Hammers for Robots: Designing Tools for Reinforcement Learning Agents

Matthew V. Law
mvl24@cornell.edu
Information Science
Cornell University
Ithaca, New York, USA

Zhilong Li
zl242@cornell.edu
Computer Science
Cornell University
Ithaca, New York, USA

Amit Rajesh
ar883@cornell.edu
Computer Science
Cornell University
Ithaca, New York, USA

Nikhil Dhawan
nd353@cornell.edu
Computer Science
Cornell University
Ithaca, New York, USA

Amritansh Kwatra
ak2244@cornell.edu
Computer Science
Cornell University
Ithaca, New York, USA

Guy Hoffman
hoffman@cornell.edu
Cornell University
Ithaca, New York, USA

ABSTRACT

In this paper we explore what role humans might play in designing tools for reinforcement learning (RL) agents to interact with the world. Recent work has explored RL methods that optimize a robot's morphology while learning to control it, effectively dividing an RL agent's environment into the external world and the agent's interface with the world. Taking a user-centered design (UCD) approach, we explore the potential of a human, instead of an algorithm, redesigning the agent's tool. Using UCD to design for a machine learning agent brings up several research questions, including what it means to understand an RL agent's experience, beliefs, tendencies, and goals. After discussing these questions, we then present a system we developed to study humans designing a 2D racecar for an RL autonomous driver. We conclude with findings and insights from exploratory pilots with twelve users using this system.

CCS CONCEPTS

• **Human-centered computing → User centered design**; **Human computer interaction (HCI)**; • **Computing methodologies** → *Reinforcement learning*.

KEYWORDS

user-centered design, reinforcement learning, human-agent interaction

ACM Reference Format:
Matthew V. Law, Zhilong Li, Amit Rajesh, Nikhil Dhawan, Amritansh Kwatra, and Guy Hoffman. 2021. Hammers for Robots: Designing Tools for Reinforcement Learning Agents. In *Designing Interactive Systems Conference 2021 (DIS '21), June 28-July 2, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3461778.3462029

1 INTRODUCTION

What would user-centered design look like if one were to design for a machine learning agent? How do design practices map onto a "user" when that user is a reinforcement learning (RL) algorithm? What are such a user's experiences, traits, tendencies, needs, goals, and "mental" models? In this paper we explore, both theoretically and empirically, the notion of designing tools for reinforcement learning agents. We propose research directions and questions emanating from this undertaking, present an experimental platform that enables lay users to build and evaluate tools for RL agents, and discuss insights from twelve pilot sessions with our system.

Reinforcement learning is a method of learning a behavioral *policy* for an *agent* through experience. Typically, a reinforcement learning task is formalized as a Markov decision process (MDP), wherein the agent exists in some environment *state* from which it acts, observes outcomes of its *actions*, and receives positive or negative *rewards.* The above-mentioned policy defines rules by which an agent selects an action for every possible state of the environment. RL searches the space of possible policies by interacting with the environment to find the policy which, when followed, maximizes the expected rewards of the agent (Figure 1 left).

Reinforcement learning is commonly applied to control tasks, e.g. learning a policy defining what motor torques to apply and when to apply them in order to grasp objects with a robotic arm. The way the environment of the agent is mapped onto states can significantly impact the policies RL learns [48]. This includes decisions on what aspects of the environment are modeled, how they are sampled, and how they are represented and presented to the agent.

We argue that the agent's environment can be meaningfully thought of as two distinct aspects: *tools* through which agents act on their surroundings, and the *external world* which they perceive and act upon (Figure 1 right). This division stems from insight that, while the world is relatively fixed, an agent's tool can sometimes be redesigned to better fit the task and agent using it. In an RL formulation, this relates to how the environment and task are mapped to an MDP, namely how the state space is defined based on the

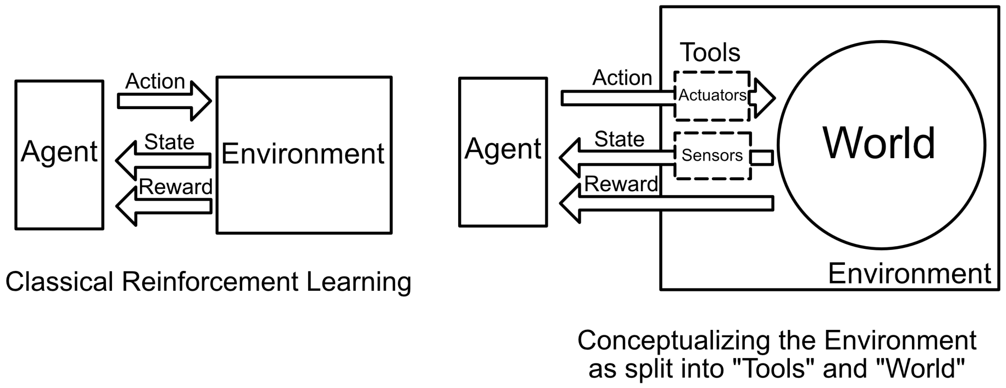

Action
State
Reward
Agent
Environment
Classical Reinforcement Learning
Tools
Actuators
Sensors
World
Environment
Conceptualizing the Environment as split into "Tools" and "World"

Figure 1: Classical reinforcement learning (RL, left) describes a method of learning a policy for an agent to act on the environment, based on observed states and rewards. In this paper, we suggest that the environment can be conceptualized as being comprised of the world (given by the task) and the tool(s) through which an agent interfaces with the world, which can be designed.

available sensor inputs and how the agent affects its state in the world, and the world itself, through actuators.

In control problems, such as the one discussed by Reda *et al.*, torque limits on a robot can increase the smoothness of policy behavior but potentially limit its performance [48]. Generalizing from this, the physical morphoplogy of a robot might be optimized, or the resolution of sensors increased, to achieve more effective policies to accomplish a task.

Indeed, recent work has sought to jointly optimize a robot's physical design parameters and control policy using RL. For example, Schaff *et al.* observe that a policy is optimal with respect to a robot's design and vice versa. They add a term ω to the MDP tuple, representing a vector of design parameters for a robot, then optimize both policy and design parameters using the same rollouts [51].

Whereas Schaff *et al.* treat the design of the agent's body as an additional optimization problem, we instead ask whether human design intervention could provide an alternative or complementary way to construct designs that improve the effectiveness of RL agents to learn and perform a task. In particular, we frame the agent as the user of the tool it is learning to control, and the tool the object of user-centered design.

The contributions of this work include: (1) a preliminary discussion of questions raised by the prospect of designing tools to be used by an RL agent; (2) a platform developed to study humans designing for such agents; and (3) insights from pilot sessions with twelve users of the proposed platform.

1.1 Learning a Policy and Design at the Same Time

Prior work has found promise in training RL agents to redesign a robot while learning to control it. In the field of embedded systems, it is a common practice to co-design software and hardware to account for complementary effects [11, 58]. A number of projects attempt to apply the same principles to robotics control. For example, Ha proposes adding design parameters to a policy network and then jointly learning the policy and design parameters, e.g. using evolutionary methods [23]. Allowing an agent to co-optimize design parameters in its environment achieved higher rewards and more quickly solved robot locomotion tasks. Ha further demonstrates the ability to optimize desired properties in the design while solving the task, such as minimizing the amount of material used, by modifying the reward function. Schaff *et al.*, as mentioned, augment the MDP with a set of design parameters, then maintain a generative distribution of effective designs, using rollouts on sampled designs to update both the design distribution and the policy [51]. They also find better performance on locomotion tasks, although joint policy and design optimization was sometimes more prone to getting stuck in local optima. On the other hand, Luck *et al.* decouple design and policy optimization to circumvent the real-world infeasibility of constructing and frequently updating a population of prototypes. They instead employ an actor-critic approach that alternates between learning a policy for an individual design and then updating that design using a global critic that accounts for design parameters and is updated during the policy step [36].

Building on the promise of work that has agents both redesign and learn to control their tools, we ask whether human capacities, principles, and tools developed for user-centered design might be useful to more efficiently design better tools for agents.

1.2 Leveraging Human Design Capacities for Tool Design in RL

Humans are well-positioned to play a role as designers of tools for RL agents. Indeed, designing is a core human competency–Norman writes that "tool making and usage constitute one of the defining characteristics of our species" [44]. Egan and Cagan argue that, despite the benefits of computers' super-human speed, precision, repeatability, and consistency when optimizing a design, humans' creativity, flexibility, and intuition are also essential [16]. Schön describes the human processes of constructing and communicating across design worlds, including "seeing-moving-seeing", recognizing unintended consequences, and appreciating design qualities, as difficult to reproduce with a computer [52]. A key challenge

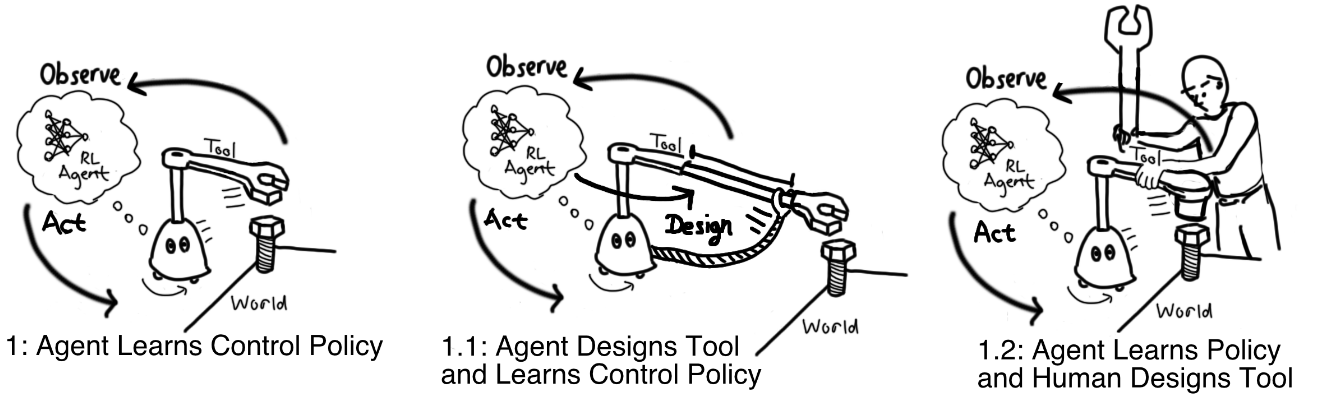

Observe
RL Agent
Tool
Act
World
1: Agent Learns Control Policy
Observe
RL Agent
Tool
Act
Design
World
1.1: Agent Designs Tool and Learns Control Policy
Observe
RL Agent
Tool
Act
World
1.2: Agent Learns Policy and Human Designs Tool

Figure 2: A typical RL agent learns a behavioral policy through acting on and observing the world (Section 1). Recent work has proposed RL agents that learn to optimize the design of the tools through which they interact with the world at the same time as they learn a policy (Section 1.1). In this work, we propose leveraging human design capacities to optimize the agent's tools as the agent learns a policy (Section 1.2).

in computational designing arises from the ill-definition and underconstrained nature of most design tasks. For humans, however, Dorst posits abduction, a reasoning process through which humans frame and impose structure on tasks, as the core of design thinking [15].

In the field of human-computer interaction, user-centered design (UCD) is a popular framework through which human designers frame design tasks [56]. UCD broadly starts with understanding the intended user, their needs, desires, tendencies, and experiences, and it involves the user throughout the design process. UCD is both a philosophy and a methodology, an imperative to prioritize knowing the user, and a set of tools to achieve this [1].

This work investigates in what sense UCD could be an appropriate framework under which to approach this task. After all, the RL agent is the intended user of the tool and understanding the agent and its needs seems crucial to success. However, due to the inherent differences between human and AI agents, this framing introduces a set of research challenges.

2 USER-CENTERED DESIGN OF TOOLS FOR REINFORCEMENT LEARNING AGENTS

UCD prioritizes *knowing the user* as the center of the design process, but what does it mean for a designer to "know" an RL agent as a user? How should a human designer understand the experiences, beliefs and tendencies, or needs and goals of a computer agent? Is there any place for affective aspects of UCD, e.g. building empathy between user and designer, or designing enjoyable experiences, and if not, how does their absence affect the designer? Finally, do methods that designers typically use to understand and involve human users in the design process apply when designing for agents? Complicating this question, different types of agents may afford different kinds of "knowing", and understanding an agent may require insight into its inner workings which may or may not be available to the designer.

One framing under which to consider these questions is through the lens of explainable AI (XAI), which explores how AI systems, including RL agents, might be understandable to humans, dealing with questions of both *transparency* and post-hoc *interpretability* of the agent's decisions. Embedded in this discussion is a question about what kinds of explanation serve different human purposes. For example, a mechanistic explanation may be useful for debugging an agent, but less so for building users' trust in a model. Páez argues that pragmatic considerations should take precedence over complete factual accuracy in the context of XAI [46]. In the case of designing for RL agents, what kind of explanations do human designers need to be effective? And how much of the actual mechanics behind how an agent interprets the world and makes decisions do they need to understand?

Explainable reinforcement learning (XRL) is a subset of XAI. XRL is an extremely new field, and a recent survey of the state of the art in deep XRL finds 15 projects, with a bias towards post-hoc explanation over transparency, as well as a trend towards ad-hoc explainability techniques, which do not generalize well [26]. Most explanations are aimed at experts and not lay users. Designers who work with AI systems tend to already face significant knowledge gaps about the underlying technology and what it is capable of [61]. Nonetheless, the advancement of XRL tools, as well as an increasing focus on developing agents with explainability built in, may provide important tools to support designers' understanding of the agents they are designing for.

In the rest of this section, we lay out four theoretical aspects of designing tools for RL agents, as listed below with related questions. Our exposition on these four aspects presents an initial and incomplete perspective on what it might mean to understand and design for agents as users that we hope will spur further discussion and expansion. A list of mappings between some elements of user understanding that arise in the following discussion and corresponding RL concepts can be found in Table 1.

- **Understanding Agent Experiences:** Studying how human users perceive and experience interactions with the world can provide insight into user behaviors and help designers support desired experiences. In a basic sense, perception plays an explanatory role in how users act and process information [7]. More broadly, Forlizzi and Battarbee refer to physical, sensual, cognitive, emotional, and aesthetic dimensions of experience and differentiate experience itself, as a constant stream of consciousness, from both individual experiences and experiences that are shared [18]. Designers have the power to shape user experiences–Laurel, for example, describes interface design as engaging a sort of interactive, participatory theater [34]. Wright and McCarthy argue that humanist design requires a holistic appreciation of users' lived experiences and how they make sense of them, with a commitment to enrich those experiences through design [60].
  RL agents can arguably be thought of as the products of their experiences, including the states they traversed, the engineering and design choices and circumstances that shaped those state traces, the information that was extracted from them, and how it was stored and applied. How might a human designer understand these experiences as perceived by the agent, and how might they judge the significance of individual experiences?
- **Understanding Agent Beliefs and Behavioral Tendencies:** Another important consideration in UCD regards how users believe systems work and how this informs their behaviors and tendencies. For example, humans are thought to develop mental models as internal representations of external systems, at varying levels of abstraction [57]. These models evolve as users interact with a system [43] and can be chaotic or misconceived, in both novice and expert users [8]. Understanding users' mental models can be useful to predict or explain their behaviors [43]. Designers can use this knowledge to cater their designs to users' beliefs, or to anticipate errors and communicate more helpful conceptual models through their designs [8, 55].
  What does it mean for a designer to understand how an agent's experiences and priors become encoded into its beliefs and tendencies? What kinds of "mental models" do agents learn and employ when acting? How do an agent's behaviors reflect its beliefs, and how should these be observed and analyzed when designing? How does this understanding relate to the structure of the agent's learning algorithm?
- **Understanding Agent Goals:** Understanding users' needs and goals is central to the aims of UCD. In designing user-centered systems, Norman and Draper write, "Concern for the needs of the users should be primary" [45]. Norman defines goal formation as the first step in a user performing any action and makes designers responsible for helping people bridge the gulfs of execution and evaluation between their goals and the physical world [42].
  How can designers come to understand the goals and desires of an agent? How does this understanding relate to the engineering of the reward function defined for the task? UCD designers also sometimes differentiate between different kinds of human goals. For example, Cooper breaks user goals down into end-goals, experience goals, and life goals [10]. Do there exist analogous structures in the goals agents seek to achieve and if so, how might understanding these different kinds of goals support RL-agent centered design?
- **Understanding Agents as Anthropomorphic and Empathy-Evoking:** Empathy is a powerful and widely-used instrument in UCD for human users. Wright and McCarthy situate empathy as a core part of how designers understand users and their perspectives [59]. However, is there a place or a need for empathy when designing for agents? How does a human's tendency to anthropomorphize computer agents affect RL-centered tool design?

In the following, we expand on each of these aspects of understanding agents as users. We then conclude this section with a discussion of elicitation formats and appropriateness for RL-centered tool design.

2.1 Understanding Agent Experiences

How might a designer of tools for an RL agent understand experiences as perceived by the agent? For RL agents, the most natural way to represent experience is by the notion of "state traces", or trajectories through the state space that were used to evaluate and improve the policy. In simple cases, it may be useful to directly visualize the features that define the state space. Consider, for example, an agent balancing an inverted pendulum, i.e. trying to keep a pole upright by moving a cart at its base back and forth. Suppose the agent learns to balance the pendulum based on the position of the cart, the current angle of the pole, and the rate of change of each. Visualizing traces of these features could give a designer a reasonable window into such an agent's experiences. In more complex scenarios, however, a designer may need to understand how an agent *interprets* the state input. This may be especially true if the state representation is very high dimensional. For example, if an agent learns how to balance an inverted pendulum based on a video feed, a designer might seek to understand the image features the agent learns to consider in each frame.

Saliency maps are a popular approach to interpreting how image features drive an agent's policy choices by highlighting pixels that are important to a model's output. For example, Greydanus *et al.* measure how perturbations to the input state affect value estimations or policy outputs [22]. Other types of saliency maps used in deep reinforcement learning are based on gradients [49], object-recognition [29], or attention [41]. Saliency maps are a promising approach to visualize what an agent is "seeing" when it makes decisions in a human-interpretable way. In fact, Zhang *et al.* find that what agents pay attention to tends to become more humanlike as they learn and improve, using saliency maps of agents learning to play Atari games [62]. However, Atrey *et al.* caution against imputing causality to such saliency maps or the explanations humans generate from them, counterfactually disproving saliency-based explanations for agent behavior in several case studies [4].

UCD Concepts	RL Concepts
Experiences	State or Observation traces
Beliefs, Mental Models	Dynamics models, Observation models
Behavioral Tendencies	Action traces, Q-values, Policy, Rollouts
Goals	Reward functions

Table 1: This table maps some UCD concepts to similar concepts in RL agents.

2.2 Understanding Agent Beliefs and Tendencies

How can a designer understand the ways in which an agent's experiences become encoded into its beliefs and tendencies? Here the structure of the agent's learning algorithm may be an important consideration. For example, some RL agents employ model-based algorithms that learn models of the world dynamics from experience. For such agents, it may be possible to directly interrogate what the agent believes about the world without necessarily observing the agent in action. For example, Ha and Schmidhuber's World Models approach learns to predict upcoming states of the world (e.g. a yet unseen turn in the road) as input to a controller [24]. By visualizing the agent's predictions, it is possible to literally see what the agent believes about the future in a given state. For model-free agents that implicitly encode knowledge about the world, however, it may only be possible to infer beliefs from the decisions that they make.

In complex tasks with large state or action spaces, analyzing an agent's tendencies might require more nuance than directly querying a network or table. In one approach, Rupprecht *et al.* [50] train a generative model that can reconstruct states with respect to some target function. Such a model could be used to generate samples of inverted pendulum states in which the agent tends to value performing one action, e.g. pushing the cart to the right, as an alternative to simulating and combing through many full episodes. Alternatively, Sequeira and Gervasio suggest a framework for extracting interesting examples of behavior from an agent, based on features like frequency and diversity [53]. Such automation could minimize how many samples of agent behavior a designer needs to see to understand its tendencies.

It may also be useful to characterize an agent's tendencies more abstractly. For example, an agent's current skill level chould influence whether a designer makes small, tailored changes or wholesale modifications intended to shape future behavior. In a similar vein, deducing an agent's willingness to explore a space, which can depend on prior experiences and hyperparameter choices, could be useful to predict how the agent will tolerate design changes that trigger new experiences. Conversely, tool design could be a way to encourage a tentative agent to explore more.

2.3 Understanding Agent Goals

With that said, how should designers understand the goals of RL agents? RL agents are inherently reward-maximizing; nonetheless, they typically are created for some purpose, which is then encoded into the agent's reward structure. This process of mapping goals to reward functions is called reward engineering, and is a critical aspect of RL agent design that becomes more important and difficult as contexts become more complex and agents more autonomous [14]. Since an agent's creator lacks either perfect foresight or the ability to constantly supervise the agent, this encoding is often imperfect, unable to account for all the situations an agent may encounter and creating a sort of principal-agent problem [25]. An agent thus in some sense can be thought to have both purposeful existential goals and immediate reward-maximizing goals as encoded in the structure of the MDP.

Understanding both the agent's underlying purpose and reward-maximizing goals could help to appropriately frame a design task. For example, suppose a proximity sensor at the tip of an inverted pendulum triggers a negative reward every time that sensor touches the ground. Under unexpected circumstances, the pendulum is deployed on a precipice, and the agent learns that, by hanging the pendulum horizontally over the edge of the cliff, it can avoid punishment indefinitely. A designer who understands both the agent's existential goal of balancing a pole and the way the reward structure encodes this goal could seek to redesign the pendulum to address this, e.g. by moving the sensor closer to the base of the pendulum so that it triggers on contact with the cart whenever the pole goes horizontal.

Note that this problem framing was facilitated by an example of unexpected behavior from the agent. Even in the absence of explicit knowledge about an agent's high-level goals or reward structure, visualizing relationships between agent behavior, rewards, and environment states could help designers to build theories about agent goals and ascertain design problems. Deshpande *et al.*, for example, suggest the utility of interactively visualizing reward traces alongside corresponding environment states to understand the relationships between reward components, behavior, and contexts that can help to tease out problems like reward-hacking [13].

2.4 Anthropomorphism and Empathy when Designing for Agents

What role, if any, might empathy play when humans design tools for RL agents? Wright and McCarthy describe empathy as a pragmatic, dialogical process, building on two popular definitions: (1) the recognition and feeling of another's emotions, and (2) intersubjective articulation of another's context through one's own [59]. This characterization exposes two critical aspects of empathy: affective understanding and differentiating the other from the self.

While humans may anthromorphize computer agents, empathy, as so defined, is incompatible with self-referential design. When designing for agents, it seems important to appreciate and understand RL agents as agents and not as humans. Our tendency to impute concepts like affect, agency, and sociality to computers makes possible rich interactions that have fostered powerful work in persuasive and affective computing (e.g. [12]). However, if RL agents do not act with or understand emotions, designing as though they do could have negative effects. Note that caveats about empathy also apply when designing for human users. Bennett and Rosner warn that empathy-building exercises can sometimes trivialize the experiences of others [5], while Heylighen and Dong encourage designers to remember and respect the limits of our ability to understand others' experiences [27].

There is, nonetheless, significant work that seeks to develop emotion as a functional component of RL agents. One such direction focuses on defining more complex feedback signals for agents than simple task-defined reward. For example, Marinier *et al.*'s agent acts on emotional responses to external stimuli, including suddenness, unpredictability, and pleasantness [37]. Similarly, Sequeira *et al.* propose an agent that is motivated intrinsically, rather than by task reward [54]. Both approaches outperform traditional RL agents on selected tasks. As a step further, Jacobs *et al.* map RL primitives back to human emotions like joy, distress, hope, and fear using psychology and behavioral science [30].

If agents are designed to act on emotions, especially emotions modelled from human emotions, then empathy-building when designing could look similar for agents and humans, and yield similar benefits. With all that said, emotional agents comprise only one subfield in reinforcement learning. Designers should take care to acknowledge that agents are not human, and seek to understand how, if at all, a particular agent experiences emotion before relying on empathy to understand the agent as a user.

2.5 Verbal and Non-Verbal Elicitation Methods for RL-Agent-Centered Tool Design

Giacomin classifies methods used by human-centered designers into (1) factual data and models about humans, (2) verbal and non-verbal tools to capture needs, desires, and meanings (e.g. interviews, think aloud, personas, cultural probes, observation), and (3) tools that simulate possible futures (e.g. focus groups, role playing, experience prototypes) [20].

Most of the aspects discussed above relate to point (1) and (3) in this formulation. How might methods to extract information from users, verbally and nonverbally, apply to agents? Ehsan *et al.*'s work on rationale generation provides one interesting and promising direction for developing tools that allow human designers to interrogate agents [17]. The authors trained a sequence to sequence network that translated state-action pairs to natural language explanations, trained on human explanations generated while observing an agent play Frogger. This brings to mind Páez's discussion on factivity in explanations: are explanations learned based on human interpretations of agent behavior useful when designing for agents, or might this stray too far towards self-referential anthropomorphism? Cideron *et al.* learn textual descriptions of goals based on trajectories without human explanation [9]. They do this utilizing hindsight experience replay, which relabels the goals for failed trajectories based on experience, to learn a textual instruction generator, which is then interpretable by humans.

More conventional RL methods may support nonverbal elicitation with agents, although not without design challenges of their own. For example, interactive reinforcement learning (IRL) puts humans in the loop to guide agents in various ways that could provide opportunities to probe and explain agent behavior [3]. However, there are still open questions about human-agent interaction in IRL. Krening and Feigh, for example, found that probabilistic and time-delayed responses to human intervention reduced the perceived transparency and intelligence of an agent [32].

3 CHOPSHOP: A PLATFORM FOR AGENT-CENTERED DESIGN

Having discussed ways in which concepts of designing for users might map to agents, we now describe a platform developed to actually observe humans designing a tool for an agent performing a benchmark RL task. This platform, inspired by open-ended technology probes [28], is not a prototype intended to support professional designers with real-world tasks. Rather, it is broadly intended to investigate how humans perceive and attempt to design for agents cast as users, interrogate how this activity may ultimately influence design outcomes and agent performance, and identify challenges and future directions in this space. In this work we report preliminary observations about how pilot users perceived and designed for a specific agent through our platform, as well as examples of how their interventions impacted agent performance. In the future, we hope to prototype and evaluate different visualization and elicitation methods through this platform in order to support empirical characterization of the different forms of agent-as-user understanding discussed above and their usefulness in the context of agent-centered design.

The platform, *ChopShop*, asks users to design a 2D race car for an RL driver (Figure 3). Our choice of task was partly motivated by a pragmatic desire to target ChopShop at novices with some familiarity of autonomous agents, designing, and/or the tool being designed. As an analogue to a commonly encountered real-world scenario, we hoped that some elements of car design would be relatable for those with experience driving or a basic understanding of car mechanics. We also hoped that designing in a 2D environment, while less realistic, would prove more accessible to a broad audience in terms of the complexity of both the design space and the agent driving the car. Finally, the existence of a well-known RL environment for testing 2D racing agents [6] offered performance benchmarks and future flexibility to rapidly prototype design interactions with different driving agents that might support different aspects of understanding or elicitation methods. For example, while the platform in this work employs a model-free RL driving agent, one might imagine using a model-based agent such as in [24] to explore the usefulness of visualizing an agent's model of the world when designing for it.

With the goal of observing the process and outcomes of participants designing for an agent, ChopShop was developed around the following design principles:

- **Create an open-ended design situation**, giving the participant the freedom to frame the design task around the agent as they choose, including what features to optimize and what outcomes or agent behaviors to focus on. Again, drawing on the tradition of technology probes [28], we wanted our platform to create a design situation with enough freedom for participants to interpret and create structure in how

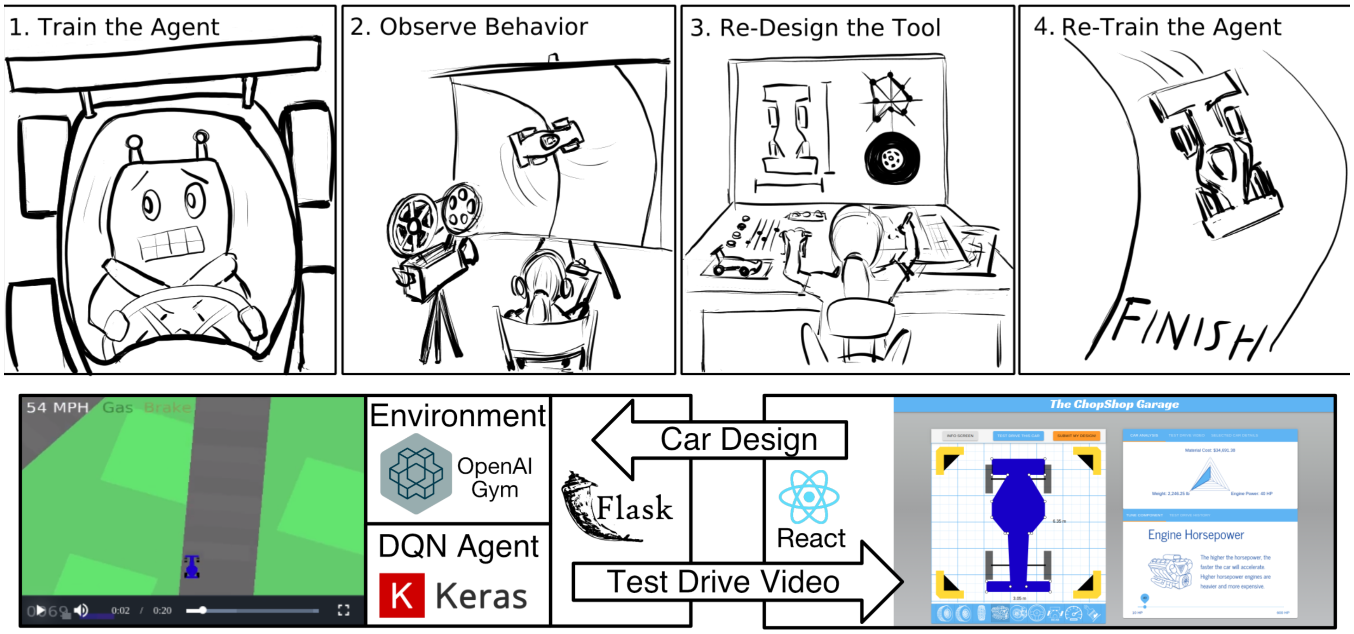


Figure 3: The ChopShop designer starts by (2) watching videos of a (1) pre-trained agent driving a default car. Based on what they observe, they (3) redesign the car for the agent, prototyping and testing several new designs with the agent. Once the designer is satisfied and finalizes the changes, the agent can be (4) retrained using the new design. Under the hood, the designer prototypes cars in a React app, submitting them to be test driven through a Flask backend. For each test drive, the Flask API loads a pretrained Keras model to instantiate a DQN driver, and initializes a CarRacing environment with the modified design. Once the test drive is complete, a video is returned to the designer via the React app, along with the episode reward.

they went about designing for the agent, allowing us to learn from their choices and reflections.

- **Encourage a user-centered approach.** Our primary goal in developing this platform was to observe humans taking a user-centered approach to designing for agents. We hoped to encourage participants, through framing and the flow of the platform, to engage in a UCD process centered around the agent. This was especially important insofar as we did not want to limit our participants to practicing designers.
- **Encourage prototyping** but allow the participant to choose how often, when, and why to try out a design. Iteration and testing with users are important aspects of UCD [21]. We wanted our platform to make it possible for participants to prototype and test ideas quickly with the agent, while still not enforcing a particular approach.
- **Manage expectations for the autonomous driver.** Humans can carry misaligned expectations for autonomous agents [31, 38] that can impact their experiences of and with them [33, 39]. To minimize the influence of preconceptions on the design process (and instead encourage the designer to seek to understand the agent as a user) we wanted to find ways to manage those expectations.

3.1 The ChopShop Interface

Ultimately, ChopShop was designed as a web platform for a designer to learn about an RL racecar driver, while directly prototyping and testing new designs to support the driver. The full technology stack underlying the platform is described in Figure 3. Code for running ChopShop locally will be made publicly available at https://github.com/hrc-d/hfr-chopshop.

ChopShop starts with a short tutorial (Figure 5), asking the participant to imagine themselves as a new employee for a firm that designs cars for autonomous drivers. Note that this framing delineates the agent as a driver from any intelligent systems which may compose the car itself. To manage expectations with respect to the driver as an AI, the tutorial tells participants that the first client for whom they will be designing is an autonomous driver that is still learning to drive. The tutorial finally walks through the user interface and describes the car features that participants can modify.

After the tutorial, participants are guided through a process inspired by Norman's iterative cycle of human-centered design (observation, idea generation, prototyping, and testing) [42]. They are first asked to observe the driver, watching ten videos of the agent driving a starter, default car on fixed randomized tracks. As they watch these videos, participants fill out a form describing the driver's approach and what the driver is doing well or struggling with. Additionally, they are asked to suggest hypothetical design solutions to support the driver.

Participants then move to the workshop interface which is pictured in Figure 4 to prototype and test designs with the agent. On the left side of the interface, participants can reshape the body of the

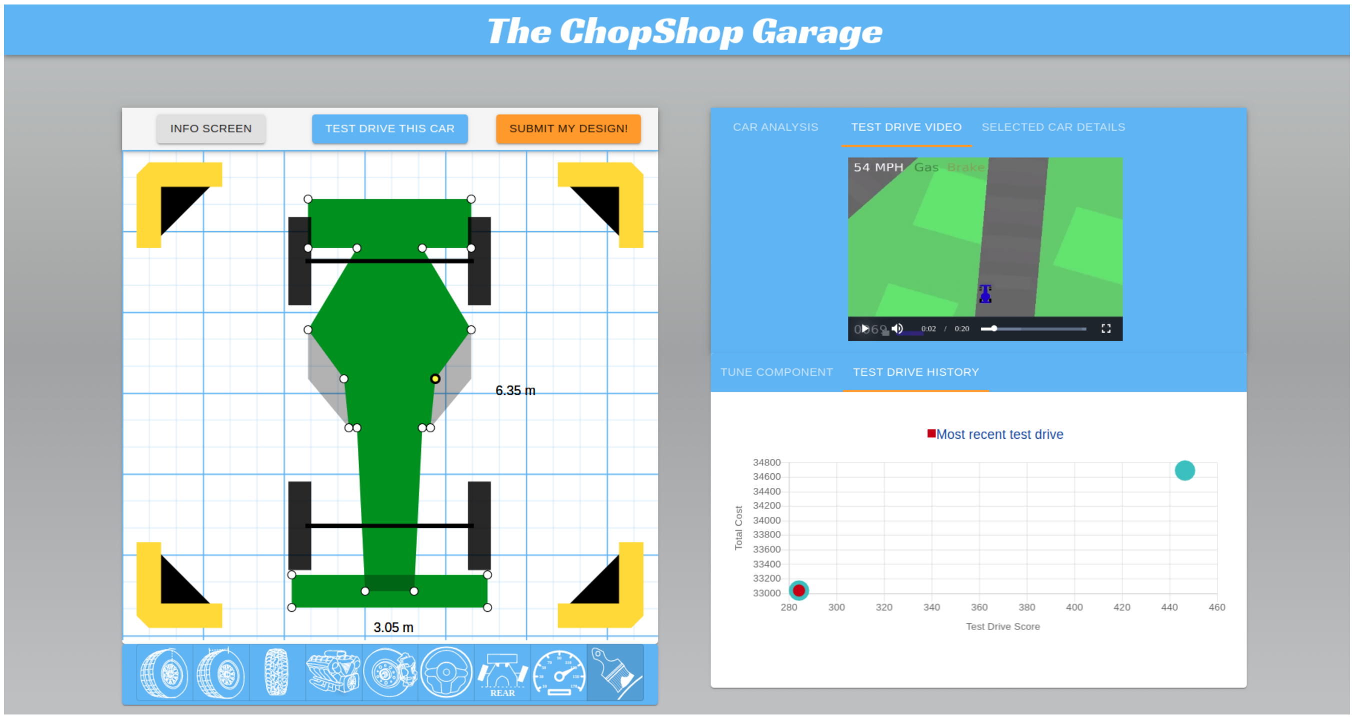


Figure 4: This is a screenshot of the *ChopShop* design platform. On the left side, the designer can adjust the shape of the car, initiate a test drive, or select a tunable feature on the bottom row. On the right side, they can adjust sliders to tune a selected feature and view configurations, outcomes, and videos of the agent for any of the designs they have test driven.

car or select other features to tune, including the car's engine power, tire tread, wheel radius and width, steering sensitivity, brake sensitivity, rear steering, maximum speed, and the car's color. Each of these features can be adjusted using a slider, which is accompanied by a short description of the feature.

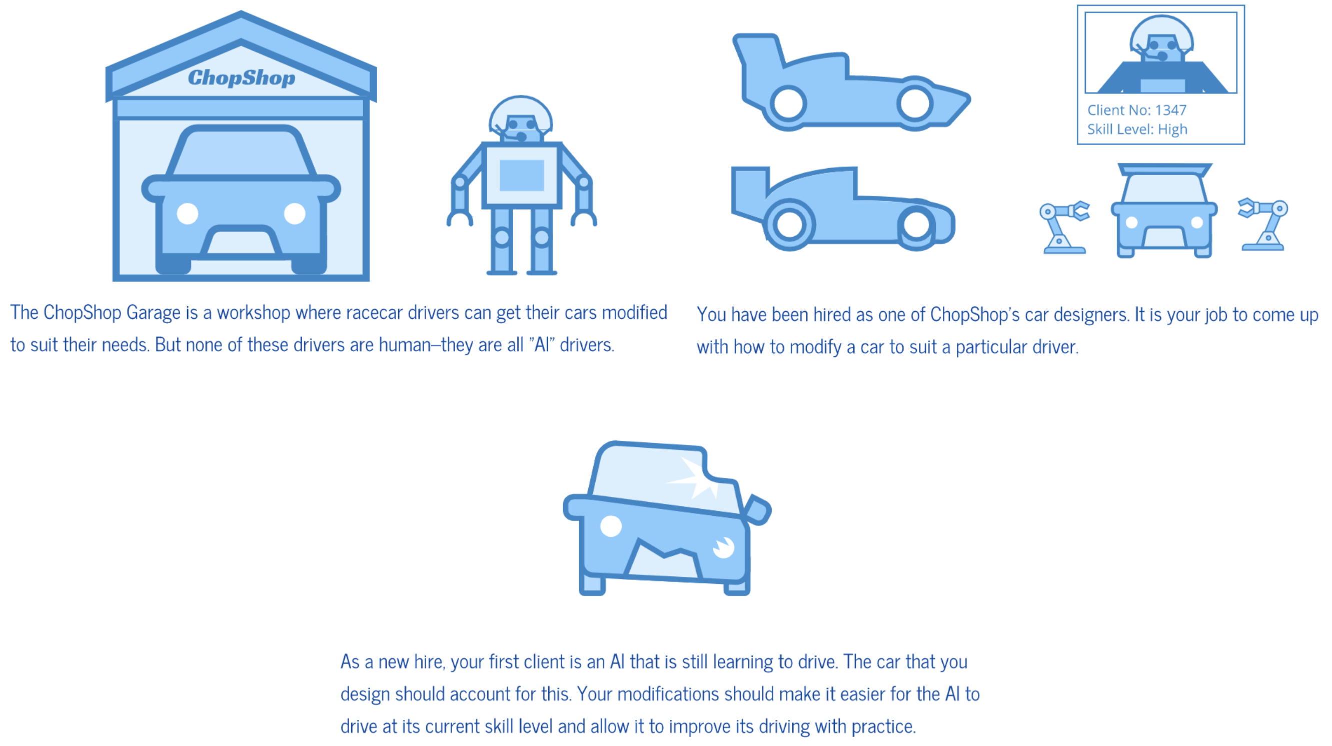
At any time, participants can ask the agent to test out a design on a random track. When they initiate a test drive, the agent is loaded from disk into a new environment, a single episode is run without any training, and a video is recorded and returned to the participant. Participants can compare test driven cars plotted by observed reward and estimated cost on a scatterplot, with the ability to inspect historical designs, overlay them on the current design, or rewatch the test drive videos. Participants are not instructed in how to interpret the videos or apply their observations to modify the design. Once the participant is finished prototyping, they submit a final design, along with a description and design rationale.

3.2 The Agent: A DQN Driver

Car Racing is a simulation environment created by OpenAI as a benchmark task for RL agents [6]. It provides a 96x96x3 pixel top-down viewport displaying a car sprite driving over a road passing through grass with a dashboard. The environment exposes three actions to an RL agent as scalars for the accelerator, brake, and steering angle. At each step, the environment returns a reward of -0.1, along with a positive reward of $1000/N$ for each road tile the encounters, where N is the number of road tiles on the map. An episode terminates when the car visits all the road tiles, if a max number of steps is reached, or if the car drives outside the map boundaries, the last incurring a -100 reward penalty.

The driving agent used in ChopShop learns via a variant of Deep Q-Network (DQN) [40], where we set a discount rate $\gamma = 0.99$, a replay batch size of 32, and employ an *Adamax* optimizer with learning rate $\alpha = 0.001$. Borrowing from [47], we discretize the action space to allow the agent to select between applying the gas fully, the brake fully, and the steering wheel to a discrete angle for each time step, and we extract state space features from the dashboard and a pixel area around the car. Also following on [47], we use an ϵ-greedy policy with a decaying ϵ of $1/\sqrt{n+100}$ where n is the episode number. We stack the last four frames in a single observation to add a temporal dimension to the state. Finally, to reduce overestimation errors and increase the stability of training, we maintain a queue of the 10 most recent Q-networks and take an average as input to estimate the true Q-value when updating the main Q-network [2].

To strike a balance between an agent that is too advanced to present design opportunities and one that is too raw to exhibit clear behaviors, we pre-trained the agent for 251 episodes with one experience replay step every 50 frames, at which point it averaged 326.53 reward over repeated test drives on 50 random tracks. Further retrainings of the agent have an experience replay every 20 frames.



The ChopShop Garage is a workshop where racecar drivers can get their cars modified to suit their needs. But none of these drivers are human–they are all "AI" drivers.

You have been hired as one of ChopShop's car designers. It is your job to come up with how to modify a car to suit a particular driver.

As a new hire, your first client is an AI that is still learning to drive. The car that you design should account for this. Your modifications should make it easier for the AI to drive at its current skill level and allow it to improve its driving with practice.

Figure 5: This figure shows the first three screens of the ChopShop user tutorial. Users are introduced to the platform as designers for a garage that customizes cars for AI drivers. They are then tasked with designing a car for an AI that is still learning to drive.

4 PILOT SESSIONS

We ran twelve exploratory pilot sessions with ChopShop and participants recruited from members of the research team, labmates, and friends. From these pilots, we hoped to learn broadly about (a) how participants perceived the agent as a user, (b) how they approached designing for the agent, and (c) what effects their designs had on the agent's behavior. These pilot sessions were not conceived as an experimental study. We did not seek to evaluate testable hypotheses about our research questions, and the participant sample, drawn at convenience, contained potential bias due to familiarity with the project or the researchers. Our intention was instead to identify preliminary research directions with respect to understanding and designing for agents by observing interactions with ChopShop. A rigorous study to evaluate hypotheses culled from this exploration is envisioned as future work.

Participants were sent a link and asked to follow the instructions in the tutorial. Participants were allowed to ask clarification questions about the system and the agent, however we did not directly observe or record the sessions, nor did we employ a protocol like think-aloud. Participants were allowed to work until they were satisfied with their design. No time limit was imposed, although participants were limited to a single session.

Participant perceptions of the driver and their designs were collected via a set of free-response questions before and after prototyping a car design for the driver (see Table 2) as described in the system description above. These questions probed participants' perceptions of the driver and their design rationales. Additionally, we collected each design that participants test-drove during the study via system log files, along with their final, submitted designs. The final designs were test-driven with the DQN agent to evaluate their performance, as described in the Section 5.3.

4.1 Participant Demographics

Eleven of the twelve participants provided demographic information. Of these eleven participants, five identified as female and six as male. Participant ages ranged between 19 and 33 years old, with a mean age of 24.6 years and standard deviation of 4.86 years. In the demographic survey, participants rated their familiarity with the following on a scale of 1 (*not familiar at all*) to 5 (*extremely familiar*):

- Driving a car ($mean = 3.73$, $std = 1.10$)
- Car mechanics ($mean = 2.18$, $std = 0.874$)
- User-centered design ($mean = 2.91$, $std = 0.944$)
- Reinforcement learning ($mean = 2.82$, $std = 0.982$)

All twelve participants consented to the use of their pilot data, anonymously, in this publication, and we also cleared its use with our institutional review board. One participant accidentally submitted after a single test drive and was allowed to redo the study at a later date.

Pilot Session Questions	
While Observing the Driver Before Prototyping	
Q1	Describe the driver's approach: How would you describe how the AI is trying to drive around the track?
Q2	What is the driver doing well? Are there specific aspects of driving it seems to have mastered?
Q3	What is the driver struggling with? Are there aspects of the track, car, or driving basics that it is struggling with?
Q4	What can you modify the car to improve? What is the driver struggling with that you think you can alleviate by modifying the car?
When Submitting the Final Car Design	
Q5	Please describe your car design:
Q6	Please describe the reasoning behind your design choices:

Table 2: This table presents the questions asked of pilot participants before and after prototyping a car for the RL driver.

5 PILOT SESSIONS FINDINGS

Data collected from the pilot sessions included designs tested and submitted by each participant, as well as answers to the questions in Table 2. Two of the authors collaboratively read through the participants' answers to identify common themes, then did a second pass to count instances of those themes, cross-referencing participants' design trajectories when helpful. Additionally, the submitted final designs were test-driven and trained on to quantify their performance, as described in more detail below.

5.1 Human Perceptions of an Agent as the User

Based on their comments, participants were largely unimpressed with the agent's ability as a driver. Four participants described the agent as an amateur, referring to it with words like "beginner", "novice", and "raw". All participants observed that the driver struggled with steering control. Further, some participants hypothesized root problems behind the driver's steering failures: two believed the driver could not identify when to turn in the track, while six believed that the driver could not figure out the direction of the turn and/or how sharp the turn should be. Two participants further remarked on the driver's inability to recover, once it had left the road. However, five participants noted that the driver was good at driving in a straight line, and four were impressed by the driver's ability to accelerate.

5.2 Design Strategies

While all participants diagnosed the driver's struggles with driving, they described different strategies to improve its performance. Unsurprisingly, eleven out of the twelve directly focused on changing how the driver performed on turns during the design phase (the twelfth instead tried to improve how the driver handled when driving straight). All eleven tried to improve the car's steering system in some way, e.g. by experimenting with the steering sensitivity, the car's mass, or properties of its wheels. Nine additionally tried to regulate the car's speed, e.g. by adjusting the engine horsepower, the max speed limiter, or modifying the car's body.

In tuning features of the car to support better driving, however, participants seemed split on whether to increase the amount of control the driver had over the car or constrain its worst tendencies. Two participants made changes to give the driver more control over the car, for example by increasing steering and brake sensitivity and setting higher speed limits. Conversely, six made changes that limited the driver's control over the car by reducing steering or brake sensitivity or lowering speed limits. As Participant 4 put it, "I chose low steering sensitivity and low rear steering power because I found the driver likes to turn sharply when he's slightly off the track, thus making it worse. Thus I restrict how much he can turn mechanically. I also chose a small engine horsepower and limit the top speed to less than a half." Four participants combined both strategies, adding driver control in some areas and removing it from others.

Curiously, before starting to prototype designs, two participants suggested adding features to the car that were outside the scope of the ChopShop interface. Participant 3 wanted to add some capability for the car to sense the lane, and Participant 11 hoped to create a signal that would alert the driver when the car went off the track. ChopShop unfortunately did not expose design features that could support these ideas, nor did it allow user-defined design features, although this would be an interesting direction to consider in the future.

The actual features which each designer modified from the original car in their final design are displayed in Figure 6. Any changes to the car's shape are indicated by the "body shape" column and an image of the car is also shown. There is diversity in both what and how many features participants ultimately chose to modify. For example, Participant 5's final design changed only two features, while Participant 1 modified all ten. The figure also enumerates the number of (non-unique) designs that each participant had the agent test-drive during the design phase. Here, again, we observed a wide range of approaches: participants tested as few as three and as many as eighty-two designs with the agent.

5.3 Effects of the Human Designs on the Agent

The car designs had varied influence on the performance of the driving agent. We measured this influence by generating fifty random road maps, then test driving each human-designed car on each map with the agent. We also test drove the original, unmodified car forty

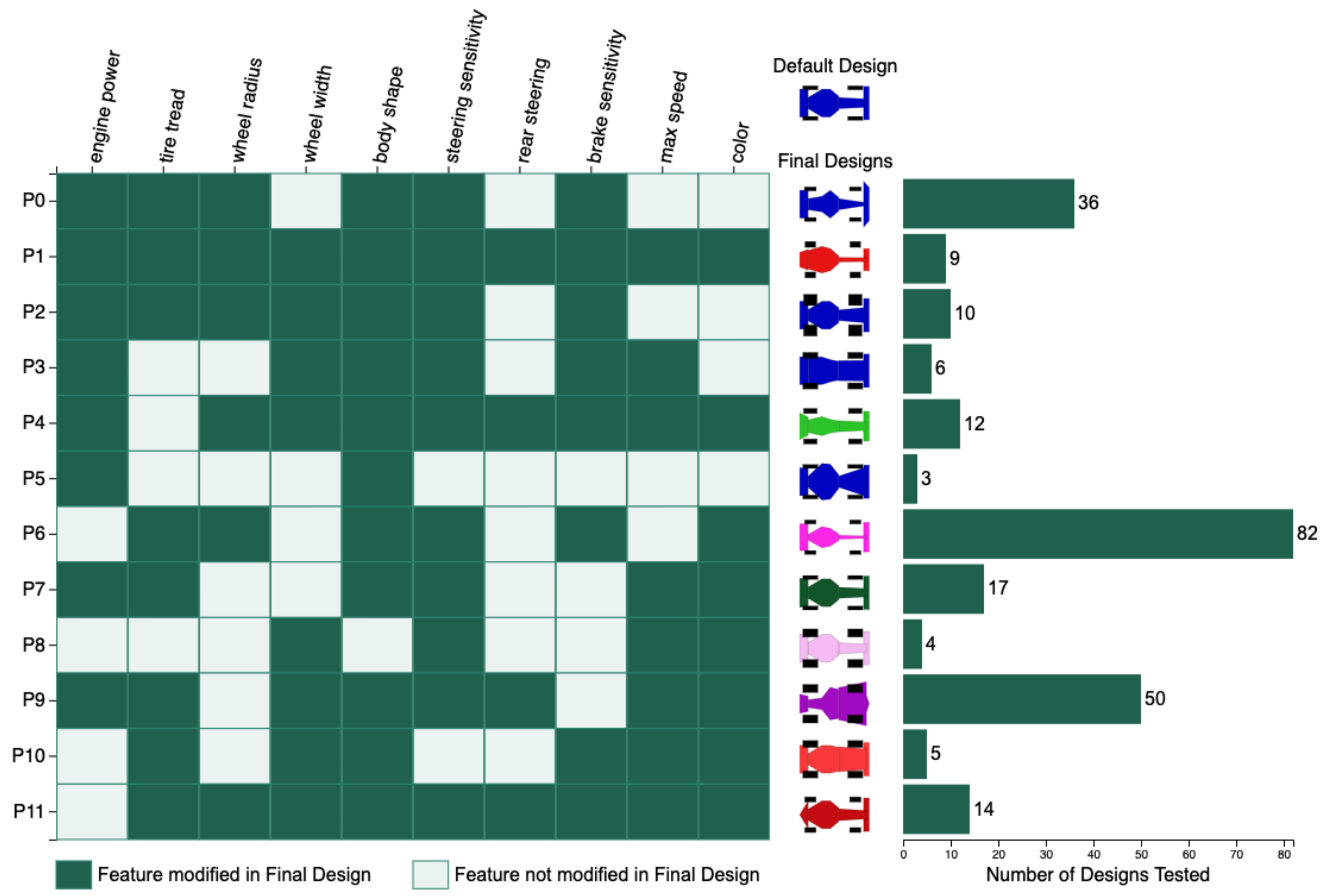
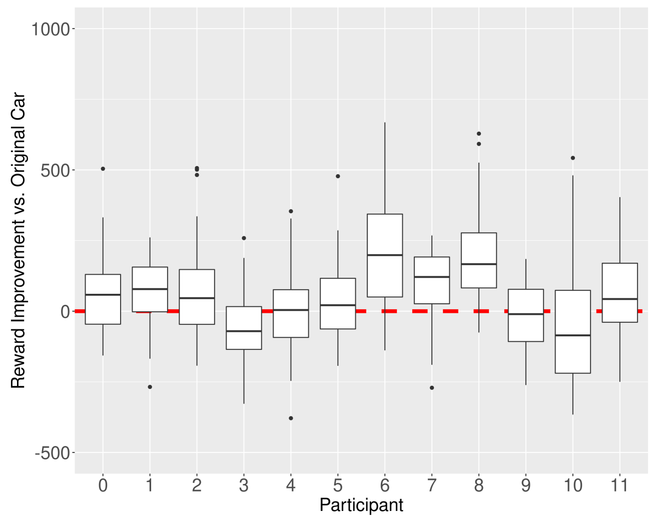


Figure 6: The above figure displays, on the left, the design features modified by each participant's final car design. For example, Participant 8 (P8) modified the wheel width, steering sensitivity, max speed, and color of the car in their final submitted design. To the right of the feature matrix, the final design produced by each participant is visualized, as well as a bar representing the number of designs tested by that participant while designing. The default, unchanged design is also visualized as a reference.

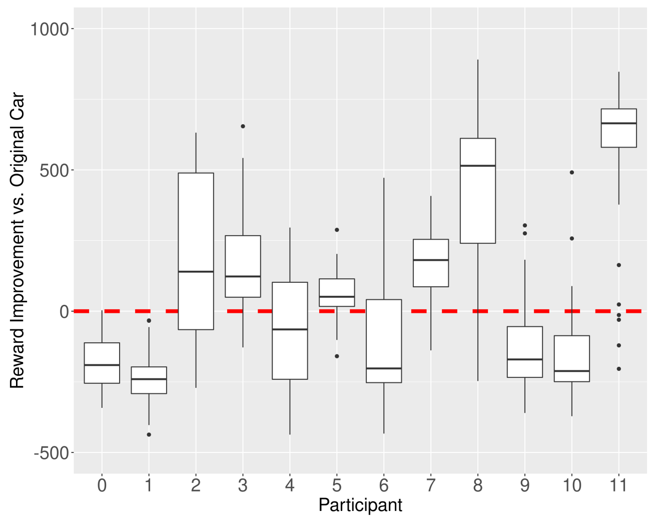
times on each of the fifty maps. We fit a linear mixed effects model on the difference between the test drive performance of each new design and the mean performance of the original design on each track. The model had random variables to account for variation due to participants and maps. The intercept of the fitted model was 56.08 with a 95% confidence interval of 6.92 to 105.24. Figure 7a shows the distribution of test-drive improvement with each design across the random maps, where improvement, again, is defined as the difference between the episode reward with a new design and the mean episode reward with the baseline car as tested on a given map. As seen in the figure, there was some diversity in how the design affected the agent's driving performance. In particular, some designs performed better than the baseline (positive improvement), while others actually tended to degrade performance with respect to the baseline car (negative improvement).

In order to measure how the designs may have influenced the agent's learning, we then retrained the agent with each human-designed car and performed test drives with the retrained agent for each design on the same fifty maps. We compared this to retraining the agent with the original car and test-driving on the fifty maps, repeating the training with the original car six times and averaging across the test drives for each map. For all of the designs, we retrained the agent for 250 episodes, using a learning rate of $\alpha = 0.01$ and a replay frequency of 20 frames. Figure 7b shows the distribution of improvements in agent performance using a redesigned car after retraining with that design versus driving the original car after retraining with the original car. Again, improvement was measured as the difference between a design's episode reward and the mean across test drives with the baseline for each map.

Fitting another linear mixed effects model on the improvement after retraining, we found an intercept of 60.01, with a 95% confidence interval of -92.04 to 212.07. When interpreting these results, it is important to note that our agent, after retraining with the baseline design, actually tended to realize a drop in performance, averaging a paltry 146.12 in episode rewards over the fifty test drive maps and six trials with the original car. With that said, this



(a) Relative performance to the original agent.



(b) Relative performances after retraining the agent.

Figure 7: These plots show the distribution of test drive rewards over a baseline for each human design over fifty random maps. The baseline is the mean episode reward for that map with the default design. The figure on the left shows results from the agent participants designed for. The figure on the right show results after retraining the agent for an additional 250 episodes with each respective design.

much larger confidence interval, spanning into negative territory, suggests that performance gains after retraining the agent on new designs seemed to vary much more with respect to the baseline performance than we observed before retraining.

Additionally, not every design had a consistent effect on the agent's performance before and after retraining with it. For example, Design 6 performed better than the baseline with the original agent, but worse than the baseline when the agent had retrained with the design. Design 11, on the other hand, was marginally better than the baseline with the original agent, but significantly better after retraining.

5.4 Case Studies With Selected Designs

In the following, we focus on four different designs in more detail. The designs are numbered according to the corresponding participant number in Figure 6 and Figures 7a and 7b. Given the small sample size, the case studies were chosen as exemplars of different performance outcomes, as described in the following subtitles.

5.4.1 Design 9: Poor Performance, Poor Learning. Design 9 performed marginally worse than the baseline before retraining and worse than the baseline after retraining. The designer described decreasing the steering sensitivity and max speed, while adjusting the wheels and increasing tire tread, with a primary intention of limiting the driver's ability to turn. This stood in stark contrast to other participants who were instead trying to *increase* the control the driver had over turning. Unlike other participants, Participant 9 was focused on trying to straighten out oscillations they observed when the driver was driving straight. As they put it, "The driver wouldn't stop driving in a zig-zag motion so I tried to inhibit its ability to turn as much as possible so that the zig-zag oscillations would steady out quickly and allow the driver to drive straight."

5.4.2 Design 6: Good Performance, Poor Learning. Design 6 performed better than the baseline before retraining but worse than the baseline after retraining. Like Participant 9, Participant 6 also reduced steering sensitivity, although their intention was not to reduce oscillations but prevent the driver from over-correcting on turns. They chose not to increase the engine power, saying, "the driver seemed not ready to handle more power". Unlike Participant 9, Participant 6 designed a smaller car body which, with smaller wheels, was intended to make the car more nimble. Participant 6 tested the most designs in our pilot, with eighty-two test drives. In describing how they finalized their design, they referred to episode rewards, stating, "My last test drive was the best of the bunch, so I submitted before I ruined it!"

5.4.3 Designs 7 and 8: Consistent Performance and Learning. Both Designs 7 and 8 performed consistently well before and after retraining the agent. Interestingly, both designers chose not to significantly modify the body of the car. As Participant 7 put it, "Since the driver is a beginner, I figured something like [shape] is much less important for seeing dramatic improvement". Both designers focused on tuning the speed and acceleration of the car, via the speed limiter and/or engine power. Participant 7 described in detail a process of first reducing the top speed until the car was in control, then increasing it until they found the boundary where the car would lose control. Both designers also balanced increases in steering sensitivity with increases in tire tread or wheel width to improve traction.

6 PILOT SESSIONS DISCUSSION

In the following, we connect observations from our pilot to our prior discussion of human designers relating to and understanding agents as users. We also discuss a potential challenge to supporting humans designing tools for agents raised by the pilot. Finally, we discuss two ways in which humans might contribute to or complement computational approaches to designing tools for agents, based on examples from the pilot.

6.1 Relating to Agents: Anthromorphism and Empathy

Reflecting on the relevance of empathy when designing for agents, we've discussed humans' tendency to anthropomorphize computer agents; however, our pilot participants mostly described our agent's behaviors in functional ways. As described above, participants tended to make observations about how the driver performed on turns or straightaways, or how it failed to recover when losing the road. Nonetheless, a few participants made remarks that implicitly ascribed human or emotional characteristics to the agent, for example calling the agent "courageous", describing its strategy as "hit the gas...and hope for the best", or referring to the agent's inability to stay on the road as a lack of effort. These characterizations evoke some of the emotions modeled in Sequiera *et al.* [54] and Jacobs *et al.* [30], namely fear, hope, and intrinsic motivation. While our agent did not incorporate any model of emotion, it would be interesting to study whether perceiving emotions in an agent that *is* emotionally driven positively influences a person's ability to understand and design for that agent.

6.2 Understanding Agent Needs and Formulating Design Goals

We were also curious how pilot participants would understand an agent's needs and goals and translate these into design goals. We saw that how a designer formulates design goals based on an agent's behavior might influence the effectiveness of the designed tool. While most participants articulated high-level goals of improving the driver's control around turns, Participant 9's focus on reducing zig-zag oscillations led them to ultimately inhibit the car's ability to turn, producing a final design that performed relatively poorly both before and after retraining the agent on the design.

Interestingly, not all participants restricted their design goals to the agent's task performance. For example, Participant 1 described changing the shape of their car to look like a wineglass because it "looks classy". Following a different course, Participant 4 described their design as low-power and energy-efficient, even choosing to paint the car green "to represent that it is environmentally friendly". It is unlikely that these goals relating to aesthetics and sustainability emanated in any way from the participants' perceptions of the agent or its behavior. However, there is a place for core human values in UCD [19], and a designer should consider what kinds of values should be brought to bear when designing for a particular agent and task.

Through our tutorial, we explicitly framed the task as designing for an agent that was learning how to drive. Unsurprisingly, several participants' design goals were shaped by their perceptions of the agent as a novice driver. For example, Participant 4 sought to create a design that would allow the driver to develop better, writing, "I found the driver is still raw, he needs slower speed to give him more tolerance so that he can practice his tactics." Others sought to create an easy-to-control car that would offset the agent's rawness, pursuing design strategies that restricted the driver's control.

More generally, a designer may need to consider whether to prioritize immediate performance or learning when designing for an agent. In our test drives, we observed that a design which performed well immediately did not always lead to better performance after retraining, or vice versa. Further, not all of the participants who referred to the agent's inexperience produced designs that performed better after training. For example, Participant 6 "tried to support the novice driver with a lower-powered, easier to control vehicle" and produced a car which performed well immediately but much worse than the baseline after retraining the agent with it. Overall, the decision to prioritize immediate performance or learning should account for the designer's impression of the driver's current tendencies and abilities, as well as the overall goals they believe that the agent is trying to achieve.

6.3 Challenges Supporting Humans Designing Tools for Agents

While we didn't explicitly solicit pain points from participants, two participants raised aspects of the platform that they struggled with when processing observations of the agent's behavior. One expressed difficulty making sense of the driver with how close-up the videos were, perhaps suggesting a trade-off with detail and the context that could be provided by a bird's eye view. Another participant told us that vast differences in the agent's driving from one test drive to the next made it hard to figure out the influence of different features on the driver. While the agent was pre-trained and fixed across test drives, the tracks were randomly generated for each test drive, and the agent's ϵ-greedy policy presented another source of randomness. The difficulty that the resulting variance caused this participant in generalizing across test drives presents a potential trade-off with showing designers diverse examples of agent behavior in different contexts. Indeed, both of these concerns point to a broader, difficult problem of exposing designers to relevant and representative examples of behavior, contexts, and information about an agent. This in itself is a design problem; in addressing it, techniques from the XRL literature like Sequiera and Gervasio's framework for interesting examples of agent behavior [53] may provide a good foundation.

6.4 Opportunities for a Human Role in Designing Tools for Agents

The impetus for this work lies in the prior success of agents that redesign their own tools as they learn to solve a task. We have conjectured that humans might be able to improve the design process through uniquely human capacities. While the scope of this pilot is too exploratory to rigorously evaluate what influence different human design behaviors have on design outcomes, we did observe anecdotal evidence for two promising directions that warrant further exploration.

One potential avenue for human designers is sample-efficiency. Participant 8, for example, was able to produce a car design that improved the agent's performance both with and without retraining despite trying out only four designs. The ability to intuitively reason about a design without brute-forcing one's way to a solution could complement an agent's ability to fine-tune through generating and evaluating many designs. There is some precedent for such a hybrid approach in the literature. For example, the Sentient Sketchbook is a game level-design tool in which humans sketch a low-resolution map idea while the system evaluates gameplay features, adds details, and generates suggestions via multiple genetic algorithms running in the background [35]. With that said, we observed great diversity in the number of designs and outcomes that our participants tested in the pilot, and it would be important to investigate more closely under what conditions humans are able to design effectively in a sample-efficient way.

Additionally, we observed multiple design approaches, suggesting that participants were framing the problem in more than one way. Indeed, different participants were able to create designs with improved performance via different combinations of features. In design tasks that allow such a diversity of focus, there might be ways that having a human frame the task could be valuable. For one, it could prove an avenue to reduce the computational complexity of design solution search, e.g. by formulating a subset of features over which to optimize. In our pilot, almost all participants only chose to modify a subset of the available design features, and specific participants constructed designs that improved performance with final modifications to as few as four or six features. Defining relevant features to search could also provide a simple way to impose specific values on a design. Ha suggests reward engineering as a way to constrain an agent's design search with an external objective, e.g. minimizing material [23]. One could alternatively imagine a human designer defining a specific engine horsepower or body shape to reflect goals relating to sustainability or aesthetics, then allowing the agent to optimize other features to improve performance.

7 LIMITATIONS AND FUTURE WORK

This work and the described pilot sessions are highly exploratory. We only tested one agent with participants, and some of the uniformity in the original agent characterizations may have been due to limited variety in the sampling of, or quirks in, the agent's behaviors. Further, our participants were sampled through convenience; the sample described was small and contained potential bias. Additionally, ChopShop is targeted at lay users and participants were not generally experts in either user-centered design or reinforcement learning. However, it would be informative to explore how designers well-versed in either of these domains would handle designing for an RL agent.

Importantly, participants only engaged in a single design session, with the agent in a fixed state. They were thus unable to interact with or observe the agent learning, limiting their ability to develop an understanding of how their design choices could affect how the agent learns to drive. Despite the practical challenges this raises, designing for learning agents should involve opportunities to interrogate and prototype with agents as they learn.

Future work should empirically evaluate hypotheses about humans designing for agents as users, e.g. testing how different feedback or elicitation methods support UCD for agents. We also hope to rigorously compare human designs and UCD design processes through ChopShop against a computional designer, with the hope of illuminating complementary strengths and weaknesses as the basis for collaboration between humans and agents to improve the design of the agents' tools. Building on this, we hope to further empirically test different modes of human-agent collaborative design for RL agents.

Beyond this, the prospect of human-driven agent-centered design raises a number of questions that warrant further exploration. For example, what role can empathy play when humans design for emotion-driven RL agents? What kinds of feedback from an agent can best support a human who designs for it, and how does this span designers with different approaches and cognitive styles? The current ChopShop platform affords a single overhead view of the car driving–how might visualizing other information, drawing from the XRL literature, influence a human's ability to understand and effectively design for different aspects of an agent, and how does this apply across different types of agents? Similarly, in what ways might human designers support different kinds of agents in different types of tasks? Finally, what are the ethical implications of exploring human design support for RL agents?

8 CONCLUSION

In this work, we propose potential benefits to introducing humans to the task of designing tools for reinforcement learning agents. We contend that treating agents as users in a user-centered design approach raises a novel set of challenges and questions. We describe a system to observe humans designing 2D cars for RL drivers in a benchmark RL task and present results and discussion from pilot sessions with twelve users. Finally, we discuss future directions to build on this work to further understand the benefits and challenges in doing agent-centered design with human and computer designers.

ACKNOWLEDGMENTS

The authors thank George Zhuang and Preston Rozwood for their contributions to the project. This work has been supported by the Civil, Mechanical and Manufacturing Innovation Program of the National Science Foundation under NSF Award No. 1635253 and 1907542.

REFERENCES

[1] Chadia Abras, Diane Maloney-Krichmar, Jenny Preece, et al. 2004. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications* 37, 4 (2004), 445–456.

[2] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17)*. JMLR.org, Sydney, NSW, Australia, 176–185.

[3] Christian Arzate Cruz and Takeo Igarashi. 2020. A Survey on interactive reinforcement learning: design principles and open challenges. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 1195–1209. https://doi.org/10.1145/3357236.3395525

[4] Akanksha Atrey, Kaleigh Clary, and David D. Jensen. 2019. Exploratory not explanatory: counterfactual analysis of saliency maps for Deep RL. *CoRR* abs/1912.05743 (2019). arXiv:1912.05743 http://arxiv.org/abs/1912.05743

[5] Cynthia L. Bennett and Daniela K. Rosner. 2019. The promise of empathy: Design, disability, and knowing the "other". In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK)

(CHI '19). Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300528
[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540
[7] Stuart K Card, Thomas P. Moran, and Alan Newell. 1983. *The psychology of human-computer interaction*. Crc Press New York, NY, USA.
[8] John M Carroll and Judith Reitman Olson. 1988. Mental models in human-computer interaction. *Handbook of human-computer interaction* (1988), 45–65.
[9] Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. 2019. Self-educated language agent with hindsight experience replay for instruction following. *CoRR* abs/1910.09451 (2019). arXiv:1910.09451 http://arxiv.org/abs/1910.09451
[10] Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel. 2014. *About face: the essentials of interaction design*. John Wiley & Sons.
[11] G De Michell and Rajesh K Gupta. 1997. Hardware/software co-design. *Proc. IEEE* 85, 3 (1997), 349–365.
[12] Ewart J De Visser, Samuel S Monfort, Ryan McKendrick, Melissa AB Smith, Patrick E McKnight, Frank Krueger, and Raja Parasuraman. 2016. Almost human: Anthropomorphism increases trust resilience in cognitive agents. *Journal of Experimental Psychology: Applied* 22, 3 (2016), 331.
[13] Shuby Deshpande, Benjamin Eysenbach, and Jeff Schneider. 2020. Interactive visualization for debugging rl. arXiv:2008.07331 [cs.LG]
[14] Daniel Dewey. 2014. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*.
[15] Kees Dorst. 2011. The core of 'design thinking'and its application. *Design Studies* 32, 6 (2011), 521–532.
[16] Paul Egan and Jonathan Cagan. 2016. Human and computational approaches for design problem-solving. In *Experimental Design Research: Approaches, Perspectives, Applications*. Springer International Publishing, Cham, 187–205. https://doi.org/10.1007/978-3-319-33781-4_11
[17] Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O. Riedl. 2019. Automated rationale generation: A technique for explainable ai and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 263–274. https://doi.org/10.1145/3301275.3302316
[18] Jodi Forlizzi and Katja Battarbee. 2004. Understanding experience in interactive systems. In *Proceedings of the 5th conference on Designing Interactive Systems*. 261–268.
[19] Batya Friedman. 1996. Value-sensitive design. *Interactions* 3, 6 (1996), 16–23.
[20] Joseph Giacomin. 2014. What is human centred design? *The Design Journal* 17, 4 (2014), 606–623.
[21] John D Gould and Clayton Lewis. 1985. Designing for usability: key principles and what designers think. *Commun. ACM* 28, 3 (1985), 300–311.
[22] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. 2018. Visualizing and Understanding Atari Agents. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 1792–1801. http://proceedings.mlr.press/v80/greydanus18a.html
[23] David Ha. 2019. Reinforcement learning for improving agent design. *Artificial Life* 25, 4 (2019), 352–365.
[24] David Ha and Jürgen Schmidhuber. 2018. World Models. *CoRR* abs/1803.10122 (2018). arXiv:1803.10122 http://arxiv.org/abs/1803.10122
[25] Dylan Hadfield-Menell, Anca D. Dragan, Pieter Abbeel, and Stuart J. Russell. 2016. Cooperative inverse reinforcement learning. *CoRR* abs/1606.03137 (2016). arXiv:1606.03137 http://arxiv.org/abs/1606.03137
[26] Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz-Rodríguez. 2021. Explainability in deep reinforcement learning. *Knowledge-Based Systems* 214 (2021), 106685. https://doi.org/10.1016/j.knosys.2020.106685
[27] Ann Heylighen and Andy Dong. 2019. To empathise or not to empathise? Empathy and its limits in design. *Design Studies* 65 (2019), 107–124.
[28] Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, et al. 2003. Technology probes: inspiring design for and with families. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 17–24.
[29] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia Sycara. 2018. Transparency and explanation in deep reinforcement learning neural networks. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (New Orleans, LA, USA) *(AIES '18)*. Association for Computing Machinery, New York, NY, USA, 144–150. https://doi.org/10.1145/3278721.3278776
[30] Elmer Jacobs, Joost Broekens, and Catholijn Jonker. 2014. Joy, distress, hope, and fear in reinforcement learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*. 1615–1616.
[31] 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.
[32] Samantha Krening and Karen M. Feigh. 2019. Effect of interaction design on the human experience with interactive reinforcement learning. In *Proceedings of the 2019 on Designing Interactive Systems Conference* (San Diego, CA, USA) *(DIS '19)*. Association for Computing Machinery, New York, NY, USA, 1089–1100. https://doi.org/10.1145/3322276.3322379
[33] Minae Kwon, Malte F Jung, and Ross A Knepper. 2016. Human expectations of social robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 463–464.
[34] Brenda Laurel. 1986. Interface as mimesis. *User centered system design: New perspectives on human-computer interaction* (1986), 67–85.
[35] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. 2013. Sentient sketchbook: computer-assisted game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*. ACM.
[36] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. 2020. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, 854–869. http://proceedings.mlr.press/v100/luck20a.html
[37] Robert P Marinier and John E Laird. 2008. Emotion-driven reinforcement learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 30. Cognitive Science Society, Nashville, TN.
[38] Christian Meurisch, Cristina A Mihale-Wilson, Adrian Hawlitschek, Florian Giger, Florian Müller, Oliver Hinz, and Max Mühlhäuser. 2020. Exploring user expectations of proactive AI systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–22.
[39] Jaroslav Michalco, Jakob Grue Simonsen, and Kasper Hornbæk. 2015. An exploration of the relation between expectations and user experience. *International Journal of Human-Computer Interaction* 31, 9 (2015), 603–617.
[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533. http://dx.doi.org/10.1038/nature14236
[41] Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J. Rezende. 2019. Towards interpretable reinforcement learning using attention augmented agents. *CoRR* abs/1906.02500 (2019). arXiv:1906.02500 http://arxiv.org/abs/1906.02500
[42] Don Norman. 2013. *The design of everyday things: Revised and expanded edition*. Basic books.
[43] Donald A Norman. 1983. Some observations on mental models. *Mental models* 7, 112 (1983), 7–14.
[44] Donald A Norman. 1991. Cognitive artifacts. *Designing interaction: Psychology at the human-computer interface* 1, 1 (1991), 17–38.
[45] Donald A Norman and Steven W Draper (Eds.). 1986. User centered system design: New perspectives on human-computer interaction. (1986).
[46] Andrés Páez. 2019. The pragmatic turn in explainable artificial intelligence (XAI). *Minds and Machines* 29, 3 (2019), 441–459.
[47] Luc Prieur. 2017. Python notebook for the Box2D Race car reinforce learning problem. https://gist.github.com/lmclupr/b35c89b2f8f81b443166e88b787b03ab.
[48] Daniele Reda, Tianxin Tao, and Michiel van de Panne. 2020. Learning to locomote: understanding how environment design matters for deep reinforcement learning. In *Motion, Interaction and Games*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. https://doi.org/10.1145/3424636.3426907
[49] Matthias Rosynski, Frank Kirchner, and Matias Valdenegro-Toro. 2020. Are gradient-based saliency maps useful in deep reinforcement learning? arXiv:2012.01281 [cs.LG]
[50] Christian Rupprecht, Cyril Ibrahim, and Christopher J. Pal. 2019. Finding and visualizing weaknesses of deep reinforcement learning agents. *CoRR* abs/1904.01318 (2019). arXiv:1904.01318 http://arxiv.org/abs/1904.01318
[51] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter. 2019. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada, 9798–9805. https://doi.org/10.1109/ICRA.2019.8793537
[52] Donald A Schön. 1992. Designing as reflective conversation with the materials of a design situation. *Knowledge-based Systems* 5, 1 (1992), 3–14.
[53] Pedro Sequeira and Melinda Gervasio. 2020. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence* 288 (2020), 103367.
[54] Pedro Sequeira, Francisco S. Melo, and Ana Paiva. 2011. Emotion-based intrinsic motivation for reinforcement learning agents. In *Affective Computing and Intelligent Interaction*, Sidney D'Mello, Arthur Graesser, Björn Schuller, and Jean-Claude Martin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 326–336.
[55] Nancy Staggers and Anthony F. Norcio. 1993. Mental models: Concepts for human-computer interaction research. *International Journal of Man-machine studies* 38, 4 (1993), 587–605.

[56] Karel Vredenburg, Ji-Ye Mao, Paul W. Smith, and Tom Carey. 2002. A survey of user-centered design practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Minneapolis, Minnesota, USA) *(CHI '02)*. Association for Computing Machinery, New York, NY, USA, 471–478. https://doi.org/10.1145/503376.503460

[57] John R Wilson and Andrew Rutherford. 1989. Mental models: Theory and application in human factors. *Human Factors* 31, 6 (1989), 617–634.

[58] Wayne H Wolf. 1994. Hardware-software co-design of embedded systems. *Proc. IEEE* 82, 7 (1994), 967–989.

[59] Peter Wright and John McCarthy. 2008. Empathy and experience in hci. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 637–646. https://doi.org/10.1145/1357054.1357156

[60] Peter Wright and John McCarthy. 2010. Experience-centered design: Designers, users, and communities in dialogue. *Synthesis Lectures on Human-Centered Informatics* 3, 1 (2010), 1–123.

[61] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. 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* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376301

[62] Ruohan Zhang, Bo Liu, Yifeng Zhu, Sihang Guo, Mary Hayhoe, Dana Ballard, and Peter Stone. 2020. Human versus machine attention in deep reinforcement learning tasks. arXiv:2010.15942 [cs.LG]