors, independently of the underlying theories, frameworks, and implementation environments for AI system design. Following our unpacking approach of HAI interactions, we demonstrate how our formalization can describe interactions from a range of AI models, approaches, and methods. However, further investigation is needed to ensure that the proposed formalization can adequately describe all possible HAI interactions, without being bound on specific AI technologies and interaction paradigms.
- *Theoretically sound:* Our proposed formalization was developed based on existing concepts and modeling elements for ACLs. Our interaction model is structurally similar to the LSC [78] and can describe the exchange of messages (provide or request information) with a specific content (model data types) between agents (human/AI actors). However, it currently lacks a formal definition of semantics to represent how this communication of messages can affect the agents' knowledge, beliefs and intentions.
- *Comprehensive:* Our proposed formalization is defined based on the review of existing frameworks, paradigms, and use cases of HAI interactions (Section 2). Through the unpacking of HAI interaction use cases (Section 3.1), we demonstrated that the current formalization using interaction primitives and patterns can describe a range of interactions and frameworks, covering different interaction paradigms and methods for XAI, HITL, and HI systems. It is possible that the selected use cases and framework may not cover all possible types of interactions between users and models; a systematic review should be carried out to ensure

that our system covers the complete range of HAI interactions and real-world systems and emerging technologies.

— *Consistent:* Considering that the specified notations and the syntax and semantics of the language must be consistently specified and not contradict with each other, our current formalization lacks a formal definition of semantics in order to assess its consistency as a semi-formal modeling language.
— *Well specified and documented:* The modeling elements of our language, i.e., interaction primitives, types, and actions (Table 1), are defined based on concepts from ACLs. The current formalization allows for the specification of actions (Tables 5 and 7), messages (Figures 6–8) and interaction patterns (Table 6). Our current formalization lacks a formal definition and documentation for the language semantics.
— *Extensible:* The proposed formalization is designed to cover a broad set of relevant paradigms and modeling approaches for HAI interactions; however, it is essentially impossible to cover all aspects inclusively. In case that our proposed formalization is insufficient for specific cases and modeling approaches, several mechanisms can be used to extend or customize it as required. Toward this, our formalization allows for the specialization, refinement, and extension of the proposed modeling components (Table 5). We provide a description of how the defined patterns can be characterized in terms of implementation and design choices (Section 4.3), which can enable AI designers and developers to specify interactions at different levels, or even refine the modeling components, i.e., primitives, types, and operations, if needed.
— *Automatable:* Our aim is to develop an executable language which can provide a computational representation of the modeling elements, i.e., actions, messages, and patterns. Our current formalization can encode such constructs based on a set of low-level objects (primitives and types). We have not yet demonstrated that the language carries enough of the semantics required for execution. In particular, model processes, data operations, and error handling are not currently represented. Model-informed prototyping [89] using low-level representations [53, 102] can be used to provide simulations of the interactions between users and the underlying model processes, e.g., model inference and updates.
— *Easy to use:* One of the main goals of our proposed language is to provide a formalization of the modeling elements which can be easy to learn and apply. Toward this, we defined a simple but descriptive vocabulary for HAI actions and patterns which can be used to formalize more complex interactions (Section 4.1). Through our unpacking process (Section 3.1), we demonstrate how descriptive actions and patterns can be defined using communicative acts in order to describe existing interactions and systems through a message-passing protocol. In order to evaluate the accessibility of our language, our proposed formalization can be tested with designers and AI engineers, through (co-)design workshops and usability studies. Finally, our main motivation is to develop a language which can be useful as part of the design and implementation process of HCAI systems. To this end, we demonstrated how our current formalization can provide ways to link HCAI implementation methods to human-centered design aspects (Section 4.3).

5.2 Envisioned Applications

We envision this work as the basis of a design space which can enable users to explore and choose between existing patterns and modify them toward new types of interactions, supported by prototyping tools that provide suggestions about the design and implementation aspects of existing and new patterns. The collection of interaction patterns developed here (Tables 5 and 6) forms the start of a design pattern language. The development of such a design pattern language is an iterative process which includes prototyping of design patterns to create new solutions and remove inconsistencies and evaluating them in terms of their morphological and functional completeness [73]. Based on the current collection of actions and patterns, we can explore two directions: (1) how the language surfaces implementation choices and supports decision making (Section 5.2.1) and (2) links between interaction patterns and AI guidelines (Section 5.2.2). Developing this would lead to an extendable collection of interaction patterns, with implications for implementation, links to high-level guidelines, and links to tools that allow for rapid prototyping of HAI interactions.

5.2.1 *Manifesting Implementation Concerns.*

A main motivation for the proposed design space is the need to bridge the gap between design and implementation choices. This can be achieved by characterizing the defined interaction patterns in terms of implementation requirements. Developing a collection of interaction patterns and characterizing them in terms of implementation aspects can provide us with insights toward manifesting common implementation issues. Our goal is to develop a *comprehensive* and *extensible* language which can cover the wide range of HAI interaction paradigms, can provide mechanisms to refine and create modeling elements, and can also specify them in terms of implementation aspects. Toward this, we provide an overview of the implementation aspects for the defined patterns, considering the different HAI interaction paradigms, demonstrating our formalization as a way to annotate modeling materials, e.g., actions and patterns, with implementation choices.

XAI-Based Interaction Patterns. Based on the literature review and the unpacking process, we identify the following implementation aspects and challenges regarding XAI-based interaction patterns: (a) the selection of appropriate explanation strategies and methods considering the interaction goals [1, 9, 70], (b) the adaptation of explanations based on user's roles and characteristics (e.g., expertise, perception) [22, 54], and (c) the evaluation of XAI methods in terms of the interaction goal and user's behavior [23, 64, 83]. These can be related to the patterns found above:

- The `prediction-with-XAI` is used to help a non-technical user to understand if they can trust the model's prediction or not. This requires the model to communicate its accuracy rate along with a prediction, in an intuitive way (e.g., chart) in order to manage user's expectations about the model decisions.
- The `prediction_based-XAI` requires a proper explanation method considering the role of the user. For a domain expert user, it needs to provide appropriate information toward altering the model's decisions, while for a non-technical user, it considers the user's characteristics, e.g., cognitive load or preferences. The accuracy of the explanations is a key factor toward an effective interaction.
- The `prediction-parameters` is used to help the user identify the weaknesses of the model, e.g., prediction with low confidence. This requires the model to communicate its confidence values along with its predictions for a set of input samples, toward a scrutable model. The user is able to explore and alter the model decisions, providing feedback for model updates.
- The `policy visualization` is used to communicate the model's policy in an online fashion. In terms of implementation, the model communicates and updates its policy (input–output) during the interactions. This requires an online policy update mechanism to enable the user get an understanding of how the model's performance changes over time.

HITL-Based Interaction Patterns. The goal of HITL methods and approaches is to efficiently integrate the human user to the learning and decision-making process of an AI system. Human users can participate in the model's development and deployment phases. The selection of appropriate iML/HITL methods and approaches depends on several aspects of the interaction, e.g., goal, user role and expertise. For a given context, HITL-based patterns can be used to design interactions where the user is part of the decision making and learning process. We identify the following implementation challenges for HITL/iML-based interactions: (a) the selection of teaching strategies considering user roles and expertise [20, 24], (b) the design of intuitive and user-friendly feedback/control interfaces to ensure high-quality feedback [31, 33], and (c) the integration of feedback (models) to model updates and decision-making [62, 65]. We can relate these to the patterns above as

— `sample-annotation` is used to enable the user provide a label for a sample. A key decision relates to the quality of the user-provided data. For the interactive robot learning, the model considers the human label as the ground truth for the generated sample, based on which it performs the learning update (online training). User is able to provide training data implicitly during the interaction.
— `evaluative-advice` is used to enable a user to evaluate the model's performance. Evaluation feedback requires an interface for numerical feedback, as well as a reward shaping mechanism to integrate human feedback into model updates. User needs to be able to perceive and evaluate the model's predictions (policy) in an online manner.
— `informative-advice` is used to enable provide an alternate decision. Informative advice requires an interface for action selection, as well as a policy shaping mechanism to integrate human feedback into model updates. User needs to be able to perceive and modify the model's predictions (policy) in an online manner.
— `modify-mparams` is used as a control pattern which enables the user to alter the model's decisions by changing a model's parameter. In terms of implementation, the model should be able to dynamically alter its decisions based on the modified input. The model uses feedback only to alter its predictions for a new parameter and not to update its learning weights.

HI and Collaborative Learning Interaction Patterns. The goal of collaborative learning and HI interactions is to enable both humans and machines (AI systems) to learn from each other in a collaborative manner. Collaborative learning interactions can be designed by combining XAI-based and HITL-based interactions, enabling both users and AI models to exchange information while solving a task. Based on our unpacking process, we identify and discuss interaction patterns used in collaborative learning and HI interactions. Implementation challenges for HI and collaborative learning systems include, but not limited to (a) identify the appropriate levels of human control and AI automation both for model learning and decision making [85, 111], (b) design systems to facilitate the interaction between explainable artificial and cognitive intelligence [101], and (c) develop methods for the adaptation of HI systems considering user needs and capabilities to enhance user's perception toward improving the model's learning process [4]. Considering these, we identify the following challenges for the extracted patterns:

— The `turn_taking-evaluation` pattern is used to enable the user collaborate with a model in a turn-taking interaction. In terms of implementation, the model needs to capture the user's input during the interaction and generate a modified sketch. The user can control what the robot will capture, as well as when to provide feedback and terminate the interaction.
— The `candidate-samples` pattern is used to support the human labeling process in a semi-automated way. The user makes the final decisions based on the model's initial decisions. The model is being updated based on user's feedback. The model needs to make online updates to improve the selection of the candidate samples, which can affect the user's and thus model's performance.
— The `turn-taking_XAI` pattern enables both model and user to justify their decisions using explanations. The implementation of this pattern requires an interface where the user can modify the model's prediction and explanations. The type of explanations (visualization) depends on user characteristics, e.g., preferences, cognitive load.

While the collection of the patterns is not comprehensive, we observe that each pattern can be linked to a set of technical aspects and concerns. In order to support both design and implementation choices, our proposed design space needs to be informed by existing frameworks and guidelines for designing HAI interactions.

5.2.2 Integration with Existing Tools, Frameworks, and Guidelines. In order to get insights about the formalization of the proposed design space, we briefly discuss how our proposed approach can be integrated with existing design guidelines and computational frameworks. Our vision is a design space which can support AI designers to explore this space between design guidelines and implementation practices by enabling the collaboration of such frameworks and guidelines (Figure 1). For example, the Microsoft guidelines for HAI interactions [8] provide a set of pattern examples along with descriptions of interactions. These design guidelines can be used as a set of suggestions and pattern examples for the ideation phase of an HAI interaction system. For example, a guideline suggests helping the user understands what the system can do.[4] One of the suggested pattern examples for this given guideline is to use explanation patterns in order to enable users to gain insights into system capabilities *(XAI-based interaction patterns)*. Based on another guideline, the design should encourage granular feedback and enable the user to indicate their preferences.[5] A suggested pattern for this guideline is to request explicit feedback on selected system outputs in order to assess the system and help it improve over time (HITL-based interactions).

Our proposed design space can be informed by the provided guidelines and pattern examples in order to develop a collection of design patterns considering these guidelines. Working at a higher-level, the Assessment List for Trustworthy AI[6] was developed for the assessment of AI systems in terms of seven requirements specified in the Ethics Guidelines for Trustworthy AI [5]. *Human Agency and Oversight* is one of the requirements and refers to the ability of human users to make informed decisions and to monitor and supervise the system. HITL-based interaction patterns can be used to integrate the user to the decision making and model learning process. Another ALTAI guideline is *Transparency* and refers to the ability of the data, system, and AI models to be transparent and explainable to the user. XAI-based interaction patterns can describe different approaches to provide explanations to the user based on the role and intent in the interaction. Considering the assessment list and requirements, our proposed design space can provide support toward selecting appropriate patterns and interactions to comply with specific assessment items.

Apart from high-level guidelines, our proposed design space can be informed by technical and implementation frameworks for AI/ML models. Considering interactions with ML systems, implementation methods are required to enable (a) the communication of information between users and models and (b) the integration of user-provided data to the model learning (and decision making) mechanism. A classification of methods and approaches for iML systems [66] considers the development lifecycle of ML systems and provides a list of implementation choices based on a given category of methods, including interactive learning and explainability. However, designing efficient HITL-based interaction patterns requires both the selection of appropriate design choices and learning methods for feedback integration. Focusing on detecting and mitigating bias in ML pipelines, FaiPrep is an open library which extracts dataflow representations to support fairness during model development [106]. Such representations can be used in our context to further characterize HAI interactions in terms of dataflows and model/data operations, e.g., fit, predict, and update data. Finally, our proposed design space can be informed by existing approaches for design

[4]https://www.microsoft.com/en-us/haxtoolkit/guideline/make-clear-what-the-system-can-do/
[5]https://www.microsoft.com/en-us/haxtoolkit/guideline/encourage-granular-feedback/
[6]https://altai.insight-centre.org/

patterns for data-driven and knowledge-driven AI models [97], which can provide guidelines for the selection of appropriate learning methods.

5.3 Evaluation Plan

Considering the current limitations and the envisioned applications of our proposed semi-formal modeling language for HAI interactions, our evaluation plan consists of the following:

— *Evaluate the functional suitability of the language.* We will develop and evaluate the proposed language through a systematic literature review on (a) model-driven software engineering methods [63, 95], (b) human-centered design frameworks and guidelines [85, 108], and (c) computational methods for HCAI [13], toward a *theoretically sound*, *generic*, and *well-specified* language which can describe interactions independently from the AI methods, theories, and technologies used. Following existing evaluation frameworks [6], we will assess whether the proposed formalization can *consistently* and *comprehensively* describe interactions between multiple human and AI agents, considering both design and implementation aspects for HCAI systems.

— *Evaluate the usability of the language as a modeling framework for HCAI system design.* Alongside evaluating the functional suitability of the language, we will develop a *documented* language, along with *executable* representations, and conduct user studies and design workshops to evaluate if our language can support AI designers and engineers: (a) to efficiently learn and apply it in various contexts and paradigms, (b) to prototype and simulate HAI interactions based on their implementation aspects, and (c) to apply best design practices for HCAI. Overall, our aim is to ensure that our proposed language can be functional as part of a human-centered development and implementation process for AI systems.

6 Conclusion

The proposed formalization for HAI interactions is aimed at the complex space between concepts and practice, between formality and accessibility. The interaction patterns can act as intermediate level knowledge for the design and understanding of HAI systems, giving a formal representation of the configurations used in existing practices. Starting from a small set of interaction primitives and types to specify the communicated information between the interacting agents, we showed that the proposed primitives can be used to describe patterns of interactions from a range of systems, resulting in a collection of crisply defined actions and interaction patterns. We demonstrated that these patterns and actions are consistent with key HAI paradigms of HITL, XAI, and HI, and that they can be used to explore and prototype a range of alternate interactions for a given situation.

This space has been built from theoretical ideas about communication between humans and models, based on ideas from ACLs, design patterns, and HAI interaction paradigms. It has then been developed and tested with examples of systems and interaction paradigms from the literature, demonstrating that it can meaningfully describe existing work. The formalization starts with data types and primitive communicative acts of providing and requesting information, and works up to high level constructs—interaction patterns—that can describe common types of interactions between users and models, aiming to bridge the semantic gap. Extracting these patterns gives people of varying technicalities a common language to talk about what a particular system is doing, by building re-usable descriptions of the interactions taking place. This level of description allows for alternative design choices to be explored, while highlighting concerns that might arise and giving a framework for implementing the interactions. This provides a way to document existing practices, re-use well-tested solutions, and also speculate about new interaction possibilities through an exploration of the design space. Finally, through focusing on the interactive and communicative

Table 7. Definitions of Actions in Terms of Primitives, Types and Subtypes, and Operations

Action	Action definition													
	Prim		Main-type			Subtype	Ref-type			Subtype	Operations			
	P	R	I	O	F		I	O	F		Select	Create	Map	Modify
select-sample(X)	■		■			raw_data					X			
generate-sample(X)	■		■			raw_data						X		
select-state(S)	■		■			state					S			
modify-sample(X,M)	■		■			raw_data	■			raw_data				X, M
modify-mparams(P,M)	■		■			mparams	■			mparams	M			P, M
modify-preferences(P,M)	■		■			fvector	■			fvector	M			P, M
select_and_modify(X,M)	■		■			raw_data	■			raw_data	X			X, M
select-class-sample(X,Y)	■		■			raw_data		■		label	X		X, Y	
generate-class-sample(X,Y)	■		■			raw_data		■		label		X	X, Y	
provide-example(X,Y)	■		■	■		raw_data,label								
select-label(L)	■			■		label					L			
create-label(L)	■			■		label						L		
annotate-sample(X,Y)	■			■		label	■			raw_data			X,Y	
select-and-annotate(X,Y)	■			■		label	■			raw_data	X		X,Y	
annotate-new_sample(X,Y)	■			■		label	■			raw_data		X	X,Y	
evaluative-feedback(S,A,F)	■				■	feedback.eval	■	■		state, action	F		S, A, F	
informative-advice(S,A,B)	■			■		action	■	■		state, action	B		S, B	A, B
modify-prediction(X,Y,B)	■			■		label	■	■		raw_data,label	B		X, B	Y, B
provide-prediction_XAI(X,Y,F)	■				■	feedback.XAI	■	■		raw_data,label		F	X, Y, F	
req-sample-selection(X)		■	■			raw_data					X			
req-sample-generation(X)		■	■			raw_data						X		
req-state_selection(S)		■	■			state					S			
req-modified-sample(X,M)		■	■			raw_data	■			raw_data				X, M
req-select-modify-sample(X,M)		■	■			raw_data	■			raw_data	X			X, M
req-class-sample(X,Y)		■	■			raw_data		■		label	X		X, Y	
req-new_class_sample(X,Y)		■	■			raw_data		■		label		X	X, Y	
req-example(X,Y)		■	■	■		raw_data,label								
req-label_selection(L)		■		■		label					L			
req-new_label(L)		■		■		label						L		
req-sample_class(X,Y)		■		■		label	■			raw_data			X,Y	
req-selection_class(X,Y)		■		■		label	■			raw_data	X		X,Y	
req-new_sample_class(X,Y)		■		■		label	■			raw_data		X	X,Y	
req-evaluative-feedback(S,A,F)		■			■	feedback.eval	■	■		state, action	F		S, A, F	
req-informative-advice(S,A,B)		■		■		action	■	■		state, action	B		S, B	A, B
req-modified-prediction(X,Y,B)		■		■		label	■	■		raw_data,label	B		X, B	Y, B
provide-prediction_XAI(X,Y,F)		■			■	feedback.XAI	■	■		raw_data,label		F	X, Y, F	

possibilities around models, the design space helps to shift thinking from a single user, single model, and single purpose viewpoint to one where various stakeholders can carry out different kinds of interaction with a single model, creating a more ecosystemic view of human-model interactions.

Appendices

A Definitions of Actions

In this section, we provide a detailed description of the defined actions, as extracted from the unpacking of existing HAI interactions (Table 7). Each action is defined based on the primitive, the main and reference types and subtypes (as indicated by the shaded parts), as well as the required operations between the including types.

B Unpacking of HAI Interactions: Use Cases

This section includes a list of uses cases to demonstrate our proposed unpacking approach. Based on the selected use cases, we extract actions and patterns for different interaction concepts, including XAI-based and HITL interactions. More specifically, the list includes an interactive sound annotation system [49], an interactive RL framework for human trainers [15], an explainable AL tool for image classification [44], a human–robot interaction for collaborative sketching [56], an music

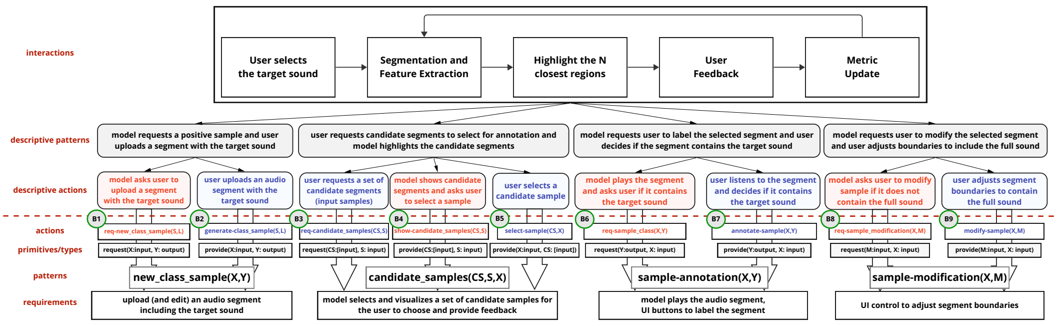

interactions
User selects the target sound
Segmentation and Feature Extraction
Highlight the N closest regions
User Feedback
Metric Update
descriptive patterns
model requests a positive sample and user uploads a segment with the target sound
user requests candidate segments to select for annotation and model highlights the candidate segments
model requests user to label the selected segment and user decides if the segment contains the target sound
model requests user to modify the selected segment and user adjusts boundaries to include the full sound
descriptive actions
model asks user to upload a segment with the target sound
user uploads an audio segment with the target sound
user requests a set of candidate segments (input samples)
model shows candidate segments and asks user to select a sample
user selects a candidate sample
model plays the segment and asks user if it contains the target sound
user listens to the segment and decides if it contains the target sound
model asks user to modify sample if it does not contain the full sound
user adjusts segment boundaries to contain the full sound
B1
B2
B3
B4
B5
B6
B7
B8
B9
actions
req-new_class_sample(S,L)
generate-class_sample(S,L)
req-candidate_samples(CS,S)
show-candidate_samples(CS,S)
select-sample(CS,X)
req-sample_class(X,Y)
annotate-sample(X,Y)
req-sample_modification(X,M)
modify-sample(X,M)
primitives/types
request(X:input, Y: output)
provide(X:input, Y: output)
request(CS:[input], S: input)
provide(CS:[input], S: input)
provide(X:input, CS: [input])
request(Y:output, X: input)
provide(Y:output, X: input)
request(M:input, X: input)
provide(M:input, X: input)
patterns
new_class_sample(X,Y)
candidate_samples(CS,S,X)
sample-annotation(X,Y)
sample-modification(X,M)
requirements
upload (and edit) an audio segment including the target sound
model selects and visualizes a set of candidate samples for the user to choose and provide feedback
model plays the audio segment, UI buttons to label the segment
UI control to adjust segment boundaries

Fig. 9. Unpacking an interactive sound annotation interaction [49] into interaction primitives and patterns.

recommendation system using personalized explanations [60], an interactive Meeting Scheduling Assistant [51], and an iML approach for game-based collaboration [42]. For each use case, we describe the interactions (unpacking) and we define the actions and patterns based on the defined primitives.

B.1 HITL Sound Event Detection and Annotation (Figure 9)

The proposed system integrates a UI for interactive sound event detection and annotation. The user sets a target sound event by selecting or uploading a sound segment which includes the target sound (e.g., door knocking). With this interaction, the user sets the possible model output classes: positive when the segment includes the sound target and negative otherwise. The model selects and highlights segments similar to the positive sample (user's input) and asks the user to classify the segments. The user can either provide a label for a segment (target or not) or adjust the segment boundaries, if the segment does not include the full sound. The model utilizes user's feedback (re-labeling/re-segmentation) to update its model parameters in an iterative and interactive manner.

We identify three types of interactions: (a) *positive sample:* the model asks the user for a positive example (segment that includes the target sound)—user provides a positive sample, (b) *select candidate sample:* user requests a set of candidate segments—model highlights the candidate samples—user selects a candidate segment, (c) *label sample:* model plays segment and asks user if it contains the target sound—user responds by labeling the segment, and (d) *modify sample:* model plays segment and asks user if it contains the full sound—user adjusts the boundaries of the segment until full sound is included. During these interactions, user and model exchange information through positive samples, visualization and selection of candidate segments and their current labels (highlighted), playing/listening to segments, and re-labeling or re-segmenting selected samples. Considering the design and implementation approach, the proposed approach focuses on two aspects: (a) the design of a human-friendly interface which can provide interactivity and feedback control, and (b) the iterative labeling and model training approach, based on which the model dynamically recalculates the model parameters (weights) based on user feedback. In terms of evaluation metrics, both user-based (interaction overhead) and algorithm-based (model accuracy) measurements were considered. Their analysis indicates that minimizing the interaction overhead (through design) can maximize machine performance and speed. Table 8 shows the messages and action definitions.

Based on these definitions, we define the following patterns (Table 9): `[B1-B2] new_class_sample`, `[B3-B5] candidate_samples`, `[B5-B7] sample-annotation`, and `[B8-B9] sample-modification`. User and model interact through input (segments) and output (labels) in an

Table 8. Messages and Action Definitions for the Interactive Sound Annotation System Interactions

Message		Action definition
B1	`<model→user,`**`req-new_class_sample(S,L)`**`, [X:uploadBtn,targetSound;Y:positiveLabel]>`	`req-new_class_sample(S,L)≡ request(X:input.raw_data,Y:output.label) ← create(S),map(S,L)`
B2	`<user→model,`**`generate-class_sample(S,L)`**`, [X:uploadTargetSound;Y:positiveLabel]>`	`generate-class_sample(S,L) ≡ provide(S:input.raw_data,L:output.label) ←create(S),map(S,L)`
B3	`<user→model,`**`req-candidate_samples(CS,S)`**`, [CS:similarSegs,highlightSeg;S:posSample]>`	`req-candidate_samples(CS,S)≡ request(CS:[input.raw_data],S:input.raw_data) ←select(CS),map(CS,S)`
B4	`<model→user,`**`show-candidate_samples(CS,S)`**`, [CS:similarSegs,highlightSeg;S:posSample]>`	`show-candidate_samples(CS,S)≡ provide(CS:[input.raw_data],S:input.raw_data) ←select(CS),map(CS,S)`
B5	`<user→model,`**`select-sample(X,CS)`**`, [X:selSegment;CS:highlightSeg]>`	`select-sample(X,CS) ≡ provide(X:input.raw_data,CS:[input.raw_data]) ←select(X,CS)`
B6	`<model→user,`**`req-sample_class(X,Y)`**`, [X:playSegment;Y:isTargetSound,labelBtn]>`	`req-sample_class(X,Y) ≡ request(Y:output.label,X:input.raw_data) ← map(X,Y)`
B7	`<user→model,`**`annotate-sample(X,Y)`**`, [X:selSegment; Y:isTargetSound,labelBtn]>`	`annotate-sample(X,Y)≡ provide(Y:output.label,X:input.raw_data) ←map(X,Y)`
B8	`<model→user,`**`req-modified-sample(X,M)`**`, [X:selSegment;M:modifySegment]>`	`req-modified-sample(X,M)≡ request(M:input.raw_data,X:input.raw_data) ←modify(X,M)`
B9	`<model→user,`**`modify-sample(X,M)`**`, [X:selSegment;M:modifySegment]>`	`modify-sample(X,M)≡ provide(M:input.raw_data,X:input.raw_data) ←modify(X,M)`

Table 9. Interaction Patterns for the Interactive Sound Annotation System

Pattern	Actions/Messages	Description
new_class_sample	`req-new-class_sample(S,L)`	model asks user for a (positive) class sample
	`generate-class_sample(S,L)`	user provides a (positive) class sample
candidate_samples	`req-candidate_samples(CS,S)`	user asks for candidate (similar) samples
	`show-candidate_samples(CS,S)`	model provides a set of candidate samples
	`select-sample(X,CS)`	user selects a candidate sample
sample-annotation	`req-sample_class(X,Y)`	model asks user to provide a label for the input
	`annotate-sample(X,Y)`	user provides the correct label for the input
sample-modification	`req-modified_sample(X,M)`	model asks user to modify the sample (if needed)
	`modify-sample(X,M)`	user modifies the selected sample

interactive manner in order to annotate all candidate samples. More specifically, the first pattern is required to set the target class which represents samples which include a target sound. This is achieved by asking the user to provide a segment which includes the required sound (positive sample). During the second pattern, the model uses the positive sample to identify and visualize a list of candidate inputs for the user to label. The third pattern describes how user provides feedback to the model by listening and annotating the selected segment. The fourth pattern describes the modification of a sample (input) by adjusting the boundaries of the selected segment.

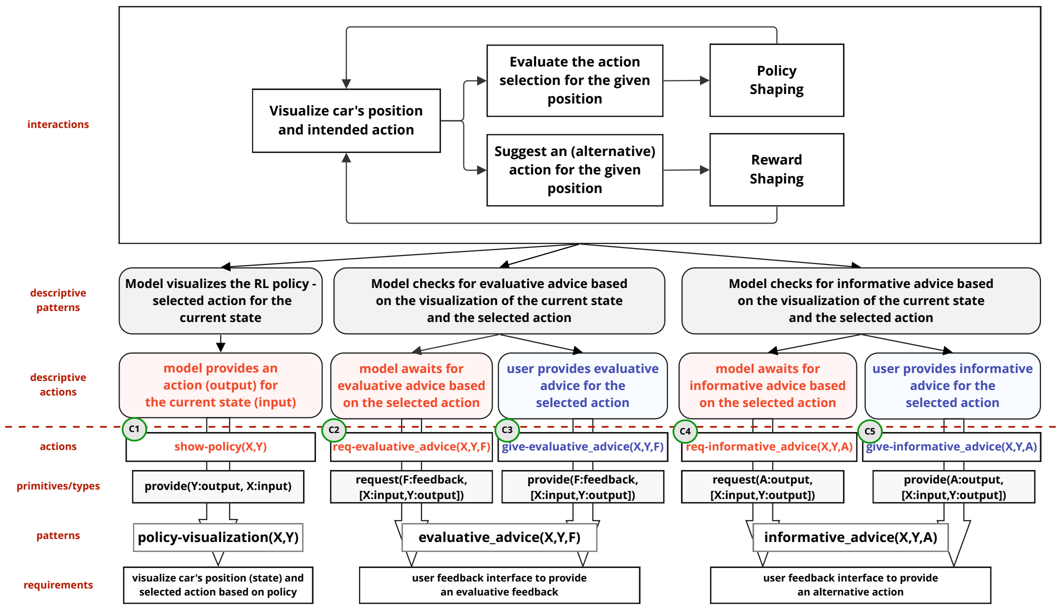

interactions
Visualize car's position and intended action
Evaluate the action selection for the given position
Suggest an (alternative) action for the given position
Policy Shaping
Reward Shaping
descriptive patterns
Model visualizes the RL policy - selected action for the current state
Model checks for evaluative advice based on the visualization of the current state and the selected action
Model checks for informative advice based on the visualization of the current state and the selected action
descriptive actions
model provides an action (output) for the current state (input)
model awaits for evaluative advice based on the selected action
user provides evaluative advice for the selected action
model awaits for informative advice based on the selected action
user provides informative advice for the selected action
actions
C1
show-policy(X,Y)
C2
req-evaluative_advice(X,Y,F)
C3
give-evaluative_advice(X,Y,F)
C4
req-informative_advice(X,Y,A)
C5
give-informative_advice(X,Y,A)
primitives/types
provide(Y:output, X:input)
request(F:feedback, [X:input,Y:output])
provide(F:feedback, [X:input,Y:output])
request(A:output, [X:input,Y:output])
provide(A:output, [X:input,Y:output])
patterns
policy-visualization(X,Y)
evaluative_advice(X,Y,F)
informative_advice(X,Y,A)
requirements
visualize car's position (state) and selected action based on policy
user feedback interface to provide an evaluative feedback
user feedback interface to provide an alternative action

Fig. 10. Unpacking an interactive RL interaction into interaction patterns. Image adapted from [15].

B.2 Interactive RL and Human Trainer Engagement (Figure 10)

The proposed system integrates human-provided feedback to an RL agent to improve its performance while executing the Mountain Car task. More specifically, the RL agent visualizes the current state and the selected action based on the model's policy (input–output pair) and allows the user to provide two types of feedback (evaluative/informative) in order to complete the task. Evaluative advice assesses the past performance of an agent and it is provided in the form of a reward, while informative advice supplements future decision-making and it is provided as an intervention—modified action. The goal of the interaction is to utilize human advice and maximize model's performance.

We identify the following interactions: (a) *policy visualization:* model visualizes its policy as a selected action for the current state (input–output pair), (b) *informative advice:* model queries the user for informative advice—user provides an alternate action (output) for the current state-action pair, and (c) *evaluative advice:* model queries the user for evaluative advice—user provides a reward (feedback) for the current state-action pair. Interactions take place in the form of a transparent interactive RL policy (visualization) and human-feedback for advice or evaluation (through keyboard input). The goal of the user study is to measure human engagement and model performance for the two types of feedback. In terms of the interaction design, human users are autonomous in terms of when they can provide feedback. According to the type of feedback, the model integrates it to the learning mechanism in different ways. If the user provides informative advice (learning from guidance), feedback is integrated to the learning mechanism through policy shaping, while for evaluative advice (learning from feedback), the RL agent uses reward shaping. The system uses an interface to visualize the RL policy and the task execution. Both types of advice are provided through a keyboard input, specific for each type. The design of the interactions plays an essential role in both model performance and human engagement. An important aspect to consider is the human feedback quality and consistency. The design challenge is to maintain user's engagement and performance considering user's perception. According to the analysis, users who provided

Table 10. Messages and Action Definitions for the Interactive RL Interactions

Message		Action definition
C1	<model⟶user,**show-policy(X,Y)**, [X:CarPosition;Y:selectedAction]>	show-policy(X,Y)≡ provide([X:input.state,Y:output.action])) ⟵ select(Y),map(X,Y)
C2	<model⟶user,**req-informative_advice(X,Y,A)**, [X:CarPosition;Y:selectedAction;A:keyboardAction]>	req-informative_advice(X,Y,A)≡ request(A:output.action,[X:input.state,Y:output.action]) ⟵modify(Y,A), map(X,A)
C3	<user⟶model,**give-informative_advice(X,Y,A)**, [X:CarPosition;Y:selectedAction;A:keyboardAction]>	give-evaluative_advice(X,Y,A)≡ provide(A:output.action,[X:input.state,Y:output.action]) ⟵modify(Y,A), map(X,A)
C4	<model⟶user,**req-evaluative_advice(X,Y,F)**, [X:CarPosition;Y:selectedAction;F:keyboardFeedback]>	req-evaluative_advice(X,Y,F)≡ request(F:feedback.eval,[X:input.state,Y:output.action]) ⟵select(F), map(X,A)
C5	<user⟶model,**give-evaluative_advice(X,Y,A)**, [X:CarPosition;Y:selectedAction;A:keyboardAction]>	give-evaluative_advice(X,Y,A)≡ provide(F:feedback.eval,[X:input.state,Y:output.action]) ⟵select(F),map(X,A)

Table 11. Interaction Patterns and Actions for Interactive RL

Pattern	Actions/Messages	Description
policy-visualization	`show-policy(X,Y)`	model visualizes action for current state
informative_advice	`req-informative_advice(X,Y,A)`	model checks if user provided advice
	`give-informative_advice(X,Y,A)`	user provides informative advice (action)
evaluative_advice	`req-evaluative_advice(X,Y,F)`	model checks if user provided feedback
	`give-evaluative_advice(X,Y,F)`	user provides evaluative feedback (reward)

informative advice were more engaged and accurate, which may be linked to how users perceive the different advice methods. Table 10 shows the messages and action definitions.

Based on these definitions, we define the following patterns (Table 11): `[C1] policy-visualization`, `[C2-C3] informative_advice`, and `evaluative_advice`. The interaction starts with the policy visualization and a human teaching method. For both patterns, the RL agent communicates its policy by visualizing the current state and the selected action. Based on the teaching method, there are two different types of interactions between the human trainer and the model: informative advice in the form of an alternate action/guidance and evaluative advice as a feedback/reward. Each patterns requires an appropriate model update mechanism for online learning. For the evaluation pattern, the user can evaluate the policy by providing evaluative feedback (reward shaping), while for the informative pattern, the user can suggest an action as informative advice (policy shaping).

B.3 Explainable AL for Collaborative Emotion Labeling (Figure 11)

This use case is based on an application of a multimodal annotation tool, called NOVA. The use case describes a collaborative annotation task for emotion recognition. The proposed tool includes XAI functionalities (transparency and visualizations) which aim to enhance user's decision making and trust to the system. More specifically, the proposed annotation system follows an AL approach to select which samples should be labeled by the user by visualizing the model's predictions confidence for these samples. The user can choose any sample and re-label it. Moreover, an XAI method (**Local Interpretable Model-agnostic Explanations (LIME)**) is used to support user's decision making (emotion detection) through saliency maps; visualizations of important visual features for the selected frame classification.

We identify the following interaction patterns: (a) model provides input–output pairs with confidence values and visualizations, and (b) user selects and visualizes an input–output pair and updates it. These patterns can be further described as (a1) model provides input, (a2) model provides

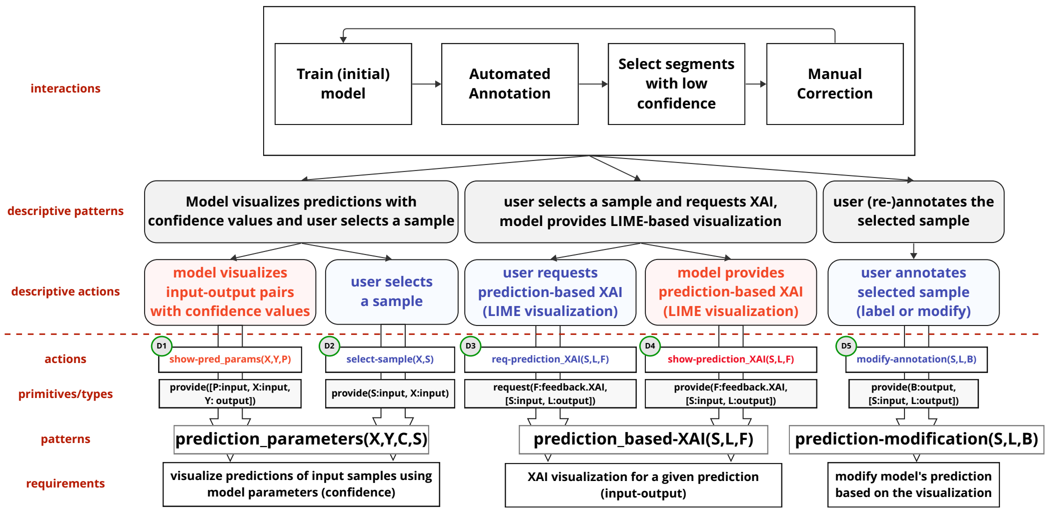

interactions
Train (initial) model
Automated Annotation
Select segments with low confidence
Manual Correction
descriptive patterns
Model visualizes predictions with confidence values and user selects a sample
user selects a sample and requests XAI, model provides LIME-based visualization
user (re-)annotates the selected sample
descriptive actions
model visualizes input-output pairs with confidence values
user selects a sample
user requests prediction-based XAI (LIME visualization)
model provides prediction-based XAI (LIME visualization)
user annotates selected sample (label or modify)
actions
D1 show-pred_params(X,Y,P)
D2 select-sample(X,S)
D3 req-prediction_XAI(S,L,F)
D4 show-prediction_XAI(S,L,F)
D5 modify-annotation(S,L,B)
primitives/types
provide([P:input, X:input, Y: output])
provide(S:input, X:input)
request(F:feedback.XAI, [S:input, L:output])
provide(F:feedback.XAI, [S:input, L:output])
provide(B:output, [S:input, L:output])
patterns
prediction_parameters(X,Y,C,S)
prediction_based-XAI(S,L,F)
prediction-modification(S,L,B)
requirements
visualize predictions of input samples using model parameters (confidence)
XAI visualization for a given prediction (input-output)
modify model's prediction based on the visualization

Fig. 11. Unpacking an explainable AL interaction into patterns. Image adapted from [44].

Table 12. Messages and Action Definitions for the AL Emotion Recognition

	Message	Action definition
D1	`<model⟶user,`**`show-prediction_params(X,Y,P)`**`, [X:frames;Y:emotionLabels;P:confValues]>`	`show-prediction_params(X,Y,P)≡ provide([X:[input.raw_data],[Y:output.label],[P:input.model_params]]))⟵ map(X,Y,P)`
D2	`<user⟶model,`**`select-sample(S,X)`**`, [X:frames;S:selFrame]>`	`select-sample(S,X)≡ provide(S:input.raw_data,X:[input.raw_data]) ⟵select(S,X)`
D3	`<user⟶model,`**`modify-prediction(S,L,A)`**`, [S:selFrame;L:emotionLabel;A:newLabel]>`	`modify-prediction(S,L,A)≡ provide(A:output.label,[S:input.raw_data,L:output.label]) ⟵modify(L,A), map(X,A)`
D4	`<model⟶user,`**`req-prediction_XAI(S,L,F)`**`, [S:selFrame;L:emotionLabel;F:LIMEVisualization]>`	`req-prediction_XAI(S,L,F)≡ request(F:feedback.XAI,[S:input.raw_data,L:output.label]) ⟵map(F,S,L)`
D5	`<model⟶user,`**`show-prediction_XAI(S,L,F)`**`, [S:selFrame;L:emotionLabel;F:LIMEVisualization]>`	`show-prediction_XAI(S,L,F)≡ provide(F:feedback.XAI,[S:input.raw_data,L:output.label]) ⟵map(F,S,L)`

output, and (a3) model provides XAI-based feedback (confidence-based visualization), and (b1) user provides input (selection), (b2) user provides output (selection), and (b3) user provides output (edit). The proposed system follows an explainable, semi-supervised AL approach. One of the main challenges of AL is to identify the appropriate queries (data points) to ask for user labeling. The proposed approach aims to improve model performance through interactive labeling. Considering both design and implementation aspects, the system utilizes a UI for model transparency and visual explanations, and integrates the HITL to facilitate the AL process, by identifying which data should be (re-)labeled. Both design and implementation choices are made to satisfy such requirements. Transparency and explainability are used to support user's decision making to refine a model through design features. The model guides the user to improve its performance for low-confidence predictions (AL), while additional explanations (LIME) can be requested to further support user's annotation task. Based on these, we define the following actions (Table 12):

We define the following patterns (Table 13): `[D1-D2] prediction_parameters`, `[D3] prediction-modification`, and `[D4-D5] prediction-based_XAI`. The first pattern describes the sample selection process, where the model supports the user to select the appropriate samples to

Table 13. Interaction Patterns and Actions for the AL Emotion Recognition

Pattern	Actions/Messages	Description
prediction_parameters	`show-prediction_params(X,Y,P)`	parameter-based sample visualization
	`select-sample(S,X)`	user selects a sample from list
prediction-modification	`modify-prediction(S,L,A)`	user annotates/modifies an annotation
prediction-based_XAI	`req-prediction_XAI(S,L,F)`	user request XAI for selected sample
	`show-prediction_XAI(S,L,F)`	model provides sample-based XAI

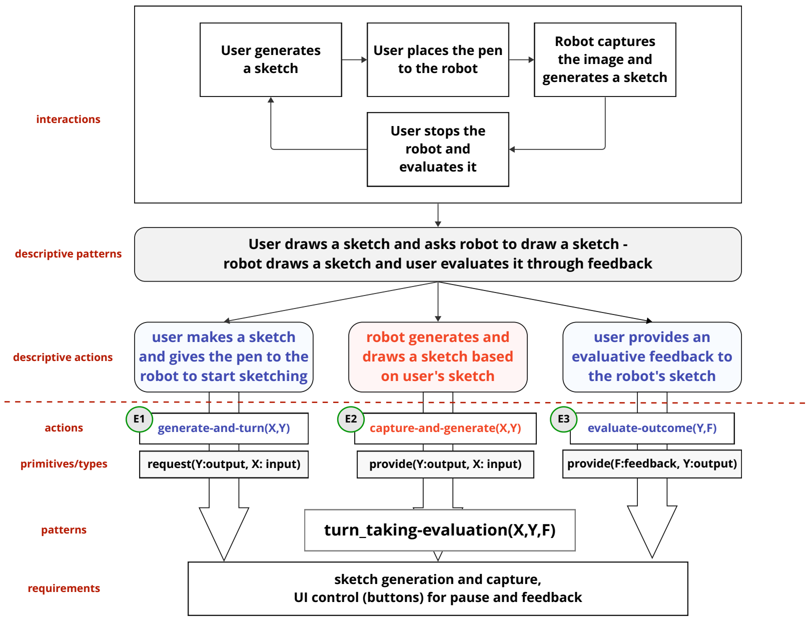

interactions
User generates a sketch
User places the pen to the robot
Robot captures the image and generates a sketch
User stops the robot and evaluates it
descriptive patterns
User draws a sketch and asks robot to draw a sketch - robot draws a sketch and user evaluates it through feedback
descriptive actions
user makes a sketch and gives the pen to the robot to start sketching
robot generates and draws a sketch based on user's sketch
user provides an evaluative feedback to the robot's sketch
actions
E1
generate-and-turn(X,Y)
E2
capture-and-generate(X,Y)
E3
evaluate-outcome(Y,F)
primitives/types
request(Y:output, X: input)
provide(Y:output, X: input)
provide(F:feedback, Y:output)
patterns
turn_taking-evaluation(X,Y,F)
requirements
sketch generation and capture, UI control (buttons) for pause and feedback

Fig. 12. Unpacking a human–robot collaborative sketching interaction into patterns. Image adapted from [56].

annotate through visualizing the model confidence for its predictions. This approach is part of the AL process based on which a set of candidate samples is selected for annotation. In this case, model instances with low confidence are presented to the user. The second pattern describes human labeling through a prediction modification approach. The third pattern describes a user request for XAI of a selected sample, where the model provides an LIME-based visualization for the user to understand the current prediction and modify it if needed. Local explanations are provided to the user to support their decision making through local interpretability. The output of LIME is a visualization of explanations representing the contribution of each feature to the prediction of the current frame. These patterns (and their actions) can be combined to design the interactions, e.g., the user can explore and select a sample based on the visualizations and either annotate it or request sample-based explanations.

B.4 Human–Robot Collaborative Sketching (Figure 12)

The proposed system describes a collaborative sketching approach between a user and a robot (Cobbie). More specifically, the proposed system utilizes the mechanism of conceptual shift to

Table 14. Messages and Action Definitions for the Collaborative Sketching Interactions

Message		Action definition
E1	`<model⟶user,`**`generate-and-turn(X,Y)`**`, [X:userSketch;Y:onPenClipper,robotSketch]>`	`generate-and-turn(X,Y)≡ request(Y:input.raw_data,C:output.raw_data)) ⟵ create(X), create(Y), map(X,Y)`
E2	`<user⟶model,`**`capture-and-generate(X,Y)`**`, [X:userSketch; Y:robotSketch]>`	`capture-and-generate(X,Y)≡ provide(Y:input.raw_data,C:output.raw_data)) ⟵create(Y),map(X,Y)`
E3	`<user⟶model,`**`evaluate-outcome(Y,F)`**`, [Y:robotSketch;F:pauseBtn,feedbackBtn]>`	`evaluate-outcome(Y,F)≡ provide(F:feedback.eval,Y:output.raw_data) ⟵select(F), map(Y,F)`

Table 15. Interaction Patterns and Actions for the Collaborative Robot Sketching

Pattern	Actions	Description
turn_taking-evaluation	`generate-and-turn(X,Y)`	user asks robot to sketch based on the drawing
	`capture-and-generate(X,Y)`	robot captures and generates new sketch
	`evaluate-outcome(Y,F)`	user pauses robot and provides feedback

support HAI co-creation and collaborative sketch ideation. Based on this approach, the user initiates the interaction by sketching an image on article. Once finished, the user gives the pen to the robot, which captures and analyzes the user's sketch. Based on its analysis, it generates a new sketch on article. The user can pause the robot and provide an evaluative feedback for the robot's drawing. If feedback is negative, the robot starts drawing a new sketch until new feedback is received. If feedback is positive, the user draws a new sketch by combining the two sketches. The robot utilizes the provided feedback to adjust its model in order to provide more useful ideas to the user.

The interaction takes place as a turn-taking sketching-based interaction, where user and model sketch a drawing considering previous drawing (co-ideation). In terms of the types of communicated information, user and robot communicate through drawing sketches, giving the pen to the robot and providing feedback to the robot through buttons. In terms of implementation aspects, the robot deploys a **Recurrent Neural Network (RNN)** recognizer to capture and classify the user's input and an adapted version of the RNN-sketch model to generate a new image based on user's input. User's feedback is used to update the network weights, and thus in terms of interaction design, the user is the dominant member of the co-creation session. The user can determine when Cobbie should start drawing an image, by placing the pen to the robot's clipper. The user can also select the robot's position (what to capture and where to draw) as well as the pen strokes. Users can use the on-platform buttons to pause and resume robot's drawing and provide feedback. Robot expressive movements and sounds indicate that the robot has successfully received a command (button pressed). In terms or model performance, the deployed AI models (image classification and RNN-sketch) are well-performing models. The goal of the interaction is to support user's creative thinking and ideation processes. The model provides appropriate outputs (not the most accurate) in order to facilitate this co-ideation process. Table 14 shows the message and action definitions for this interaction.

Based on this interaction, we can define `[E]:turn_taking-evaluation` as a user-driven control and evaluation pattern, where the user can control and evaluate the collaboration with the model (Table 15). Based on this pattern, the user initiates the interaction by drawing a sketch and giving the pen to the robot for idea generation. The goal of the robot is to support user's ideation process by generating creative and diverse sketches. This is achieved through object detection (image classification) and conceptual shift (sketch-RNN). The user evaluates the robot's sketches (model

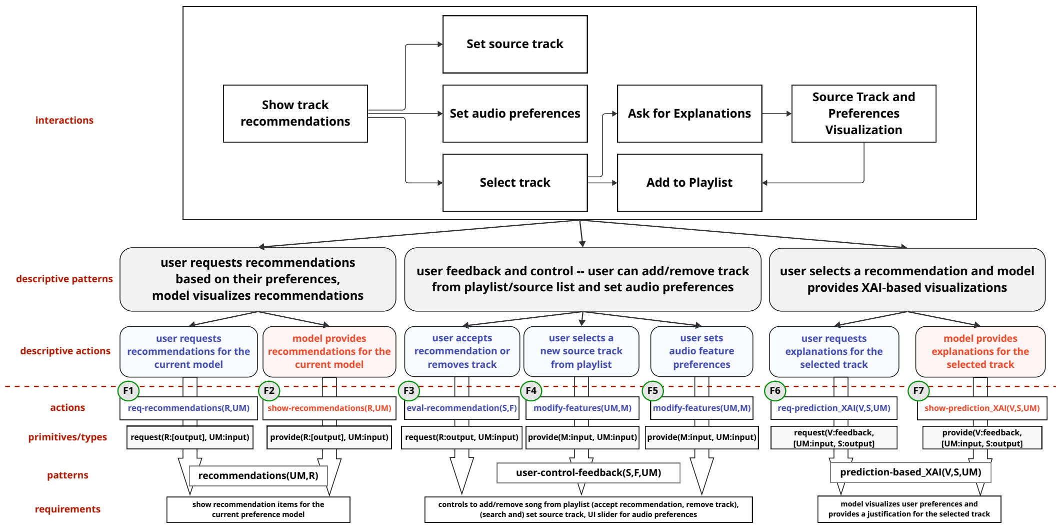

interactions
Set source track
Show track recommendations
Set audio preferences
Ask for Explanations
Source Track and Preferences Visualization
Select track
Add to Playlist
descriptive patterns
user requests recommendations based on their preferences, model visualizes recommendations
user feedback and control -- user can add/remove track from playlist/source list and set audio preferences
user selects a recommendation and model provides XAI-based visualizations
descriptive actions
user requests recommendations for the current model
model provides recommendations for the current model
user accepts recommendation or removes track
user selects a new source track from playlist
user sets audio feature preferences
user requests explanations for the selected track
model provides explanations for the selected track
F1
F2
F3
F4
F5
F6
F7
actions
req-recommendations(R,UM)
show-recommendations(R,UM)
eval-recommendation(S,F)
modify-features(UM,M)
modify-features(UM,M)
req-prediction_XAI(V,S,UM)
show-prediction_XAI(V,S,UM)
primitives/types
request(R:[output], UM:input)
provide(R:[output], UM:input)
request(R:output, UM:input)
provide(M:input, UM:input)
provide(M:input, UM:input)
request(V:feedback, [UM:input, S:output]
provide(V:feedback, [UM:input, S:output]
patterns
recommendations(UM,R)
user-control-feedback(S,F,UM)
prediction-based_XAI(V,S,UM)
requirements
show recommendation items for the current preference model
controls to add/remove song from playlist (accept recommendation, remove track), (search and) set source track, UI slider for audio preferences
model visualizes user preferences and provides a justification for the selected track

Fig. 13. Unpacking an explainable music recommendation system into patterns. Image adapted from [60].

output) using the feedback buttons. This interaction pattern can describe an explorative collaborative learning process, where both user and system aim to explore and identify new insights through their interaction.

B.5 Explainable Music Recommendation System (Figure 13)

The proposed system aims to explore the design of explanations in a music recommender system in order to fit the user's preferences (selected songs, audio preferences) and personal characteristics (i.e., need for cognition, musical sophistication, and openness). The system provides recommendations to the user based on a source song (i.e., a playlist song) and user preferences for audio features (danceability, energy, happiness, and popularity). User can search and add recommended tracks to the playlist, remove existing track from the playlist, select a playlist song as a source song for recommendations, set their preferences through the audio features, and request and explore explanations. Explanations could be requested (and provided) both for a selected recommendation as well as for all songs at once.

We identify the following types of interaction: (a) model visualizes playlist and recommendations with control options—user adds (or removes) a song from the playlist/source list songs, as well as through the audio feature preferences, and (b) user can request further explanations for a given track or all songs—model provides visual and textual explanations to justify recommendations. In terms of the communicated information, the model and the user interact through showing and selecting (explainable) recommendations and audio feature preferences. In terms of implementation, the model utilizes user's actions (adding/removing songs, setting preferences, asking for XAI), in order to provide personalized recommendations and explanations. After certain interactions with the user, i.e., add/remove/search song or change audio preferences, the model updates its recommendations (updated feature vector). In terms of design aspects, the model provides different types of explanations in order to match the individual's preference and characteristics. Based on the outcomes from a set of user studies, the authors provide a set of design suggestions towards selecting appropriate explanation styles and levels of transparency based on user's personal characteristics. For example, users with low musical sophistication may prefer brief explanations that do not require domain knowledge. we can define the following actions (Table 16):

Table 16. Messages and Action Definitions for the Interactive Sound Annotation System Interactions

	Message	Action definition
F1	`<model⟶user,`**`req-recommendations(R,UM)`**`, [[UM:audioPrefs,srcTrack;R:reccTracks]>`	`req-recommendations(R,UM)≡ request(R:[output.item],UM:input.fvector)) ⟵ select(R),map(UM,R)`
F2	`<model⟶user,`**`show-recommendations(R,UM)`**`, [[UM:audioPrefs,srcTrack;R:reccTracks]>`	`show-recommendations(R,UM)≡ provide(R:[output.item],UM:input.fvector)) ⟵ select(R),map(UM,R)`
F3	`<user⟶model,`**`modify-features(UM,M)`**`, [UM:audioPrefs;M:modifyPrefs,UIslider]>`	`modify-features(UM,M)≡ provide(M:input.fvector,UM:input.fvector) ⟵modify(UM,M)`
F4	`<model⟶user,`**`evaluate-recommendation(F,S)`**`, [F:addPlaylistTrack;S:selTrack]>`	`evaluate-recommendation(F,R)≡ provide(F:feedback.eval,S:output.item) ⟵select(S),map(S,F)`
F5	`<user⟶model,`**`modify-features(UM,M)`**`, [UM:srcTrack;M:newSrcTrack,click]>`	`modify-features(UM,M)≡ provide(M:input.fvector,UM:input.fvector) ⟵modify(UM,M)`
F6	`<model⟶user,`**`req-prediction_XAI(V,S,UM)`**`, [V:clickXAIBtn;S:selTrack;UM:audioPrefs,srcTrack]>`	`req-prediction_XAI(UM,S,V) ≡ request(V:feedback.XAI;[UM:input.fvector, S:output.item]) ⟵ select(S), map(V,S,UM)`
F7	`<model⟶user,`**`show-prediction_XAI(V,S,UM)`**`, [V:openUserModel;S:selTrack;UM:audioPrefs,srcTrack]>`	`show-prediction_XAI(UM,S,V) ≡ provide(V:feedback.XAI;[UM:input.fvector, S:output.item]) ⟵ select(S), map(V,S,UM)`

Table 17. Interaction Patterns for the Explainable Music Recommendation System

Pattern	Actions	Description
recommendations	`req-recommendations(UM,TR)`	user requests recommendations for preferences
	`show-recommendations(UM,TR)`	model visualizes recommendations for preferences
user-control-feedback	`modify-features(UM,M)`	user sets preferences and updates feature vector
	`evaluate-recommendation(S,PL)`	user adds or removes playlist song
prediction-based_XAI	`req-prediction_XAI(S,V,UM)`	user requests XAI for recommendations
	`show-prediction_XAI(S,V,UM)`	model visualizes user model

We define the following patterns (Table 17): `[F1] recommendations`, `[F2-F3] user-control-feedback`, and `[F4-F5] prediction-based_XAI`. Based on the first pattern, the user is provided with a list of recommended tracks for the current preferences. The second pattern described the user's interactions with the recommendations and their preferences. The model updates its decisions based on these user actions. The third pattern is part of an XAI-interaction where the system visualizes the preference model and the similarity to justify a given recommendation. The user makes the final decision about a recommended track (output) and the feature values (input).

B.6 Transparent Meeting Scheduling Assistant (Figure 14)

The proposed system uses an AI model to automatically detect meeting requests from free-text emails. The scheduling assistant provides the user with additional information (XAI), including an accuracy indicator component and a textual description for example-based explanations. The accuracy indicator visualizes the model's accuracy for the predictions in the form of a chart. The example-based explanations aim to enhance user's understanding about the underlying AI model and include a set of example sentences (inputs) and the model prediction (output) for each sentence, which can vary from "very unlikely" to "very likely" to describe the model's confidence, and depend on the model's sensitivity. The user is able to control model's sensitivity and see the updated results, through a UI slider which provides information about how sensitivity affects model's decisions.

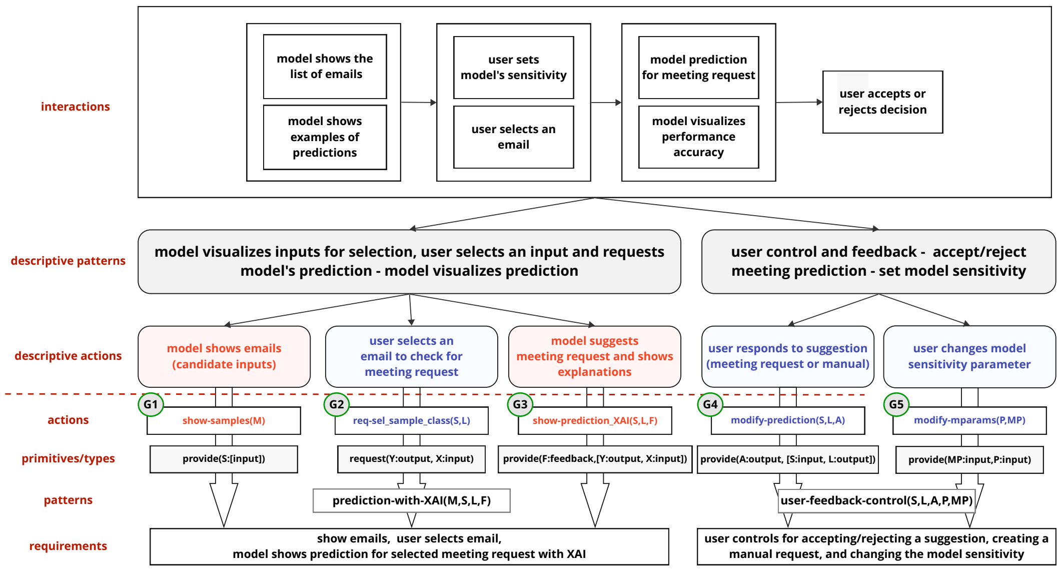

interactions
model shows the list of emails
model shows examples of predictions
user sets model's sensitivity
user selects an email
model prediction for meeting request
model visualizes performance accuracy
user accepts or rejects decision
descriptive patterns
model visualizes inputs for selection, user selects an input and requests model's prediction - model visualizes prediction
user control and feedback - accept/reject meeting prediction - set model sensitivity
descriptive actions
model shows emails (candidate inputs)
user selects an email to check for meeting request
model suggests meeting request and shows explanations
user responds to suggestion (meeting request or manual)
user changes model sensitivity parameter
actions
G1
show-samples(M)
G2
req-sel_sample_class(S,L)
G3
show-prediction_XAI(S,L,F)
G4
modify-prediction(S,L,A)
G5
modify-mparams(P,MP)
primitives/types
provide(S:[input])
request(Y:output, X:input)
provide(F:feedback,[Y:output, X:input])
provide(A:output, [S:input, L:output])
provide(MP:input,P:input)
patterns
prediction-with-XAI(M,S,L,F)
user-feedback-control(S,L,A,P,MP)
requirements
show emails, user selects email, model shows prediction for selected meeting request with XAI
user controls for accepting/rejecting a suggestion, creating a manual request, and changing the model sensitivity

Fig. 14. Unpacking the Meeting Scheduling Assistant [51] into patterns and primitives.

Table 18. Messages and Action Definitions for the Meeting Scheduling Assistant

	Message	Action definition
G1	`<model⟶user,`**`show-samples(M)`**`,` `[M:emailList,freeText]>`	`show-samples(M)≡` `provide(M:[input.raw_data])` `⟵ select(M)`
G2	`<user⟶model,` **`req-sel_sample_class(S,L)`**`,` `[S:selectedEmail;L:L:isMeeting]>`	`req-sel_sample_class(S,L)≡` `request(L:output.label,S:input.raw_data)` `⟵ select(S),map(S,L)`
G3	`<model⟶user,` **`show-prediction-XAI(S,L,F)`**`,` `[S:selectedEmail;L:L:isMeeting;F:XAIexamples]>`	`show-prediction-XAI(S,L,F)≡` `provide(F:feedback.XAI;[S:input.raw_data,L:output.label])` `⟵ map(S,L), map(F,S,L)`
G4	`<model⟶user,`**`modify-prediction(S,L,A)`**`,` `[S:selectedEmail; L:isMeeting;A:acceptBtn,createBtn]>`	`modify-prediction(S,L,A)≡` `provide(A:output.label,[S:input.raw_data,` `L:output.label])⟵ select(S),modify(L,A),map(S,A)`
G5	`<model⟶user,`**`modify-mparams(P,MP)`**`,` `[P:sensitivityValue; MP:modifiedValue,UIslider]>`	`modify-mparams(P,MP)≡` `provide(MP:input.model_params,P:model_params))` `⟵ modify(P,MP)`

We can identify the following interactions: (a) model visualizes the inputs (e-mails) and user selects an item to check model's prediction—model provides prediction with XAI, (b) user provides feedback to the model through accepting or rejecting suggestions (predictions) and/or setting the sensitivity value through the slider. During these interactions, user and model communicate messages through visualization and selection of model inputs, textual and visual explanations and control interfaces (slider), as well as accepting or rejecting model's decisions. IN order to enhance user's decision making, model provides explanations and transparency in terms of performance. Interactions with low-performance models (low confidence) can be effective for the user, if they provide an appropriate level of transparency and explainability. Such interaction can enhance user's decision making. In terms of implementation aspects, the different types of user feedback (e.g., accepting/rejecting suggestions) can be used to update the model in an interactive way. Users can set the model sensitivity parameter value based on their observations of how it affects model's decisions. we define the following actions (Table 18):

Table 19. Interaction Patterns for the Transparent Meeting Scheduling Assistant

Pattern	Actions	Description
prediction-with-XAI	`show-samples(M)`	model visualizes all inputs (emails)
	`req-sel_sample_class(S,L)`	user selects an email and asks for suggestion
	`show-prediction_XAI(S,L,F)`	model visualizes prediction with XAI
user-feedback-control	`modify-prediction(S,L,A)`	user accepts or modifies prediction
	`modify-mparams(P,MP)`	user sets (a new) model sensitivity parameter

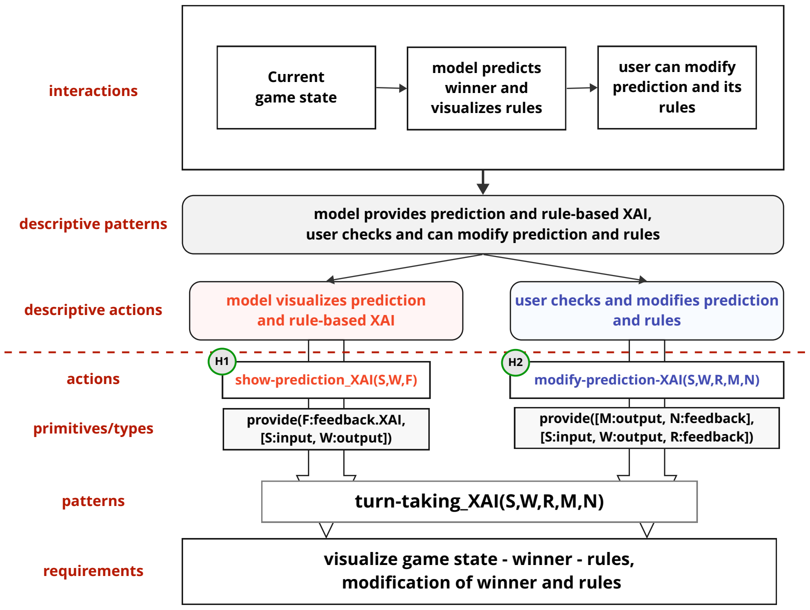

interactions
Current game state
model predicts winner and visualizes rules
user can modify prediction and its rules
descriptive patterns
model provides prediction and rule-based XAI, user checks and can modify prediction and rules
descriptive actions
model visualizes prediction and rule-based XAI
user checks and modifies prediction and rules
actions
H1
show-prediction_XAI(S,W,F)
H2
modify-prediction-XAI(S,W,R,M,N)
primitives/types
provide(F:feedback.XAI, [S:input, W:output])
provide([M:output, N:feedback], [S:input, W:output, R:feedback])
patterns
turn-taking_XAI(S,W,R,M,N)
requirements
visualize game state - winner - rules, modification of winner and rules

Fig. 15. Unpacking an iML system for game rules into patterns and primitives. Image adapted from [42].

We identify two patterns: `[G1-G3] prediction-with-XAI`, and `[G4-G5] user-feedback-control` (Table 18). The first pattern describes an XAI-based interaction for input selection and prediction, where the model provides textual explanations of prediction examples to the user to enhance their understanding about the model's prediction. The second pattern describes a feedback control pattern; the user provides feedback to set the model sensitivity and interacts with the model decisions. Depending on the prediction, the user can agree with the model and accept a predicted request (true positive) or a predicted non-request (true negative). If the model does not predict a meeting request (false negative) the user can manually create a meeting request. If the model predicts a false meeting request (false positive), the user can ignore the predicted request.

B.7 Explainable-Driven iML for Game Outcome Prediction (Figure 15)

The proposed system integrates an explanation-driven iML mechanism to improve user's trust and satisfaction during the interaction with the system. The use case is the Tic-Tac-Toe game, where user and model make predictions about the winner of the game for a given state (game instance), in a turn-taking interaction. Both user and model can justify their predictions using rule-based explanations.

Table 20. Messages and Action Definitions for the XAI-Based iML for Game Rules

	Message	Action definition
G1	`<model⟶user, show-prediction_XAI(S,W,R), [S:gameState;W:predWinner;R:XAIrules]>`	`show-prediction-and-XAI(S,W,R)≡ provide(R:feedback.XAI;[S:input.raw_data,W:output.label]) ⟵ map(S,L), map(F,S,L)`
G2	`<model⟶user,modify-prediction-and-XAI(S,W,R,M,N), [S:gameState;W:predWinner;R:XAIrules; M:modifiedPred;N:modifiedRules]>`	`modify-prediction-and-XAI(S,W,R,M,N)≡ provide([M:output.label,N:feedback.XAI], [S:input.raw_data,W:output.label,R:feedback.XAI]) ⟵ modify(W,M),modify(R,N),map(S,M,N)`

Table 21. Interaction Patterns for the Explainable/Interactive Game Outcome Predictions

Pattern	Actions	Description
turn-taking_XAI	show-prediction_XAI(S,W,R)	model provides rule-based explanations for prediction
	modify_prediction_rules(S,W,MW,MR)	user accepts or modifies prediction and rules

During this turn-taking interaction, the model visualizes a game instance and its prediction for the winner. In order to justify its prediction, it visualizes the rule based on which the decision was made. The user can accept or modify both prediction and rules. The rules are a set of Boolean rules in disjunctive normal form. The authors conducted a user study to evaluate the effects of interactivity and visualization on user's trust and satisfaction. The outcomes of their analysis indicate that both aspects can have an effect on users' perception of control over the different types of visualization. The model visualizes its prediction and reasoning to enhance user's trust in model performance and does not update its prediction based on user 's input. However, similar interactions can take place in HI systems, where both users and models support their own decisions/predictions in order to augment each other's perception. Table 20 shows the action definitions of the pattern (Table 21). `turn-taking_XAI` a turn-taking pattern where both user and model can exchange the same information in an interactive manner.

References

[1] Ajaya Adhikari, Edwin Wenink, Jasper van der Waa, Cornelis Bouter, Ioannis Tolios, and Stephan Raaijmakers. 2022. Towards FAIR Explainable AI: A standardized ontology for mapping XAI solutions to use cases, explanations, and AI systems. In *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*. 562–568.

[2] Raheel Ahmad, Shahram Rahimi, and Bidyut Gupta. 2007. An intelligence-aware process calculus for multi-agent system modeling. In *Proceedings of the 2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems*. IEEE, 210–215.

[3] Moamin Ahmed, Mohd S. Ahmad, and Mohd Z. M. Yusoff. 2009. A review and development of agent communication language. *Electronic Journal of Computer Science and Information Technology* 1, 1 (2009), 7-12.

[4] Zeynep Akata, Dan Balliet, Maarten De Rijke, Frank Dignum, Virginia Dignum, Guszti Eiben, Antske Fokkens, Davide Grossi, Koen Hindriks, Holger Hoos, Hayley Hung, Catholijn Jonker, Christof Monz, Mark Neerincx, Frans Oliehoek, Henry Prakken, Stefan Schlobach, Linda van der Gaag, Frank van Harmelen, Herke van Hoof, Birna van Riemsdijk, Aimee van Wynsberghe, Rineke Verbrugge, Bart Verheij, Piek Vossen, and Max Welling. 2020. A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence. *Computer* 53, 08 (2020), 18–28.

[5] Pekka Ala-Pietilä, Yann Bonnet, Urs Bergmann, Maria Bielikova, Cecilia Bonefeld-Dahl, Wilhelm Bauer, Loubna Bouarfa, Raja Chatila, Mark Coeckelbergh, Virginia Dignum, Jean-Francois Gagné, Joanna Goodey, Sami Haddadin, Gry Hasselbalch, Fredrik Heintz, Fanny Hidvegi, Klaus Höckner, Mari-Noëlle Jégo-Laveissière, Leo Kärkkäinen, Sabine Theresia Köszegi, Robert Kroplewski, Ieva Martinkenaite, Raoul Mallart, Catelijne Muller, Cécile Wendling, Barry O'Sullivan, Ursula Pachl, Nicolas Petit, Andrea Renda, Francesca Rossi, Karen Yeung, Françoise Soulié Fogelman, Jaan Tallinn, Jakob Uszkoreit, and Aimee Van Wynsberghe. 2020. *The Assessment List for Trustworthy Artificial Intelligence (ALTAI)*. European Commission.

[6] Omer Faruk Alaca, Baris Tekin Tezel, Moharram Challenger, Miguel Goulão, Vasco Amaral, and Geylani Kardas. 2021. AgentDSM-eval: A framework for the evaluation of domain-specific modeling languages for multi-agent systems. *Computer Standards & Interfaces* 76 (2021), 103513.

[7] Kars Alfrink, Ianus Keller, Gerd Kortuem, and Neelke Doorn. 2022. Contestable AI by Design: Towards a Framework. Minds & Machines. 33, 4, 613-639. 10.1007/s11023-022-09611-z

[8] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[9] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115.

[10] John L. Austin. 1975. *How to Do Things with Words*. Vol. 88. Oxford University Press.

[11] Antonio Balderas, Juan Manuel Dodero, Manuel Palomo-Duarte, and Ivan Ruiz-Rube. 2015. A domain specific language for online learning competence assessments. *International Journal of Engineering Education* 31, 3 (2015), 851–862.

[12] Kyle J. Behymer and John M. Flach. 2016. From autonomous systems to sociotechnical systems: Designing effective collaborations. *She Ji: The Journal of Design, Economics, and Innovation* 2, 2 (2016), 105–114.

[13] Jürgen Bernard, Marco Hutter, Michael Sedlmair, Matthias Zeppelzauer, and Tamara Munzner. 2021. A taxonomy of property measures to unify active learning and human-centered approaches to data labeling. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 11, 3–4 (2021), 1–42.

[14] Umang Bhatt, Javier Antorán, Yunfeng Zhang, Q. Vera Liao, Prasanna Sattigeri, Riccardo Fogliato, Gabrielle Melançon, Ranganath Krishnan, Jason Stanley, Omesh Tickoo, Lama Nachman, Rumi Chunara, Madhulika Srikumar, Adrian Weller, and Alice Xiang. 2021. Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 401–413.

[15] Adam Bignold, Francisco Cruz, Richard Dazeley, Peter Vamplew, and Cameron Foale. 2023. Human engagement providing evaluative and informative advice for interactive reinforcement learning. Neural Comput. & Applic.. 35, 25 (Sep 2023), 18215-18230. 10.1007/s00521-021-06850-6

[16] Abeba Birhane. 2021. Algorithmic injustice: A relational ethics approach. *Patterns* 2, 2 (2021), 100205.

[17] Luciano Cavalcante Siebert, Maria Luce Lupetti, Evgeni Aizenberg, Niek Beckers, Arkady Zgonnikov, Herman Veluwenkamp, David Abbink, Elisa Giaccardi, Geert-Jan Houben, Catholijn M. Jonker, Jeroen van den Hoven, Deborah Forster, and Reginald L. Lagendijk. 2022. Meaningful human control: Actionable properties for AI system development. *AI and Ethics* 3, 1 (2022), 241–255.

[18] Chengliang Chai and Guoliang Li. 2020. Human-in-the-loop techniques in machine learning. *IEEE Data Engineering Bulletin* 43, 3 (2020), 37–52.

[19] Stevie Chancellor. 2023. Toward practices for human-centered machine learning. *Communications of the ACM* 66, 3 (2023), 78–85.

[20] Valerie Chen, Umang Bhatt, Hoda Heidari, Adrian Weller, and Ameet Talwalkar. 2023. Perspectives on incorporating expert feedback into model updates. *Patterns* 4, 7, DOI: https://doi.org/10.1016/j.patter.2023.100780

[21] Ruijia Cheng, Alison Smith-Renner, Ke Zhang, Joel R. Tetreault, and Alejandro Jaimes. 2022. Mapping the design space of human-AI interaction in text summarization. arXiv.2206.14863. Retrieved from https://doi.org/10.48550/arXiv.2206.14863

[22] Michael Chromik and Andreas Butz. 2021. Human-XAI interaction: A review and design principles for explanation user interfaces. In *Proceedings of the IFIP Conference on Human-Computer Interaction*. Springer, 619–640.

[23] Michael Chromik and Martin Schuessler. 2020. A taxonomy for human subject evaluation of black-box explanations in XAI. *Exss-atec@ iui* 94 (2020).

[24] Yuchen Cui, Pallavi Koppol, Henny Admoni, Scott Niekum, Reid G. Simmons, Aaron Steinfeld, and Tesca Fitzgerald. 2021. Understanding the relationship between interactions and outcomes in human-in-the-loop machine learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 4382–4391.

[25] Dominik Dellermann, Adrian Calma, Nikolaus Lipusch, Thorsten Weber, Sascha Weigel, and Philipp Ebel. 2021. The future of human-AI collaboration: A taxonomy of design knowledge for hybrid intelligence systems. arXiv.2105.03354. Retrieved from https://doi.org/10.48550/arXiv.2105.03354

[26] Shipi Dhanorkar, Christine T. Wolf, Kun Qian, Anbang Xu, Lucian Popa, and Yunyao Li. 2021. Who needs to know what, when? Broadening the explainable AI (XAI) design space by looking at explanations across the AI lifecycle. In *Proceedings of the 2021 Designing Interactive Systems Conference*. 1591–1602.

[27] Zijian Ding and Joel Chan. 2023. Mapping the design space of interactions in human-AI text co-creation tasks. arXiv.2303.06430. Retrieved from https://doi.org/10.48550/arXiv.2303.06430

[28] Mark d'Inverno, Michael Luck, Pablo Noriega, Juan A. Rodriguez-Aguilar, and Carles Sierra. 2012. Communicating open systems. *Artificial Intelligence* 186 (July 2012), 38–94. DOI: https://doi.org/10.1016/j.artint.2012.03.004

[29] Graham Dove and Anne-Laure Fayard. 2020. Monsters, Metaphors, and Machine Learning. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '20)*. ACM, New York, NY, 1–17. DOI: https://doi.org/10.1145/3313831.3376275

[30] Graham Dove, Kim Halskov, Jodi Forlizzi, and John Zimmerman. 2017. UX design innovation: Challenges for working with machine learning as a design material. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 278–288.

[31] John J. Dudley and Per Ola Kristensson. 2018. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems* 8, 2 (2018), 1–37.

[32] Therese Enarsson, Lena Enqvist, and Markus Naarttijärvi. 2022. Approaching the human in the loop–legal perspectives on hybrid human/algorithmic decision-making in three contexts. *Information & Communications Technology Law* 31, 1 (2022), 123–153.

[33] Alex Endert, Patrick Fiaux, and Chris North. 2012. Semantic interaction for sensemaking: Inferring analytical reasoning for model steering. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2879–2888.

[34] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. 1994. KQML as an agent communication language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*. 456–463.

[35] Jules Françoise, Baptiste Caramiaux, and Téo Sanchez. 2021. Marcelle: Composing interactive machine learning workflows and interfaces. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology*. 39–53.

[36] Elisa Giaccardi and Johan Redström. 2020. Technology and more-than-human design. *Design Issues* 36, 4 (2020), 33–44.

[37] Joan Giner-Miguelez, Abel Gómez, and Jordi Cabot. 2023. A domain-specific language for describing machine learning datasets. *Journal of Computer Languages* 76 (2023), 101209.

[38] Imke Grabe, Miguel González-Duque, Sebastian Risi, and Jichen Zhu. 2022. Towards a framework for human-AI interaction patterns in co-creative GAN applications. In *Proceedings of the 3rd Workshop on Human-AI Co-Creation with Generative Models (HAI-GEN '22) at ACM IUI Workshops,* 3124 (2022).

[39] Stefan Grafberger, Paul Groth, Julia Stoyanovich, and Sebastian Schelter. 2022. Data distribution debugging in machine learning pipelines. *The VLDB Journal* 31, 5 (2022), 1103–1126.

[40] Stefan Grafberger, Shubha Guha, Julia Stoyanovich, and Sebastian Schelter. 2021. Mlinspect: A data distribution debugger for machine learning pipelines. In *Proceedings of the 2021 International Conference on Management of Data*. 2736–2739.

[41] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea L. Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems,* Vol. 26.

[42] Lijie Guo, Elizabeth M Daly, Oznur Alkan, Massimiliano Mattetti, Owen Cornec, and Bart Knijnenburg. 2022. Building trust in interactive machine learning via user contributed interpretable rules. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*. 537–548.

[43] Andrea L. Guzman and Seth C. Lewis. 2020. Artificial intelligence and communication: A human–machine communication research agenda. *New Media & Society* 22, 1 (2020), 70–86.

[44] Alexander Heimerl, Tobias Baur, Florian Lingenfelser, Johannes Wagner, and Elisabeth André. 2019. NOVA-a tool for eXplainable cooperative machine learning. In *Proceedings of the 8th International Conference on Affective Computing and Intelligent Interaction (ACII '19).* IEEE, 109–115.

[45] Tad Hirsch, Kritzia Merced, Shrikanth Narayanan, Zac E. Imel, and David C. Atkins. 2017. Designing contestability: Interaction design, machine learning, and mental health. In *Proceedings of the 2017 Conference on Designing Interactive Systems*. 95–99.

[46] Lars E. Holmquist. 2017. Intelligence on tap: Artificial intelligence as a new design material. *Interactions* 24, 4 (2017), 28–33.

[47] Ethan K. Jackson and Janos Sztipanovits. 2006. Towards a formal foundation for domain specific modeling languages. In *Proceedings of the 6th ACM & IEEE International Conference on Embedded Software*. 53–62.

[48] Pranav Khadpe, Ranjay Krishna, Li Fei-Fei, Jeffrey T. Hancock, and Michael S. Bernstein. 2020. Conceptual metaphors impact perceptions of human-AI collaboration. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW2 (2020), 1–26.

[49] Bongjun Kim and Bryan Pardo. 2018. A human-in-the-loop system for sound event detection and annotation. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 8, 2 (2018), 1–23.

[50] Hankyung Kim and Youn-Kyung Lim. 2021. Teaching-learning interaction: A new concept for interaction design to support reflective user agency in intelligent systems. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference*. 1544–1553.

[51] Rafal Kocielnik, Saleema Amershi, and Paul N. Bennett. 2019. Will you accept an imperfect AI? Exploring designs for adjusting end-user expectations of AI systems. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–14.

[52] Andreas Kunft, Asterios Katsifodimos, Sebastian Schelter, Sebastian Breß, Tilmann Rabl, and Volker Markl. 2019. An intermediate representation for optimizing machine learning pipelines. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1553–1567.

[53] Michelle S. Lam, Zixian Ma, Anne Li, Izequiel Freitas, Dakuo Wang, James A. Landay, and Michael S. Bernstein. 2023. Model sketching: Centering concepts in early-stage machine learning model design. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–24.

[54] Q. Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: Informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.

[55] Q. Vera Liao and Kush R. Varshney. 2021. Human-centered explainable AI (XAI): From algorithms to user experiences. arXiv.2110.10790. Retrieved from https://doi.org/10.48550/arXiv.2110.10790

[56] Yuyu Lin, Jiahao Guo, Yang Chen, Cheng Yao, and Fangtian Ying. 2020. It is your turn: Collaborative ideation with a co-creative robot through sketch. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[57] Joseph Lindley, Haider Ali Akmal, Franziska Pilling, and Paul Coulton. 2020. Researching AI legibility through design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[58] Michal Luria. 2018. Designing robot personality based on fictional sidekick characters. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI '18)*. ACM, New York, NY, 307–308. DOI: https://doi.org/10.1145/3173386.3176912

[59] Benedikt Maettig and Hermann Foot. 2020. Approach to improving training of human workers in industrial applications through the use of intelligence augmentation and human-in-the-loop. In *Proceedings of the 15th International Conference on Computer Science & Education (ICCSE '20)*. IEEE, 283–288.

[60] Millecamp Martijn, Cristina Conati, and Katrien Verbert. 2022. "Knowing me, knowing you": Personalized explanations for a music recommender system. *User Modeling and User-Adapted Interaction* 32, 1 (2022), 215–252.

[61] Yongwu Miao, Tim Sodhi, Francis Brouns, Peter Sloep, and Rob Koper. 2008. Bridging the gap between practitioners and e-learning standards: A domain-specific modeling approach. In *Proceedings of the 3rd European Conference on Technology Enhanced Learning: Technologies Across Learning Contexts. Times of Convergence (EC-TEL '08)*. Springer, 284–289.

[62] Chris J. Michael, Dina Acklin, and Jaelle Scheuerman. 2020. On interactive machine learning and the potential of cognitive feedback. arXiv.2003.10365. Retrieved from https://doi.org/10.48550/arXiv.2003.10365

[63] Parastoo Mohagheghi and Vegard Dehlen. 2008. Developing a quality framework for model-driven engineering. In *Proceedings of the Models in Software Engineering: Workshops and Symposia at MoDELS 2007*. Springer, 275–286.

[64] Sina Mohseni, Niloofar Zarei, and Eric D. Ragan. 2021. A multidisciplinary survey and framework for design and evaluation of explainable AI systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 11, 3–4 (2021), 1–45.

[65] Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. 2022. Human-in-the-loop machine learning: A state of the art. *Artificial Intelligence Review* 56, 4 (2022), 3005-3054.

[66] Eduardo Mosqueira-Rey, Elena H. Pereira, David Alonso-Ríos, and José Bobes-Bascarán. 2022. A classification and review of tools for developing and interacting with machine learning systems. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 1092–1101.

[67] Dave Murray-Rust, Iohanna Nicenboim, and Dan Lockton. 2022. Metaphors for designers working with AI. In *Proceedings of the DRS Biennial Conference Series*. DOI: https://doi.org/10.21606/drs.2022.667

[68] Dave Murray-Rust, Petros Papapanagiotou, and Dave Robertson. 2015. Softening electronic institutions to support natural interaction. *Human Computation* 2, 2 (2015), 155-188.

[69] Mario Nadj, Merlin Knaeble, Maximilian Xiling Li, and Alexander Maedche. 2020. Power to the oracle? Design principles for interactive labeling systems in machine learning. *KI-Künstliche Intelligenz* 34, 2 (2020), 131–142.

[70] Ingrid Nunes and Dietmar Jannach. 2017. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction* 27, 3 (2017), 393–444.

[71] Paul D. O'Brien and Richard C. Nicol. 1998. FIPA—Towards a standard for software agents. *BT Technology Journal* 16, 3 (1998), 51–59.

[72] Ozlem Ozmen Garibay, Brent Winslow, Salvatore Andolina, Margherita Antona, Anja Bodenschatz, Constantinos Coursaris, Gregory Falco, Stephen M. Fiore, Ivan Garibay, Keri Grieman, John C. Havens, Marina Jirotka, Hernisa Kacorri, Waldemar Karwowski, Joe Kider, Joseph Konstan, Sean Koon, Monica Lopez-Gonzalez, Iliana Maifeld-Carucci, Sean McGregor, Gavriel Salvendy, Ben Shneiderman, Constantine Stephanidis, Christina Strobel, Carolyn

Ten Holter, and Wei Xu. 2023. Six human-centered artificial intelligence grand challenges. *International Journal of Human–Computer Interaction* 39, 3 (2023), 391–437.

[73] Stefan L. Pauwels, Christian Hübscher, Javier A. Bargas-Avila, and Klaus Opwis. 2010. Building an interaction design pattern language: A case study. *Computers in Human Behavior* 26, 3 (2010), 452–463.

[74] Dorian Peters, Karina Vold, Diana Robinson, and Rafael A. Calvo. 2020. Responsible AI—Two frameworks for ethical design practice. *IEEE Transactions on Technology and Society* 1, 1 (2020), 34–47.

[75] Ivens Portugal, Paulo Alencar, and Donald Cowan. 2016. A preliminary survey on domain-specific languages for machine learning in big data. In *Proceedings of the IEEE International Conference on Software Science, Technology and Engineering (SWSTE '16)*. IEEE, 108–110.

[76] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *Proceedings of the International Conference on Machine Learning*. PMLR, 8821–8831.

[77] Jeba Rezwana, and MaryLou Maher. 2023. Designing Creative AI Partners with COFI: A Framework for Modeling Interaction in Human-AI Co-Creative Systems. ACM Trans. Comput-Hum. Interact.. 30, 5 (Oct 2023), 1-28. 10.1145/3519026

[78] David Robertson. 2005. A lightweight coordination calculus for agent systems. In *Proceedings of the Declarative Agent Languages and Technologies II: Second International Workshop (DALT '04)*. Springer, 183–197.

[79] Sebastian Schelter, Yuxuan He, Jatin Khilnani, and Julia Stoyanovich. 2019. Fairprep: Promoting data to a first-class citizen in studies on fairness-enhancing interventions. arXiv.1911.12587. Retrieved from https://doi.org/10.48550/arXiv.1911.12587

[80] Stefan Schmager, Ilias Pappas, and Polyxeni Vassilakopoulou. 2023. Defining human-centered AI: A comprehensive review of HCAI literature. In *Proceedings of the Mediterranean Conference on Information Systems*, 1–12.

[81] Douglas C. Schmidt. 1995. Using design patterns to develop reusable object-oriented communication software. *Communications of the ACM* 38, 10 (1995), 65–74.

[82] Tjeerd A. J. Schoonderwoerd, Emma M. van Zoelen, Karel van den Bosch, and Mark A. Neerincx. 2022. Design patterns for human-AI co-learning: A wizard-of-Oz evaluation in an urban-search-and-rescue task. *International Journal of Human-Computer Studies* 164 (2022), 102831.

[83] Gesina Schwalbe and Bettina Finzel. 2021. A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts. DOI: https://doi.org/10.1007/s10618-022-00867-8

[84] Onn Shehory and Arnon Sturm. 2001. Evaluation of modeling techniques for agent-based systems. In *Proceedings of the 5th International Conference on Autonomous Agents*. 624–631.

[85] Ben Shneiderman. 2020. Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human–Computer Interaction* 36, 6 (2020), 495–504.

[86] Fabian Sperrle, Mennatallah El-Assady, Grace Guo, Rita Borgo, D. Horng Chau, Alex Endert, and Daniel Keim. 2021. A survey of human-centered evaluations in human-centered machine learning. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 543–568.

[87] Fabian Sperrle, Mennatallah El-Assady, Grace Guo, Duen Horng Chau, Alex Endert, and Daniel Keim. 2020. Should we trust (x) AI? Design dimensions for structured experimental evaluations. arXiv.2009.06433. Retrieved from https://doi.org/10.48550/arXiv.2009.06433

[88] Arnon Sturm and Onn Shehory. 2004. A comparative evaluation of agent-oriented methodologies. In *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*. F. Bergenti, M. P. Gleizes, and F. Zambonelli (Eds.), Springer, 127–149.

[89] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. 2021. Protoai: Model-informed prototyping for Ai-powered interfaces. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*. 48–58.

[90] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. 2021. Towards a process model for co-creating AI experiences. In *Proceedings of the 2021 Designing Interactive Systems Conference*. 1529–1543.

[91] Harini Suresh and John Guttag. 2021. A framework for understanding sources of harm throughout the machine learning life cycle. In *Proceedings of the Equity and Access in Algorithms, Mechanisms, and Optimization*. 1–9.

[92] Stephen Tang, Martin Hanneghan, Tony Hughes, C. Dennett, S. Cooper, and M. Ariff Sabri. 2008. Towards a domain specific modelling language for serious game design. In *Proceedings of the 6th International Game Design and Technology Workshop*. Liverpool, UK.

[93] Stefano Teso, Öznur Alkan, Wolfgang Stammer, and Elizabeth Daly. 2023. Leveraging explanations in interactive machine learning: An overview. *Frontiers in Artificial Intelligence* 6 (2023), 1066049. DOI: https://doi.org/10.3389/frai.2023.1066049

[94] Mike Treanor, Alexander Zook, Mirjam P. Eladhari, Julian Togelius, Gillian Smith, Michael Cook, Tommy Thompson, Brian Magerko, John Levine, and Adam Smith. 2015. AI-based game design patterns. In *Proceedings of the 10th International Conference on the Foundations of Digital Games (2015)*. Society for the Advancement of Digital Games, USA. ISBN 9780991398249.

[95] Ivan Trencansky and Radovan Cervenka. 2005. Agent modeling language (AML): A comprehensive approach to modeling MAS. *Informatica* 29, 4 (2005), 391-400.
[96] Konstantinos Tsiakas, Maher Abujelala, and Fillia Makedon. 2018. Task engagement as personalization feedback for socially-assistive robots and cognitive training. *Technologies* 6, 2 (2018), 49.
[97] Michael van Bekkum, Maaike de Boer, Frank van Harmelen, André Meyer-Vitali, and Annette ten Teije. 2021. Modular design patterns for hybrid learning and reasoning systems. *Applied Intelligence* 51, 9 (2021), 6528–6546.
[98] Niels van Berkel, Mikael B Skov, and Jesper Kjeldskov. 2021. Human-AI interaction: Intermittent, continuous, and proactive. *Interactions* 28, 6 (2021), 67–71.
[99] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y. Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–15.
[100] Katharina Weitz, Dominik Schiller, Ruben Schlagowski, Tobias Huber, and Elisabeth André. 2021. "Let me explain!": Exploring the potential of virtual agents in explainable AI interaction design. *Journal on Multimodal User Interfaces* 15, 2 (2021), 87–98.
[101] John Wenskovitch and Chris North. 2020. Interactive artificial intelligence: Designing for the "two black boxes" problem. *Computer* 53, 8 (2020), 29–39.
[102] Christina Wiethof and E. Bittner. 2021. Hybrid intelligence-combining the human in the loop with the computer in the loop: A systematic literature review. In *Proceedings of the 42nd International Conference on Information Systems*.
[103] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems* (2022).
[104] Wei. Xu. 2019. Toward human-centered AI. Interactions.. 26, 4 (Jun 2019), 42-46. 10.1145/3328485
[105] Wei Xu, Marvin J. Dainoff, Liezhong Ge, and Zaifeng Gao. 2022. Transitioning to human interaction with AI systems: New challenges and opportunities for HCI professionals to enable human-centered AI. *International Journal of Human–Computer Interaction* 39, 3 (2022), 494–518.
[106] Ke Yang, Biao Huang, Julia Stoyanovich, and Sebastian Schelter. 2020. Fairness-aware instrumentation of preprocessing˜ pipelines for machine learning. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA '20)*.
[107] Qian Yang, Nikola Banovic, and John Zimmerman. 2018. Mapping machine learning advances from HCI research to reveal starting places for design innovation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–11.
[108] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020b. Re-examining whether, why, and how human-AI interaction is uniquely difficult to design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.
[109] Chuang Yu and Adriana Tapus. 2019. Interactive robot learning for multimodal emotion recognition. In *Proceedings of the International Conference on Social Robotics*. Springer, 633–642.
[110] Mireia Yurrita, Dave Murray-Rust, Agathe Balayn, and Alessandro Bozzon. 2022. Towards a multi-stakeholder value-based assessment framework for algorithmic systems. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. ACM, New York, NY, 535–563. DOI: https://doi.org/10.1145/3531146.3533118
[111] Zelun T. Zhang, Yuanting Liu, and Heinrich Hussmann. 2021. Forward reasoning decision support: Toward a more complete view of the human-AI interaction design space. In *Proceedings of the 14th Biannual Conference of the Italian SIGCHI Chapter (CHItaly '21*). 1–5.
[112] Xiaofei Zhou, Jessica Van Brummelen, and Phoebe Lin. 2020. Designing AI learning experiences for K-12: Emerging works, future opportunities and a design framework. arXiv.2009.10228. Retrieved from https://doi.org/10.48550/arXiv.2009.10228
[113] Julian Zucker and Myraeka d'Leeuwen. 2020. Arbiter: A domain-specific language for ethical machine learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 421–425.

Received 26 March 2023; revised 19 January 2024; accepted 25 March 2024