Finding AI's Faults with AAR/AI: An Empirical Study

ROLI KHANNA, JONATHAN DODGE, ANDREW ANDERSON, RUPIKA DIKKALA, JED IRVINE, ZEYAD SHUREIH, KIN-HO LAM, CALEB R. MATTHEWS, ZHENGXIAN LIN, MINSUK KAHNG, ALAN FERN, and MARGARET BURNETT, Oregon State University, USA

Would you allow this AI agent to make decisions on your behalf? If the answer is "not always", the next question becomes "in what circumstances"? Answering this question requires human users to be able to assess an AI agent—and not just with overall pass/fail assessments or statistics. Here users need to be able to *localize* an agent's bugs, so that they can determine when they are willing to rely on the agent and when they are not. After-Action Review for AI (AAR/AI), a new AI assessment process for integration with Explainable AI systems, aims to support human users in this endeavor, and in this paper, we empirically investigate AAR/AI's effectiveness with domain-knowledgeable users. Our results show that AAR/AI participants not only located significantly *more* bugs than non-AAR/AI participants did (i.e., showed greater recall), they also located them more *precisely* (i.e., with greater precision). In fact, AAR/AI participants outperformed non-AAR/AI participants on every bug and were, on average, almost 6 times as likely as non-AAR/AI participants to find any particular bug. Finally, evidence suggests that incorporating labeling into the AAR/AI process may encourage domain-knowledgeable users to abstract above individual instances of bugs; we hypothesize that doing so may have contributed further to AAR/AI participants' effectiveness.

CCS Concepts: • **Human-centered computing → Empirical studies in HCI**.

Additional Key Words and Phrases: AAR/AI, After-Action Review for AI, Explainable AI (XAI)

ACM Reference Format:
Roli Khanna, Jonathan Dodge, Andrew Anderson, Rupika Dikkala, Jed Irvine, Zeyad Shureih, Kin-Ho Lam, Caleb R. Matthews, Zhengxian Lin, Minsuk Kahng, Alan Fern, and Margaret Burnett. 2021. Finding AI's Faults with AAR/AI: An Empirical Study . *ACM Trans. Interact. Intell. Syst.* 99, 99, Article 99 (2021), 34 pages. https://doi.org/99.9999/9999999.9999999

1 INTRODUCTION

Explainable AI (XAI) has recently begun to expand its scope. Besides simply explaining AI to its users, some XAI researchers are focusing on explanation-based systems to help users *assess* an AI system's decisions (e.g., [12, 23, 35, 38, 48, 60, 61, 70]).

Imagine "Pat," a user knowledgeable in some domain who is trying to make an educated decision about whether or when to rely upon a particular intelligent agent in a *particular* situation important to them. In the domain of AI-powered aviation, Pat might be a test pilot, deciding on the circumstances in which a human pilot should override AI-generated decisions in a new AI-powered airplane. In the domain of health care, Pat might be a remote aging-in-place caregiver, deciding

Authors' address: Roli Khanna, khannaro@oregonstate.edu; Jonathan Dodge, dodgej@oregonstate.edu; Andrew Anderson, anderan2@oregonstate.edu; Rupika Dikkala, dikkalar@oregonstate.edu; Jed Irvine, irvine@eecs.oregonstate.edu; Zeyad Shureih, shureihz@oregonstate.edu; Kin-Ho Lam, lamki@oregonstate.edu; Caleb R. Matthews, mattheca@oregonstate.edu; Zhengxian Lin, linzhe@oregonstate.edu; Minsuk Kahng, minsuk.kahng@oregonstate.edu; Alan Fern, Alan.Fern@oregonstate.edu; Margaret Burnett, Oregon State University, Corvallis, OR, USA, burnett@eecs.oregonstate.edu.

2160-6455/2021/99-ART99 $15.00
https://doi.org/99.9999/9999999.9999999

when to follow an AI system's recommendations on actions their grandparent may need from them right away. These particular situations matter to Pat and to the person or people Pat affects. No matter how thoroughly trained an AI system is, for Pat the dilemma is not about the AI system's overall correctness statistics—it is about their responsibility for making the most appropriate decision for this particular case. The European Commission's European Group on Ethics in Science and New Technologies put it this way: *"[autonomous systems] must not impair [the] freedom of human beings to set their own standards and norms and be able to live according to them"* [21, 50].

In assessing domains like these, no objective "ground truth" is available, because (1) only Pat knows *this* airplane's or *their* grandparent's behaviors in the context in which they happened, and (2) there may be multiple response paths Pat could take that would produce positive outcomes[1].

To enable domain experts such as Pat to assess an AI agent and find its faults in circumstances like these, we devised AAR/AI (After-Action Review for AI) [19, 46], which we detail in Section 1.3. AAR/AI aims to help domain experts assess AI agents in sequential decision making environments. This paper evaluates how well AAR/AI achieves this aim.

1.1 The Domain

Real-Time Strategy (RTS) games are a popular sequential decision making domain for AI research. In RTS games, players attempt to strategically maneuver through a plethora of choices to win the game. Sometimes in AI research, the RTS player is an AI agent maneuvering on behalf of a human[2], which is the case in this paper.

For our study, we used a model-based reinforcement learning (RL) agent that played an RTS game. The game was the same StarCraft 2 "Tug-of-War" custom game as was used in earlier AAR/AI

[1]Groce et al. also pointed out [23] that no objective ground truth exists in many human/AI decision domains—there is only "good enough for my purposes".

[2]e.g., as a proxy for dangerous strategy-centered situations such as military operations.

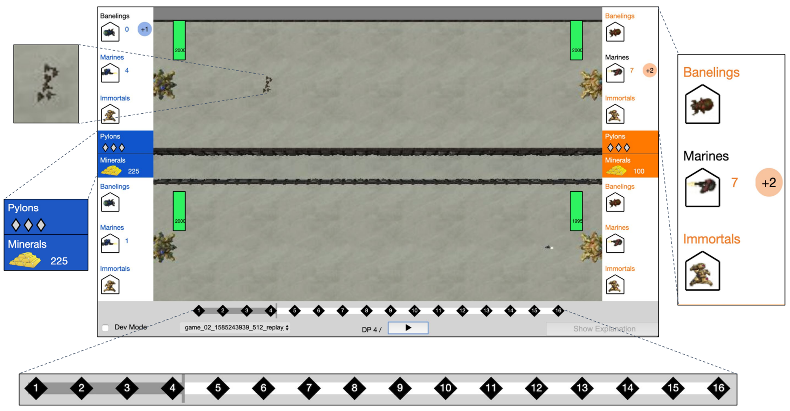


Fig. 1. The participants' replay view of the game just past Decision Point 4 (fourth diamond; see callout at bottom). The game board has two lanes where action takes place: a top lane and a bottom lane. The (blue) friendly AI agent's "home" is the left side, and the (orange) enemy is the right side. Each lane's troop inventories are shown in a side panel for that lane; e.g., the callout at right blows up the side panel for the enemy's top lane. Both players' side panels also summarize resources; e.g., the blue callout (middle left) shows the friendly AI's resources. On the game board, a group of friendly AI marines in the top lane are currently moving toward the enemy's Nexus (top left callout).

publications [19, 46]. Tug-of-War games entail two evenly matched players, a *Friendly AI* and *Enemy AI* pursuing the same goal. In our game, tugs of war occur in the top and bottom "lanes" of a game (Figure 1), over the course of a maximum of 40 Decision Points (DPs) or rounds. Players perform actions in either lane at each DP, depending on affordability (e.g., how much they spent and earned prior to the current decision point). Actions include purchasing troop production buildings and/or purchasing Pylons to increase income. Figure 1 shows a screenshot of the game replay as it appeared to our study's participants.

In the game, troops battle to win, with troop types in a rock-paper-scissors relationship: Marines are effective against Immortals; Immortals against Banelings; Banelings against Marines. Different troops cost different amounts, depending on these capabilities. Troops spawn behind the Nexus (the player's base, represented as gold star-shaped objects on the gameboard in Figure 1). Once spawned, they march down the lane and attack enemies in pre-programmed fashion, as with the marines shown in Figure 1. Players can win in 2 ways: (1) destroy one of its opponent's Nexuses, or (2) if all the Nexuses remain after 40 DPs, the player whose Nexus has the lowest health loses.

1.2 An AI RTS player's failures and faults

What if an AI agent, such as the one playing this game, makes a flawed decision? An AI agent's flawed decisions are analogous to the software engineering concept of "failures". Ammann and Offutt define a "failure" as "...external, incorrect behavior with respect to the requirements..." [3]. In the RTS domain, our analogous requirement is the AI agent deciding upon good "enough" actions (according to a human knowledgeable in the domain), so we define failures as user-visible decisions the AI agent makes that are not adequate[3] according to that particular user's standards.

Still, a failure is only a symptom of something going wrong under the hood. Ideally, an interactive XAI system could not only help Pat spot such symptoms, but also locate the root causes of those symptoms. Only in this way can Pat know which decisions the AI is making for acceptable reasons, so as to avoid "lucky guesses", ward off ethical concerns, or defend against potential legal challenges (e.g., a malpractice suit) [21, 37].

In medicine, the causes of symptoms are diseases; in software engineering literature, the causes of symptoms (failures) are termed "faults". Avizienis et al. [6] define a fault to be the underlying cause or condition that may lead to a failure; and "fault localization" to be the act of identifying the *locations* of faults. Building upon these definitions, in an RTS game with XAI support, we define a fault to be erroneous reasoning by the AI agent—ideally revealed to the users in the explanations—and fault localization to be finding the component of the explanation that reveals the erroneous reasoning[4]. In this paper, we also use the term "bug" synonymously with "fault".

1.3 Human users assessing an AI RTS player's failures and bugs with/without AAR/AI

Finding an AI's failures and bugs involves users understanding the AI's behaviors. However, numerous empirical studies have reported difficulties users face in building this understanding, even in the presence of explanations (e.g., [5, 14, 40, 60]). One possibility is that presenting explanations to users is not enough—because explanations unscaffolded by a *process* require the user to not only build their understanding, but also to build their own process for doing so.

In this paper, we empirically investigate whether a process known as After-Action Review for AI (AAR/AI) [19, 46] can improve XAI users' ability to understand an AI agent well enough to localize its bugs. AAR/AI is a new member of the After-Action Reviews (AAR) family. After-Action

[3]As with Ammann/Offutt's definition, an AI agent's failure is not always a "show-stopper". That is, a bad decision is a failure even if the AI agent later makes good enough decisions to overcome the initial bad decision.

[4]In medicine, identifying the disease is a necessary step toward a cure, but still may not be sufficient to produce an effective cure. Similarly in software engineering, localizing a fault is necessary but still may not be sufficient to produce a fix.

Reviews were originally devised by the U.S. Army [64] for assessing human decisions. The AAR is a facilitated, team-based debriefing method. In the military, it has been used to assess solider training sessions. Sawyer and Deering characterize the AAR process using the acronym "DEBRIEF": Define rules, Explain objectives, Benchmark performance, Review what was supposed to happen, Identify what happened, Examine why, and Formalize learning [59]. More generally, AAR has been used for decades to assess human decisions in the military (e.g. [24]), and has also been adapted to manned-unmanned teams [9]. Civilians have also used AAR processes in transportation [45], medical treatment [54, 59] and emergency response [18, 30, 42]. A recent meta-analysis of 61 studies by Keiser et al. [33] found that using AAR produced beneficial and practical effects.

AAR/AI is the first use of AAR in AI. The original AAR/AI publication [46] describes AAR/AI in 7 steps, adapted from the DEBRIEF sequence above. These steps are conducted with a Facilitator and one or more Assessors as follows:

(1) The Facilitator defines the domain.
(2) The Facilitator explains the agent's objective.
Next begins an "inner loop" for each decision to be assessed:
(3) The Facilitator reviews what was supposed to happen.
(4) An Assessor identifies what happened.
(5) An Assessor describes why it happened.
(6) An Assessor formalizes learning from this decision.
Finally, at the end of these iterations:
(7) An Assessor formalizes learning holistically from every decision they analyzed.

The AAR/AI process allows flexibility in the details within each step, to allow customization to the assessors' purpose in their domain.

Although there is some qualitative evidence revealing some of AAR/AI's strengths [19, 46], AAR/AI has not been empirically compared with *not* using AAR/AI. Only a comparison of with-AAR/AI vs. without-AAR/AI can measure causality—whether using AAR/AI leads human users to significantly greater effectiveness at assessing an AI system than they would achieve without AAR/AI. To address this need, we prototyped an AAR/AI-supported XAI system (which will be illustrated in Section 3.2), and used it to conduct a controlled lab experiment comparing the effectiveness of domain-knowledgeable users localizing faults (bugs) using AAR/AI versus without AAR/AI. In both treatments, participants could use information in the game itself and a full explanation of the AI system's reasoning.

Our study investigated the following research questions:

RQ1 Does the AAR/AI process help domain-knowledgeable users to localize faults in an XAI-based system?
RQ2 Does the type of fault interact with **RQ1**?
RQ3 When supported by AAR/AI, do users somehow abstract beyond individual instances of faults? If so, how?

2 BACKGROUND & RELATED WORK

A substantial body of research has investigated human users *understanding*, *finding* (e.g., through testing), and/or *debugging/improving* AI systems, all of which relate to humans localizing an AI agent's faults.

Fault localization in an AI agent requires the human doing the localizing to have at least a partial understanding of how the AI agent reasons. Explainable AI (XAI) aims at exactly this goal. One of its aims is to improve people's mental models [4, 5, 38, 39]—representations people construct in their heads about how something works from whatever they have experienced with it [49].

However, affecting someone's mental model is not always straightforward. Although people have mental models about most things, their mental models are not always accurate and sometimes are not be very malleable. For example, Tullio et al. found explanations helped clarify some misconceptions, but overall mental model structure went largely unchanged [63]. This exposes a central challenge XAI faces when trying to help people understand an AI system, which Yang et al. describe as "[humans'] uncertainty surrounding AI's capabilities... [and]... AI's output complexity" [69].

To address this problem, some XAI researchers have drawn from social science the strategy of helping humans generate "self-explanations," a process which has been shown to support knowledge acquisition [56]. A user might generate self-explanations as a result of being prompted to do so, or might do so on their own accord [26].

As an example of XAI work involving self-explanations, Chi et al. showed that learners relying more heavily on examples had worse outcomes [13], which they credited to inability to engage in self-explanation. Another example occurred in our early qualitative AAR/AI results [46], in which participants' uses of self-explanation, in combination with other factors, produced high levels in Bloom's learning taxonomy [8]. Still another example of encouraging self-explanations is the use of counterfactuals; Byrne offers evidence that counterfactuals enable people to explain how events relate to one another such as identifying cause-effect or reason-action relationships [10].

XAI consumers correspond to human learners, in that the humans consuming XAI are doing so to learn how the AI reasoning went. This correspondence opens the possibility of drawing from Cognitive Load Theory (CLT) for insights. CLT models tasks as having three kinds of load: intrinsic ("nature of the material"), extraneous ("manner in which the material is presented"), and germane load ("reflects the effort that contributes to the construction of schemas.") [62]. The recommendation of much of CLT work is to increase germane load and decrease extraneous load where possible.

Where can XAI researchers and developers turn to find concrete XAI-pertinent guidance to fulfill recommendations like these? For non-AI systems, when faced with the challenge of helping people form more accurate mental models, UI designers can draw upon substantial work distilling research results into usability fundamentals and practical guidelines. Unfortunately, however, few such works yet exist for XAI. Although Hoffman et al. [27] recently conducted a large-scale literature survey on guidance for empiricists *measuring* XAI's effects, explanation design was not covered. In a very large-scale literature survey by Abdul et al. on XAI with an HCI (human-computer interaction) perspective, none of the usability papers reported were tailored for XAI [1]. Soon after Abdul et al.'s paper, Amershi et al. [2] created 18 usability-centric guidelines for human-AI interaction. A few of these guidelines are applicable to XAI, but as one of the first works in the direction of usability guidelines in AI, the guidelines mainly contribute a set of design goals to achieve for usable AI, not how to achieve them. For example, Guideline 11 is "Make clear why the system did what it did," which is an important design goal, but is not guidance on how to do so. Complementing Amershi's work, Wang et al. presented a theory-centric framework connecting social science fundamentals on human reasoning and human biases to XAI techniques [65]. This work produced six XAI lessons learned from the social science research. These six, like Amershi's 18, are at the design goal level (e.g., "support hypothesis generation"), but unlike Amershi's 18, these six also drill down one more level of theory. For example, one recommendation is to support hypothesis generation via contrastive reasoning, hypothetico-deductive reasoning, and abductive reasoning. However, for XAI researchers not adept with social science concepts on how these kinds of reasoning work in humans, more concrete operationalizations may still be needed.

Given the paucity of XAI-specific usability fundamentals, many researchers have turned to advancing community knowledge through empirical studies. Taxonomies and related sets of principles are ways to build upon these researchers' individual empirical results—they abstract above individual experiments, thus providing intellectual tools for understanding the dimensions of XAI that researchers have been investigating.

For example, Kulesza et al. [38, 40], taxonomize XAI research via on two proposed principles—soundness and completeness—illustrated in the phrase, "the whole truth (completeness) and nothing but the truth (soundness)". Understanding explanations' attributes according to these principles have implications for XAI's consumers. For example, when completeness is too low, explanation consumers may perceive it as "sneaking," a UI dark pattern adapted to XAI [15]. However, if completeness is too high, users' searches for "the right" information can so become onerous that finding failures or localizing faults in an explanation may be reminiscent of finding the proverbial needle in a haystack.

In their "intelligibility types" taxonomy, Lim and Dey categorized explanations according to the kinds of questions they answer (e.g. What, Why, etc) and its relationship to the system (e.g. Inputs, Model, Outputs) [43, 44]. The relative importance of each intelligibility types can vary by domain. For example, Lim and Dey found that users wanted Why Not information when they perceived flaws, whereas other researchers found a heavy emphasis on What information in domains like smart homes and Real-Time Strategy (RTS) games [11, 52]. Research has reported that supporting the "right" intelligibility types for a particular situation or domain improved users' confidence in the system [17].

Although most current XAI research focuses on helping people interpret models' inner workings (e.g. [28, 66]), some tools in the closely related area of interactive ML are intended for failure detection and/or fault localization. Examples include using scalable query-based approaches for NLP [68], clustering around user-selected example-based "anchors" [12], or "covering" different input/output combinations [23]. Others support fault localization ("visual debugging" [61]) by revealing system internals—in this case, latent vectors for sequence-to-sequence models for translation. The explanations in our study also include revealing system internals, but in a model-based agent's search tree.

For this paper, the domain is a complex, sequential decision-making environment based on Real-Time Strategy (RTS) games. Ontañón et al. has pointed to the gap in research about human needs for understanding AI for RTS [51], and researchers have been working to fill this gap. As a few examples, Metoyer et al. contributed formative work via human explanations of RTS games used within expert-novice pairs [47], Dodge and Penney investigated how expert broadcasters explain RTS [20, 52], Kim et al. investigated human responses to Human vs. AI battles [34, 35], and Penney et al. investigated pairs of AI players making sense of "simulated AI" behavior [52, 53]. Although some of these RTS investigations included participants *noticing AI failures* (symptoms of faults), none except the AAR/AI work [46] offer insights into humans attempting to *localize AI faults* in this domain. To help fill this gap, in this paper we present the first quantitative evaluation of humans' effectiveness with the AAR/AI process.

3 METHODOLOGY

To investigate the effectiveness of the AAR/AI process for localizing AI's faults/bugs, we conducted an empirical study with domain-knowledgeable participants using AAR/AI vs. without AAR/AI. Due to COVID-19, we conducted sessions over teleconference (Zoom) and a browser-based custom combination of the platform (game and explanation system, including AAR/AI features for the AAR/AI treatment) and questionnaires. The participants were experienced with RTS games but had no AI or machine learning (ML) background.

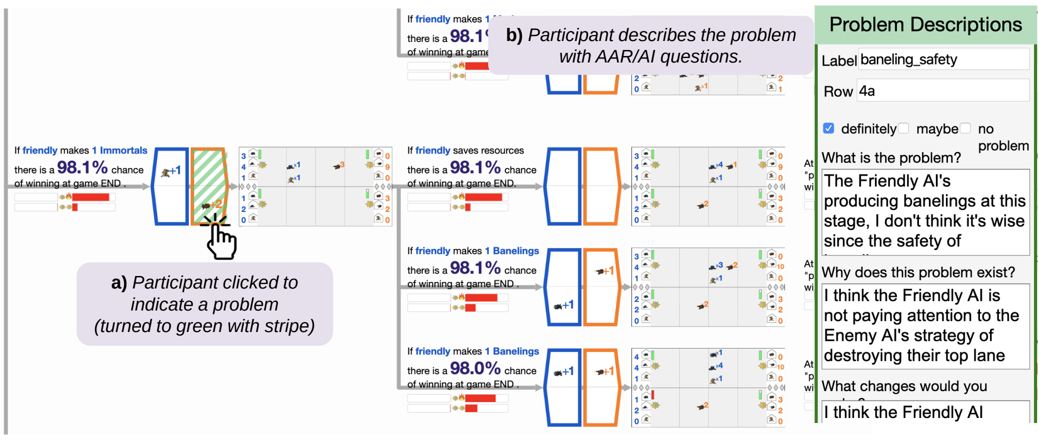
3.1 Participants and Procedure

We required participants to be at least 18 years of age, and to have 10+ hours of prior experience with real-time strategy (RTS) games to ensure they would understand our domain. In addition, we excluded respondents who had taken any AI or ML class before. (We later disqualified one participant who became persistently inattentive during the study session.) Of the final 65 participants, 49 self-identified as men, 15 as women, and 1 as transgender (Table 1). Participants were randomly assigned by flipping a coin to one of two treatments: AAR/AI and non-AAR/AI. Each zoom session had one to seven participants. Upon completing the study, they received a $20 Amazon gift card as compensation.

All participants observed an AI agent playing the web-based RTS game described in Section 1.1. Participants' task was to localize the AI agent's bugs. Participants in both the treatments saw the same explanation (which we describe in Section 3.2)—the only difference between the treatments

	AAR/AI	**Non-AAR/AI**	Total
Man	25	24	49
Woman	7	8	15
Transgender	1	0	1
Undergrad	13	14	27
Grad	10	7	17
Non-student	10	11	21
Total	33	32	65

Table 1. Participants demographics as per their questionnaire responses to their gender identification (including a free-form response), student/non-student status, and age. Median age was 24 (minimum: 17; maximum: 48), with about half below and half above. (One 17-year-old claimed to meet the >=18 inclusion criterion before the study, then gave their actual age on the questionnaire.) AAR/AI vs. non-AAR/AI participant demographics were similar for all categories.


If friendly makes 1 Immortals
there is a 98.1% chance of winning at game END .
If friendly saves resources
there is a 98.1% chance of winning at game END.
If friendly makes 1 Banelings
there is a 98.1% chance of winning at game END .
If friendly makes 1 Banelings
there is a 98.0% chance of winning at game END .
a) Participant clicked to indicate a problem (turned to green with stripe)
b) Participant describes the problem with AAR/AI questions.

Problem Descriptions

Label baneling_safety

Row 4a

[✓ definitely] maybe no problem

What is the problem?

The Friendly AI's producing banelings at this stage, I don't think it's wise since the safety of

Why does this problem exist?

I think the Friendly AI is not paying attention to the Enemy AI's strategy of destroying their top lane

What changes would you

I think the Friendly AI

Fig. 2. How a (hypothetical) participant could mark up the Explanation UI for the AAR/AI Treatment. (a) Participant selects what they think is a problem on the diagram. (b) Participant describes the problem by including a label, location, level of certainty, and responses to the What, Why, and What changes questions.

was the presence/absence of the AAR/AI supports. Data collected were participants' responses to a pre-task demographic questionnaire; their in-task answers to the AI agent's reasoning, and where in the explanation they saw these problems (Figure 2); a click log of their interactions; and their responses to a post-task NASA/TLX questionnaire [25]. Additional details about the study procedures can be found in Appendix B, and all questionnaires are included in the Supplemental Documents accompanying this paper.

The study proceeded as follows. The participants agreed to an IRB-approved informed consent form, then filled out the pre-task demographic questionnaire, then performed Steps 1-3 below (also illustrated in Figure 3), and finally filled out the NASA/TLX questionnaire and were compensated.

Step 1: Tutorial: The researcher began the tutorial by informing participants that 1) they would observe a game between Friendly and Enemy AI players, 2) the Friendly AI would lose, and 3) their main task was to find "problems" in the Friendly AI's actions. ("Problems" was the vocabulary we used with participants to encourage them to find any/all of the Friendly AI agent's failures and faults/bugs as defined in Section 1.)

The researcher guided the participants through working with the interface to familiarize them with the game interface and explanations for 30-40 minutes. The tutorial included example problems, such as "violation of game rules for buying troops in both the top and bottom lanes", but ultimately they were told that, "If you think it's a problem, it's a problem.". This set-up and tutorial handled the first two steps of AAR/AI: (1) defining the rules and (2) explaining the agent's objective.

Step 2: Entering the game interface: After the hands-on tutorial, the participants got access to the main task's interface (Figure 4A), and they began watching a sped-up replay of a game with the Friendly AI competing against the Enemy AI. Participants could only observe the game; they did not play the game.

From here onward, the researcher had real-time access to the actions taken by all the participants through a Dashboard. This ensured that the researcher could track signs of inattention or inappropriate actions taken by the participants, and deal directly with the participant about them. (One participant's inattention could not be resolved, and we ultimately discarded their data; this is in addition to the 65 participants reported in this paper.)

Step 3: Main task loop: (Predict and Describe with Replay; Locate and Describe with Explanation).

Predict and Describe with Replay: The game automatically paused at a Decision Point (DP) *before* the one they would analyze, and participants provided written answers to what they thought the

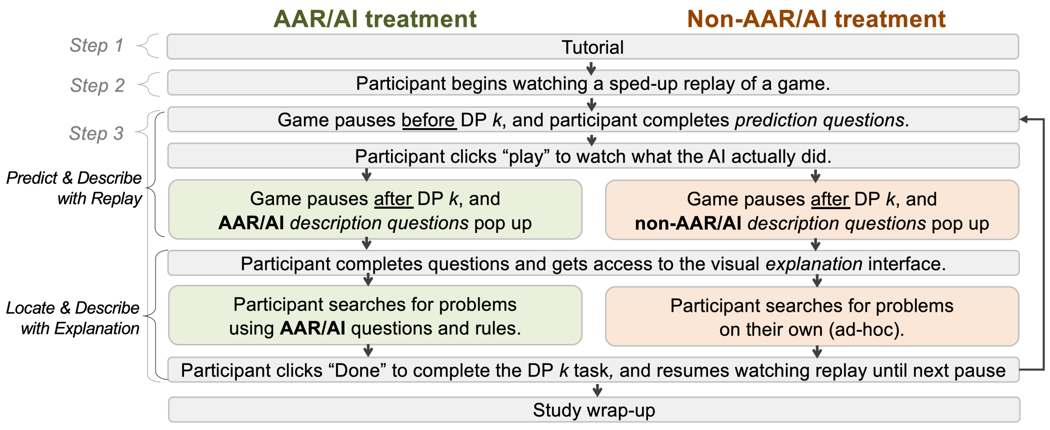

AAR/AI treatment
Non-AAR/AI treatment
Step 1
Tutorial
Step 2
Participant begins watching a sped-up replay of a game.
Step 3
Game pauses before DP k, and participant completes prediction questions.
Predict & Describe with Replay
Participant clicks "play" to watch what the AI actually did.
Game pauses after DP k, and AAR/AI description questions pop up
Game pauses after DP k, and non-AAR/AI description questions pop up
Participant completes questions and gets access to the visual explanation interface.
Locate & Describe with Explanation
Participant searches for problems using AAR/AI questions and rules.
Participant searches for problems on their own (ad-hoc).
Participant clicks "Done" to complete the DP k task, and resumes watching replay until next pause
Study wrap-up

Fig. 3. Summary of study procedure.

Friendly AI would do by the *next* DP. Specifically, participants said which lane it would build in, and whether it would make any marines, any banelings, any immortals, and/or a pylon. The purpose of these questions was to get the participant active in trying to figure out the AI's reasoning.

The participants then watched the AI's decision and answered a set of questions. The participants in the non-AAR/AI group answered a question asking what the Friendly AI had just done; and the AAR/AI participants answered three questions as part of the AAR/AI process: *what* had the Friendly AI just done; *why* they thought it made those decisions; and *what changes* they would make in the Friendly AI's decisions.

Locate and Describe with Explanation: After they had answered the initial questions, participants were able to see the explanations (Figure 4B). Participants were then told to locate problems in the AI's reasoning using the explanations by clicking parts of the explanation they found problematic, and answering a set of questions. AAR/AI participants also had the option of labeling the problems they found via a free-form textbox[5]. We captured all their interactions in a click log.

AAR/AI participants could start at any row they wanted, but once they had started a row, they had to finish locating bugs in that row and describing them via the AAR/AI questions as in Figure 2. Once they said they were finished with the row (by clicking on "Done with this row"), they moved on to whatever next row they wanted. (They could later go back to review any previous row, but they could not change it after they had said they had finished it.) In contrast, non-AAR/AI participants could move freely among rows, tackling the task of locating and describing them however they pleased.

Once participants completed finding and describing problems in one DP, they then could click on the "Done" button to indicate the completion of the task. They then could resume watching the game. The game would pause again at another DP, and participants repeated the same process as above in this new DP. The two decisions points participants worked with were DP 8 and DP 15, selected for the bugs they exhibited. Participants could spend a minimum of 10 minutes and a maximum of 40 minutes per DP.

3.2 Explanations

Figure 4B shows a visual explanation of the agent for a given decision point (DP). It visualizes the internal search tree the agent made to find the best actions. The leftmost node (also shown in Figure 4B-1) graphically represents a current state of the game (i.e., root of the search tree). Note that the state in Figure 4B-1 is an approximate thumbnail of the gameboard (Figure 4A). The tree expansion to the right of the current state shows different combinations of actions and states the agent predicts could happen next. Figure 4B-2 shows the Friendly AI's action in the blue box and the Enemy's action in orange. Next is the predicted "next state" (child), followed by another pair of actions, and the predicted grandchild state. The explanation interface shows 5 of the 20 searched actions: the top 2, median, and bottom 2 (Figure 4B).

Outcome predictions, shown in Figure 4B-3, appear in two ways: a sentence describing the win probability associated with that action and a visualization decomposing that win probability into 4 stacked bars, one for each nexus. Each bar shows two probabilities: one for the nexus being destroyed (shown in red) and the other for that nexus having the lowest health at the end of a game (shown in pink). The sum of all 8 probabilities is 100% (the game has to end in one of 8 ways); thus, a single player's win probability is that sum minus the 4 probabilities that they lose (encoded by the total size of the red and pink bars on the right side), shown by the large bold number.

[5]In formative investigations and pilots, we observed that one benefit of AAR/AI seemed to be its encouragement of consistent search practices [7]. We added labeling to AAR/AI to further encourage these practices.

3.3 The Reinforcement Learning Agent

In our study, the Friendly AI "player" is powered by a model-based reinforcement learning agent[6], which determines the Friendly AI's next action by predicting future states. This section summarizes the agent, and Appendix C explains its architecture in more detail.

The agent makes its predictions using the following neural-network driven functions:

(1) Action-Ranking Function (ARF) with type signature `(State, Action)` $\longrightarrow$ `float`: answers "How good is taking this Action in this State?"
(2) Transition Function (TF) with type signature `(State, Action1, Action2)` $\longrightarrow$ `State`: answers "What State will arise if I (Friendly AI) take Action1 and the opponent (Enemy AI) takes Action2?"
(3) Leaf Evaluation Function (LEF) with type signature `(State)` $\longrightarrow$ `(outcome-probabilities)`: answers "How good is this state?"

[6]This agent was also used in Mai et al [46].

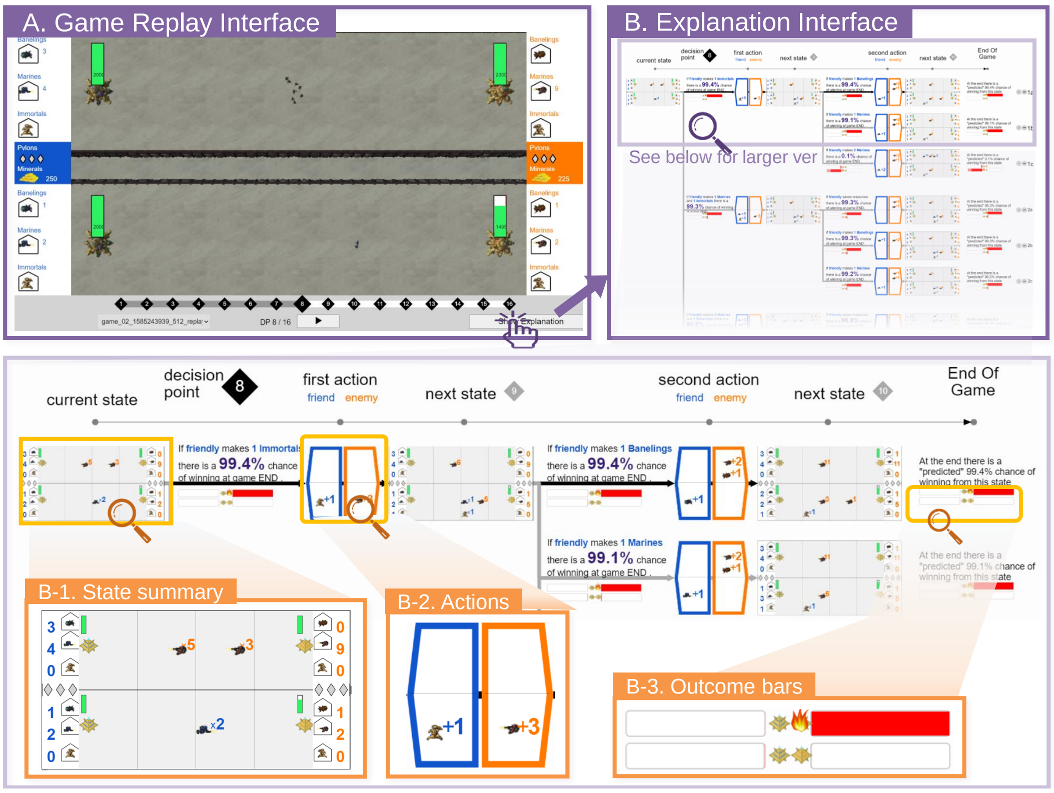

A. Game Replay Interface
B. Explanation Interface
See below for larger ver
Show Explanation
DP 8 / 16
current state
decision point
8
first action
friend enemy
next state
second action
friend enemy
next state
End Of Game
If friendly makes 1 Immortals there is a 99.4% chance of winning at game END.
If friendly makes 1 Banelings there is a 99.4% chance of winning at game END.
If friendly makes 1 Marines there is a 99.1% chance of winning at game END.
At the end there is a "predicted" 99.4% chance of winning from this state
At the end there is a "predicted" 99.1% chance of winning from this state
B-1. State summary
B-2. Actions
B-3. Outcome bars

Fig. 4. **A.** The game interface that participants used to watch the game in action; **B.** This interface visually explains Friendly AI's explanation for its actions. The screenshot shows top 2 next actions. The top row represents the best next action among multiple actions: **B-1.** Current state is graphically represented; **B-2.** AI's predicted action pair (Friendly in blue and Enemy in orange). Also, its child state, the AI's next predicted action pair, and grandchild state are shown. **B-3.** At the right, the outcome bars are shown to represent how the probability is calculated.

Using these components, the agent internally builds a search tree to select the best action for the Friendly AI player. It first enumerates all actions available in the current state, applying the Action Ranking Function (ARF) to each, and pruning all but the top 20 actions (shown in blue at Figure 4B). It then applies the ARF again from the opponent's perspective, pruning all but the top 10 actions (the top one shown in orange next to the blue one at Figure 4B). Then for each combination of promising moves, the agent applies the Transition Function (TF), predicting the resultant child state. By this point one level of the game tree has been built. It then builds one more level in the same fashion, starting from the child state, with smaller numbers of actions. Although we could repeat this process indefinitely, we stop the prediction here, because in many domains searching enough to reach a terminal state would be intractable. Thus, after the agent applies the Leaf Evaluation Function (LEF) to each grandchild state, it propagates resulting values back up the tree via minimax search.[7]

3.4 The Bugs

The participants' task was to identify the AI agent's bugs. We based our bugs on the agent's naturally occurring bugs. For example, one of the bugs we found was that the AI predicted that a health value of the nexus increases over time, which cannot occur in a real game.

To harvest these bugs, we wrote scripts to find similar cases that were objectively wrong (such as ones that violate game rules, like the nexus health example above), and then hand-validated the results. After rigorous analysis [41] of these naturally occurring bugs and iteratively trying them in our preliminary pilots, we harvested 10 of these bug instances, selected DPs that contained those bugs, and exaggerated those whose effects were too small to notice easily. Appendix B's Table 6 enumerates the complete list of bugs and any exaggerations we made. All participants' explanations contained all of the bugs enumerated. Five bug instances were in each of two decision points (DP). Half of the bugs were in the agent's Transition Function (TF) and half were in the agent's Leaf Evaluation Function (LEF).

For example, one of the Leaf Evaluation Function (LEF) bugs is Bug ID #1 at DP 8; this bug is shown in detail later in Figure 7. At one point (which will be called out in row 1C in the figure), the agent predicts that the Friendly AI will *lose* with its *bottom* nexus being destroyed, but it consistently predicts that the Friendly AI will *win* by destroying the enemy AI's *top* lane nexus in other rows. Also, although the actions in 1B and 1C are very similar, their outcomes are radically different, which is not possible. This is a bug with the win probabilities flipped for both the Friendly and Enemy AI's top and bottom lanes.

An example of a Transition Function (TF) bug is Bug ID #4 at DP 8, which is shown later in Figure 8. The agent predicts that the enemy AI will have two immortals in the next state; however, it is not possible because there exists only one building for producing immortals and each building can produce only one in a single round.

4 RESULTS

4.1 RQ1 Results: Does AAR/AI help localize faults?

RQ1 asks whether the AAR/AI process helps domain-knowledgeable users localize faults. To answer this question, we measured participants' ability to find and describe the 10 bugs enumerated in Section 3.4. To code their efforts, two researchers independently coded 20% of the data corpus, and achieved an inter-rater reliability (IRR) of 80.6% (Jaccard index [31]). Given this level of agreement, they then split up the remaining coding. The code set followed a scoring system. Participants could earn up to 2 points for each bug they reported: if they correctly *located* a bug, they could earn up

[7] A full discussion on minimax and game tree search can be found in Chapter 5 in [58].

to 1 point, and if they correctly *described* the bug, they could earn another point. Table 2 details the coding rules for no credit, partial, or full credit for locating and describing bugs.

First, we compare sheer volume of AAR/AI vs. non-AAR/AI participants' problem reports. Figure 5 shows the distributions of how many problems participants reported. AAR/AI participants reported significantly more problems than Non-AAR/AI participants (t-test, t(63) = 5.7829, p < .0001)[8]. In fact, even the AAR/AI participant with the fewest problem reports (9) still submitted more than 75% of the participants in the Non-AAR/AI treatment did (Q3 = 8.25).

To evaluate the correctness of their problem reports, we use the term "bug" to refer to the bugs in Table 6, and the term "problem" to denote whatever participants reported as problematic. We use these concepts to compute two metrics commonly used in machine learning – recall and precision[9]. Recall measures the proportion of the system's 10 bugs the participants reported, and precision measures the proportion of participants' problem reports that were actually bugs. An "ideal" participant whose problem reports would show perfect recall and precision would include all of the bugs in Table 6 (perfect recall) and nothing else (perfect precision).

Using these measures, the AAR/AI participants had both significantly greater average recall (Welch's t-test, t(55.666) = 4.5479, p < .0001) and precision (t-test, t(63) = 2.0358, p = .04598) than

[8]Levene's test for equal variance determined when to use a standard t-test vs. Welch's t-test; we point out Welch's whenever we use it.

[9]See Eqs. (5) and (6) in Appendix A for details of how we computed recall and precision for each participant. Because this data labeling would be considered multi-class and multi-label, we could not use the basic formulae. Further, while Zhang et al. [71] offer Eqs. (3) and (4) for such labellings, they do not incorporate other issues present in our data corpus. For example, few negative examples are present in the corpus because most participants only reported bugs they thought to be present.

	No credit (0)	**Partial credit** (+0.5)	**Full credit** (+1)
Location: Participant... (A)... marked location correctly	Not (A)	N/A	(A)
Description: Participant... (A)... completely described bug correctly (B)... partially described bug correctly	Neither (A) nor (B)	(B)	(A)

Table 2. The coding rules we used to code participants' problem reports. (For Location, if a participant's location markings were combined in a way that introduced ambiguity, we disambiguated by looking for location information in their free-form descriptions.)

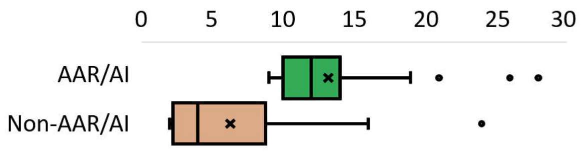


Fig. 5. Problem report count per participant. AAR/AI: Mean=13.182, SD=4.565; Non-AAR/AI: Mean=6.281, SD=5.050

. The AAR/AI participants submitted significantly more problem reports than their Non-AAR/AI counterparts.

the Non-AAR/AI participants (Figure 6). Cohen's *d* showed a large effect size (d = 1.121) for the recall difference, and a medium effect size (d = .505) for the precision difference[10].

Together, these three results suggest that the AAR/AI process not only encouraged participants to report significantly more problems (Figure 5), it also encouraged them to report problems that were indeed bugs, as measured by their significantly higher recall and precision (Figure 6). These results are especially encouraging given that none of the participants had backgrounds in AI/ML.

4.2 RQ2 Results: Does the type of fault matter?

RQ2 raises the question of whether AAR/AI vs. non-AAR/AI participants' success differences depended on which particular bugs or types of bugs they were pursuing. We begin by considering the types of bugs: Leaf Evaluation Function bugs vs. Transition Function bugs.

Leaf Evaluation Function (LEF) bugs occur when the neural network provides an inaccurate game outcome for an input state, such as in Figure 7. Transition Function (TF) bugs occur when the neural network predicts an inaccurate future state, given a current state and actions, such as in Figure 8. The experiment's 10 bugs were evenly split between 5 LEF and 5 TF bugs. RQ2 asks whether AAR/AI vs. non-AAR/AI played out differently for these two bug types.

[10]We consider Cohen's $d \in [0, 0.2)$ to be no effect, $d \in [0.2, 0.5)$ to be small, $d \in [0.5, 0.8)$ to be medium, and $d \in [0.8, 1.4)$ to be large, by convention [16].

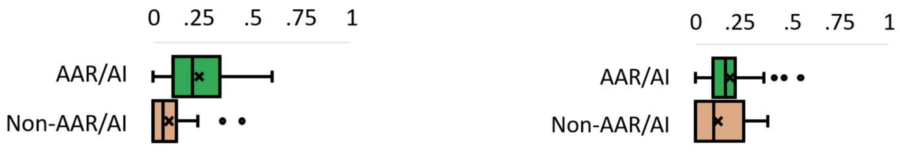
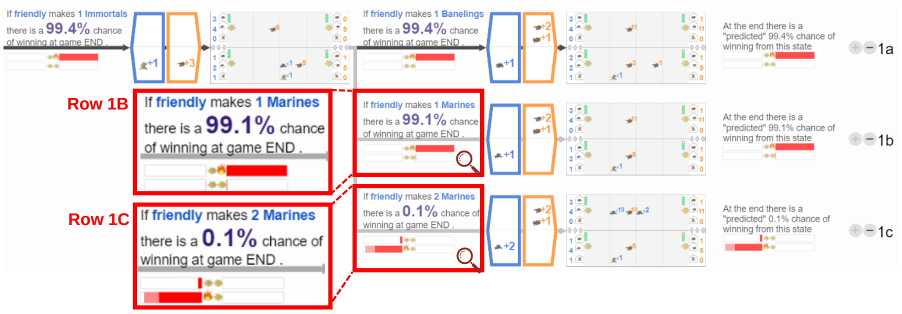


Fig. 6. Participants' recall (left) and precision (right). Recall AAR/AI: Mean=0.233, SD=0.160; Non-AAR/AI: Mean=0.080, SD=0.105. Precision AAR/AI: Mean=0.179, SD=0.129; Non-AAR/AI: Mean=0.116, SD=0.121 over all 10 bugs

. AAR/AI participants performed significantly better than Non-AAR/AI participants with both measures.


If friendly makes 1 Immortals there is a 99.4% chance of winning at game END .

If friendly makes 1 Banelings there is a 99.4% chance of winning at game END .

At the end there is a "predicted" 99.4% chance of winning from this state

If friendly makes 1 Marines there is a 99.1% chance of winning at game END .

At the end there is a "predicted" 99.1% chance of winning from this state

If friendly makes 2 Marines there is a 0.1% chance of winning at game END .

At the end there is a "predicted" 0.1% chance of winning from this state

Fig. 7. Example of Leaf Evaluation Function (LEF) bug (Bug ID #1), present at DP 8. The game outcome for row 1C is suspicious. Since the Friendly AI's action (i.e., 2 marines) is similar to that for 1A and 1B (especially 1B: 1 marine), we can expect that the Friendly AI would win (>99% chance of winning) by destroying the top enemy nexus (as in row 1B), however, the agent predicts that Friendly AI will lose (0.1% chance of winning), which implies that the win probabilities for row 1C have likely been flipped (i.e., LEF bug).

To answer this question, we analyzed recall and precision separately for LEF bugs and TF bugs. (Note that the number of target bugs is now split into two for analysis, which affects the distributions.) Figure 9 shows the results, with recall in the left two pairs (LEF and TF bugs, respectively), and precision in the right two pairs. The AAR/AI vs. non-AAR/AI recall differences for both bug types were significant. Specifically, AAR/AI participants found significantly greater proportions of both LEF bugs (t-test, t(63) = 3.0358, p = .0035) and TF bugs (Welch's t-test, t(51.341) = 4.7479, $p < .001$). Average precision differences in AAR/AI participants vs. non-AAR/AI participants were suggestive, but did not reach significance for either LEF bugs (t-test, t(63) = 1.1891, p = .2389) or TF bugs (t-test, t(63) = 1.5878, p = .1173).

As these LEF vs. TF recall and precision results show, bug type did not determine when AAR/AI participants were more effective than non-AAR/AI participants—AAR/AI participants performed at least as well as non-AAR/AI participants on both bug types. In fact, as Figure 10 shows, AAR/AI participants outperformed non-AAR/AI participants on *every* bug. On average, AAR/AI participants were about 6 times as likely as non-AAR/AI participants to find each bug (odds ratio for two sample proportions [55], $\hat{\phi} = 5.889$).

4.3 RQ3 Results: Labeling and Abstractions

This study's third research question explored whether adding labeling to the AAR/AI process would facilitate participants' ability to spot patterns of bugs and potentially develop abstractions capturing these patterns. Toward that end, our interface enabled AAR/AI participants to label the bugs they localized as they went along. They had free rein to label any way they chose, or not to label at all. We then analyzed the kinds of labels participants used and whether their use of different kinds of labels related to their successes at localizing bugs.

4.3.1 *What kinds of labels did they devise?*

Participants used a wide variety of labels to characterize the bugs they found. Some seemed to use labels simply as a way to group similar instances (e.g., "1", "problem2"); some used location information in their labeling schemes (e.g., "first row"); and some used labels for lighter purposes (e.g., "bored"). However, some participants' labels abstracted above individual instances into concepts, either from a "naive-AI" perspective (e.g., "badprediction") or from a domain perspective (e.g., "marines to be made").

We coded the labels into categories. Two researchers generated the code set using a process similar to Hsieh & Shannon's summative content analysis [29], where keywords are identified before and during data analysis to form the code set. This generated the categories of labels shown in Table 3. When a participant's label was applicable to more than one category, we coded it in

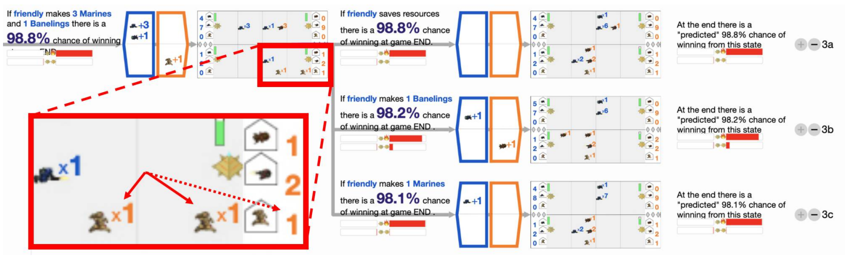

If friendly makes 3 Marines and 1 Banelings there is a 98.8% chance of winning
+3
+1
+1
If friendly saves resources there is a 98.8% chance of winning at game END.
At the end there is a "predicted" 98.8% chance of winning from this state
3a
If friendly makes 1 Banelings there is a 98.2% chance of winning at game END .
At the end there is a "predicted" 98.2% chance of winning from this state
3b
If friendly makes 1 Marines there is a 98.1% chance of winning at game END .
At the end there is a "predicted" 98.1% chance of winning from this state
3c
x1
x1
x1
1
2
1

Fig. 8. Example of Transition Function (TF) bug (Bug ID #10), present at DP 8. The bug (in the highlighted box), is that the agent predicts there will be 2 immortals in the bottom lane (solid arrows), even though there is only one immortal production building (dashed arrow).

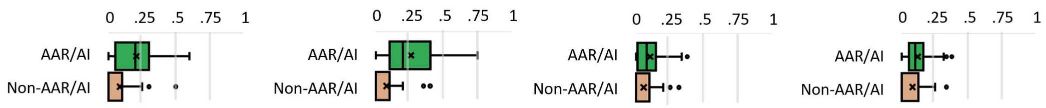
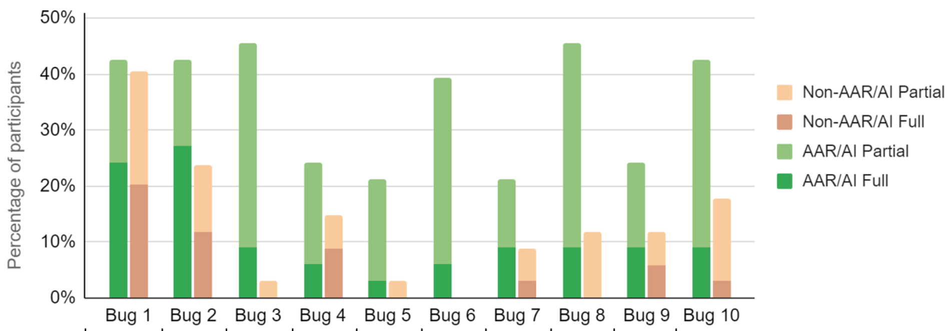


Fig. 9. AAR/AI participants' vs. non-AAR/AI participants' (left to right): recall for leaf-evaluation function (LEF) bugs only (AAR/AI: Mean=0.208, SD=0.185; Non-AAR/AI: Mean=0.075, SD=0.105), recall for transition function bugs (TF) only (AAR/AI: Mean=0.258, SD=0.188; Non-AAR/AI: Mean=0.086, SD=0.135), precision for LEF bugs only (AAR/AI: Mean=0.100, SD=0.105; Non-AAR/AI: Mean=0.068, SD=0.109), and precision for TF bugs only (AAR/AI: Mean=0.122, SD=0.101; Non-AAR/AI: Mean=0.078, SD=0.122).



	Bug 1	Bug 2	Bug 3	Bug 4	Bug 5	Bug 6	Bug 7	Bug 8	Bug 9	Bug 10
Type	LEF	TF	TF	TF	LEF	TF	LEF	LEF	LEF	TF
DP	8	8	8	8	8	15	15	15	15	15
Odds Ratio	1.018	2.211	25.83	1.728	8.346		2.603	5.833	2.24	3.193

Fig. 10. Percentage of participants who found each bug. More AAR/AI participants than Non-AAR/AI participants found every one of the bugs. Odds Ratios show AAR/AI participants' increased likelihood of finding each bug. (Bug 6's odds ratio is undefined because 0 Non-AAR/AI participants found that bug.)

all the applicable categories. For example, "battlefield counting issues" was coded as an instance of both Domain Concepts and Counting/Math. The researchers independently coded 20% of the data with 85% agreement (Jaccard index [31]), coding the labels directly or in the context of the participants' reports when necessary to disambiguate labels. Given these researchers' high level of code consistency, only one researcher was needed to complete the rest of the coding.

As Figure 11 shows, participants' use of labels most frequently tended towards abstractions relating to concepts of the game (Domain Concepts) or concepts of AI and/or AI Explanations (AI/XAI Concepts), with more than 120 instances of each. The next most frequent was using labels as simply grouping mechanisms (Identified Groups) (e.g., "ai3", "problem2"); there were more than 80 instances of these. Participants also sometimes used "Not a Problem After All" labels as a way to document "all clear" diagnoses after perusing the Explanation UI; there were 70 instances of this category. The Bug Location category (e.g., "first row") and "Un-category" category (labels not even attempting to categorize; e.g., "abc", "can't understand") were also somewhat common, with more than 30 instances of each. Lowest in frequency was the No Time category, in which participants used labels to document where they ran out of time, with 6 instances.

4.3.2 *Which of their labels correlated with success?*

Of these categories, the three codes showing positive correlations with participants' success (scores) were AI/XAI concepts, Counting/Math, and Domain Concepts (Figure 12). Each of these

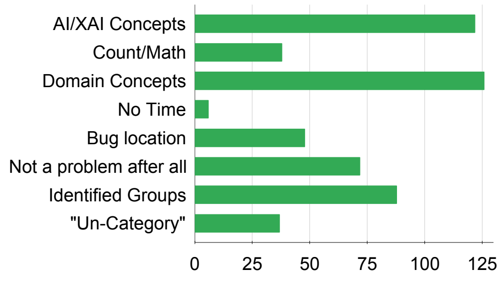


Fig. 11. Frequencies of AAR/AI participants' ways of labeling faults into these categories. (Non-AAR/AI participants did not have a labeling feature.) Domain Concepts and AI/XAI Concepts were the most common.

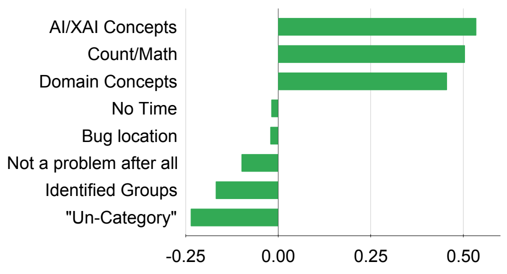


Fig. 12. Correlation between the scores in each categorization scheme and the participant's total score.

had a correlation coefficient r between $[0.45, 0.54]$, which is generally considered to be a moderate correlation [22]. In contrast, the remaining categories had small *negative* correlations with participants' success.

AI/XAI Concept labels had the highest correlation with the participants' success scores ($r = 0.535$). Among the participants' labels hinting at AI/XAI-concept abstractions were "bad prediction" (P158, score 7; P116, score 5), "winning percentage" (P130, score 14), and "overconfidence top lane" (P106,

Code: Description	**Examples of participants' labels**
AI/XAI Concepts: Concepts/terminology related to XAI/AI	health prediction, incorrect decision, success outcome change, overestimation of ability...
Bug location: Location on the Explanation UI	next state 17, outcome 2, second action, error row 5a, 2b not possible...
Count/Math: Related to counting, math, or calculation	battlefield counting issues, nexus health calculation, too many enemies... marines...
Domain Concepts: Concepts/terminology related to the game domain (troops, lanes, nexus)	lane 1 nexus, unit disappear, nexus randomly dies, suddenly immortals, banelings in the bottom lane...
Identified Groups: Evidence of an "ID" of some kind used repeatedly to group similar instances	ai, ai2,... 1ssue, 2 issue,... problem, problem2,...
No Time: Did not have time to complete search for problems	no time, out of time
Not a problem after all: Location was marked as a (potential) problem, but added a label indicating no problem after all	no problem, n/a, no, ignore this, no issues...
"Un-category": Did not attempt to categorize; labels ranged from gibberish to messages to the researcher	error, bored, alex, cant understand, abc...

Table 3. Code set used to categorize AAR/AI participants' labels on the faults they localized.

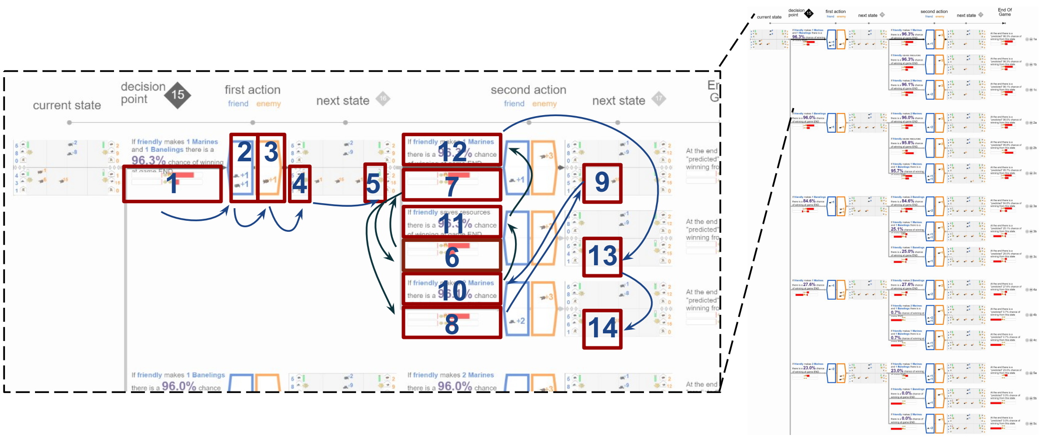


Fig. 13. P119's search path leading to a problem report they labeled "bad decisions". To report this problem, P119 walked down to the second level, then perused each node at the second level, drilling down further if the node seemed potentially problematic (e.g., node 8), then returned to their progression through the second level until the need arose to drill down again (e.g., node 12).

score 5.5). Participants' label usage in this category, when matched up with their click histories, suggested that they may have located bugs by walking through the explanation tree showing the AI's reasoning, in a "reasoning walkthrough" somewhat analogous to a code walkthrough. Figure 13 illustrates one such walkthrough. Another example excerpted from a participant's report is:

P120 (Labeled "Action-prediction incompatibility"): *"What: AI adds a friendly baneling to the top row while assuming the opponent will save money. Despite this addition to units, the prediction for success does not increase from 98%. I would think it would increase."*

Counting/Math labels also correlated with participants' successful outcomes (r = .504). Participants' labels in the Counting/Math category pointed out where some number (in the game UI) or calculation (made by the AI player and shown in the explanation tree) was incorrect. Some of these referred to only to math (P102, score 3: "number off"), but some also brought in AI/XAI concepts (P123, score 11: "chances <probabilities> are wrong"), or Domain Concepts (P111, score 12: "battlefield counting issues"). Participants' reports with Counting/Math labels seemed to suggest that participants were localizing bugs by "auditing" the counting/math via the explanation tree:

P128 (Labeled "number"): *"the numbers are inconsistent. the total marines are 9 but it only shows 6 in the game area."*

P104 (Labelled "troop_calculation"): *"during action, friendly will have 4 marines, 1 each of immortal and baneling on map, enemy has 5 marines, 1 baneling. 3 friendly marines and 1 baneling was destroyed while only 1 enemy baneling was destroyed. does not seem to add up if it was a fair game."*

Domain Concept labels also were correlated with participants' successful outcomes (r = .455). Participants' use of Domain Concept labels often showed that they were localizing bugs by mapping what they saw in the explanations to game concepts, then using them to spot portions of the explanation running counter to the logic of the game. For example, P111 and P137 spotted bugs via game-logic contradictions, which they both labeled using Domain Concepts:

P111 (Labeled "evaluated battlefield error"): *"What: there are 3 more marines that are alive the next time than the previous. Though the situations have not changed, so neither should the speculated battlefield.*
Why: I'm not actually sure why, other than a misprediction.

> *What changes: Change the 10 state bottom lane to have 3 enemy marines to match the previous prediction."*
>
> P137 (Labeled "Nexus health"): *"What: The health bar for Enemy bottom Nexus isn't consistent. It's low in the first state and then becomes full."*

Although some examples in the Domain Concept category, like the above, were solely Domain Concept labels, about half the instances in this category also related to the AI/XAI category and/or the Counting/Math category. For example, the "evaluated battlefield error" entry above by P111 (score 12) was also in the AI/XAI Concept category. Another example was P120's "improper unit count", which was in both the Domain Concept category and the Counting/Math category (P120, score 5). These co-occurrences mark instances in which participants may have been reasoning about a single bug in multiple ways.

Taken together, the correlations between success and the AI/XAI Concepts, Counting/Math audits, and Domain Concepts suggest that adding labeling to the AAR/AI process may be a potentially powerful aid. Perhaps the participants' labeling effort facilitated forms of self-explanation, in which participants were able to make sense of the individual instances of bugs by (self-)explaining them via other patterns of reasoning, such as concepts of math, RTS gameplay, or how they assumed AI agents work.

5 DISCUSSION

Did using AAR/AI impose an extra cognitive load on the AAR/AI participants beyond the load experienced by non-AAR/AI participants? We expected it to, because in our past XAI research [5], participants paid a statistically significant cognitive load cost for their successes with the most efficacious explanations. Thus, we analyzed participants' perceptions of cognitive load via the NASA Task-Load indeX (TLX), to see if adding the AAR/AI process on top of the explanations imposed a cognitive load "tax".

To our surprise, none of the 5 NASA TLX dimensions gathered[11], showed statistically significant differences between the AAR/AI and Non-AAR/AI participants; all p-values fell within [.133, .878]. That said, the temporal demand dimension was particularly interesting.

As with the other NASA/TLX questions, AAR/AI vs. Non-AAR/AI participants did notdiffer statistically in the amount of temporal demand that they reported (TLX Temporal dimension: t-test, t(49) = 1.5276, p = .1331). However, somewhat contradicting their self-reports, AAR/AI participants actually *spent* significantly more time than non-AAR/AI participants did, for both DP7 (Welch's t-test, t(50) = 2.3095, p = .0251) and DP14 (Welch's t-test, t(49) = 2.5874, p = .0127). Given this contradiction between participants' self-reported perceptions of time and actual time taken, and in light of the remaining non-significant differences in the TLX responses, further studies are required to answer whether adding AAR/AI to XAI adds cognitive load to domain-knowledgeable users' efforts to assess their AI agents—and if so, whether the added cognitive load is desirable (e.g., participants better scrutinizing the AI) or undesirable (e.g., participants wasting time due to AAR/AI features).

As with all empirical studies, our study has limitations and threats to validity [36, 67]. One challenge with controlled experiments involving human participants in XAI is controlling *which* portions of the explanations that participants see. If participants are allowed to explore freely through a huge virtual explanation space, no two participants see the same explanations. In a quantitative experiment, such uncontrolled variations in participants' experiences would introduce too much experimental noise for inferential statistics to be useful. For example, in a qualitative

[11]The "physical" dimension was not gathered because the study was online rather than in the lab. 14 participants (21.6%) did not complete the questionnaire: 8/33 AAR/AI participants and 6/32 of the non-AAR/AI participants.

study in the RTS domain, Penney et al. [52, 53] showed that different participants focused on different things.

To ensure that participants in both treatments could start the experiment seeing exactly the *same* subset of the virtual explanation space, we pruned the XAI explanation tree they could view[12]. Constraining their view had the benefit of keeping their attention in parts of the explanation space where we had previously confirmed bugs. Participants could then pan/zoom/collapse the amount of information on the screen, but could not expand beyond the original subset. We selected the explanation subset such that it included information pertinent to every bug. This involved pruning away large portions of the virtual explanation space, which raises an ecological validity threat. This is an example of a classic trade-off for most controlled experiments, in which some ecological validity must be traded off to achieve the controls necessary to isolate an independent variable [36] (in our study, to remove other factors so as to isolate the AAR/AI vs. non-AAR/AI independent variable).

Another threat to ecological validity is the bugs participants needed to locate. Our bugs were naturally occurring bugs—they turned up without our help in the AI-learned model. As Ko et al. point out, naturally occurring bugs increase ecological validity [36]. However, our pilot participants revealed that some of these natural bugs were so subtle, participants rarely could locate them. This could have led to "floor effects", in which the task is so difficult, no participant can complete them in the allotted time no matter which tool they use [57], and no effects can be revealed by statistics. To address this issue, we exaggerated the size[13] of some of these naturally occurring bugs, as enumerated in Table 6. These exaggerations likely affected participants' success rates; however, this threat was equally present in both treatments.

Another potential threat is in how we calculated the measures of participants' success rates with recall and precision. Calculating recall and precision with bug identification is normally straightforward, because a single area of code either is or is not faulty, and a single bug report usually describes a single issue. However, in our experiment, a single problem report could (and sometimes did) point at multiple bugs. Also, when two of the bugs were co-located in a single node of the explanation (Bug ID 1 and 2), attribution of participants' bug reports to one of those two bugs was even more difficult. These complexities break the 1-to-1 correspondence between report and bug, yielding a multi-class, multi-label problem. Thus, we had to derive our own calculations for recall and precision, as detailed in Appendix A. The uniqueness of our recall and precision calculations affect the ability to precisely compare them against simpler precision and recall calculations used in other fault localization literature.

Ideally, participants would have been given enough time to find all the bugs (high recall), and to do so carefully enough to achieve high precision (with few false positives). In our study, neither of these measures' averages reached 25%. A likely contributor to this was our need to control the amount of time a participant could spend in the study. Given the practicality of people's schedules, we settled on 2-hour session times, which does not seem to have been enough time to complete the task. However, this constraint applied equally to both treatments, so it does not threaten the validity of our comparisons between treatments.

Another threat to participants' ability to localize the bugs is that we avoided defining what a fault/bug/problem is, because we did not want to influence participants' assessment efforts. Even

[12]Some of the virtual explanation space was not available even to us, because like many AI agents, the AI system proactively pruned away unpromising portions of its potential solution space—and, as a side effect, the explanation space—to reduce calculation time.

[13]As an additional safeguard, we included multiple exaggerations amounts for the same bug type in our experiment (e.g., some bugs being a small, medium, or large exaggeration of another bug type, as detailed in Table 6). Participants' results did not reveal any patterns as to whether the size of the exaggeration mattered.

when participants asked if the problem they found was really a problem, the researcher did not answer, for ecological validity reasons—in software debugging, there is no "oracle" monitoring someone's debugging efforts to tell them whether or not a line of code they are puzzling over is problematic. However, this design choice introduced a threat: how could participants find a bug without knowing what constitutes one? We attempted to head off this threat by telling them that if they thought a problem was a problem, they should report it. We also provided "definitely", "maybe", and "never mind" bug reporting options (Figure 2), to encourage participants to report everything they thought was even potentially problematic. If we had defined to participants what did and did not count as a bug, participants' success rates would probably have been different.

Limitations like these can be addressed only by additional studies across a spectrum of empirical methods, to isolate different independent variables of study and to establish generality of findings over different explanation styles, different bugs, different measures, different AI algorithms, different domains, and different populations attempting to find an AI agent's problematic behaviors.

6 CONCLUSION

In this paper, we presented the results of an empirical study comparing domain-knowledgeable users' attempts to find an AI agent's bugs using AAR/AI (After-Action Review for AI) vs. not using AAR/AI. The results showed that:

- AAR/AI participants' recall rate on the bugs they reported was significantly higher than non-AAR/AI participants', with a large effect size. This indicates that AAR/AI participants found more of the actual bugs, one of the key goals of assessment in XAI domains.
- AAR/AI participants also reported a significantly larger number of problems. This result would be worrying if it showed that they achieved high recall simply by reporting that everything was wrong. However, the results showed that this was not the case because...
- ...AAR/AI participants also showed significantly higher precision than non-AAR/AI participants, with a medium effect size. Typically, there is a tradeoff in precision vs. recall, so increasing both is a very strong improvement.
- When considering the bugs one by one, we saw no evidence of AAR/AI's advantages being particular to specific bugs or types of bugs—rather, AAR/AI participants outperformed non-AAR/AI participants on *every* bug. On average, AAR/AI participants were almost 6 times as likely as non-AAR/AI participants to find any particular bug.
- The AAR/AI participants' labeling behaviors suggest that incorporating labeling into the AAR/AI process may bring important benefits. In this study, some AAR/AI participants used labels to abstract above individual instances of bugs, using concepts from the domain or from (X)AI. Others used labels in ways suggestive of auditing. Use of these types of labels correlated with higher recall rates in finding the bugs.

Finally, recall that the only difference in treatments was AAR/AI vs. no AAR/AI—all participants consumed the *same* explanations. These results suggest the importance of integrating explanations with an assessment process such as AAR/AI, to enable domain-knowledgeable users to make informed decisions about when to follow an AI agent's recommendations and when *not* to.

ACKNOWLEDGMENTS

We thank Tom Dietterich for words of wisdom during the research that produced this paper. This work was supported by DARPA #N66001-17-2-4030. Any opinions, findings and conclusions or recommendations expressed are the authors' and do not necessarily reflect the views of DARPA, the Army Research Office, or the US government.

REFERENCES

[1] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. 2018. Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, Article 582, 18 pages.

[2] 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 (CHI '19)*. ACM, Article 3, 13 pages.

[3] Paul Ammann and Jeff Offutt. 2016. *Introduction to software testing*. Cambridge University Press.

[4] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Burnett. 2019. Explaining Reinforcement Learning to Mere Mortals: An Empirical Study. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI '19)*.

[5] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Matthew Olson, Alan Fern, and Margaret Burnett. 2020. Mental Models of Mere Mortals with Explanations of Reinforcement Learning. *ACM Transactions on Interactive Intelligent Systems* 10, 2, Article 15 (May 2020), 37 pages. https://doi.org/10.1145/3366485

[6] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1, 1 (2004), 11–33.

[7] Adam T. Biggs and Stephen R. Mitroff. 2015. Improving the Efficacy of Security Screening Tasks: A Review of Visual Search Challenges and Ways to Mitigate Their Adverse Effects. *Applied Cognitive Psychology* 29, 1 (2015), 142–148. https://doi.org/10.1002/acp.3083

[8] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. 1956. *Taxonomy of Educational Objectives*. Longmans, Green and Co LTD.

[9] Ralph Brewer, Anthony Walker, E. Ray Pursel, Eduardo Cerame, Anthony Baker, and Kristin Schaefer. 2019. Assessment of Manned-Unmanned Team Performance: Comprehensive After-Action Review Technology Development. In *2019 International Conference on Human Factors in Robots and Unmanned Systems (AHFE '19)*. Springer Nature Switzerland AG, Cham, CHE, 119–130.

[10] Ruth M. J. Byrne. 2019. Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI'19)*. International Joint Conferences on Artificial Intelligence Organization, 6276–6282. https://doi.org/10.24963/ijcai.2019/876

[11] Nico Castelli, Corinna Ogonowski, Timo Jakobi, Martin Stein, Gunnar Stevens, and Volker Wulf. 2017. What Happened in my home? An end-user development approach for smart home data visualization. In *ACM Conference on Human Factors in Computing Systems*. ACM, 853–866.

[12] Nan-Chen Chen, Jina Suh, Johan Verwey, Gonzalo Ramos, Steven Drucker, and Patrice Simard. 2018. AnchorViz: Facilitating Classifier Error Discovery through Interactive Semantic Data Exploration. In *Proceedings of the 23th International Conference on Intelligent User Interfaces (IUI '18)*. ACM, 269–280.

[13] Michelene T.H. Chi, Miriam Bassok, Matthew W. Lewis, Peter Reimann, and Robert Glaser. 1989. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science* 13, 2 (4 1989), 145–182. https://doi.org/10.1207/s15516709cog1302_1

[14] Michael Chromik, Malin Eiband, Felicitas Buchner, Adrian Krüger, and Andreas Butz. 2021. I Think I Get Your Point, AI! The Illusion of Explanatory Depth in Explainable AI. In *26th International Conference on Intelligent User Interfaces* (College Station, TX, USA) *(IUI '21)*. Association for Computing Machinery, New York, NY, USA, 307–317. https://doi.org/10.1145/3397481.3450644

[15] Michael Chromik, Malin Eiband, Sarah Theres Völkel, and Daniel Buschek. 2019. Dark Patterns of Explainability, Transparency, and User Control for Intelligent Systems. In *IUI Workshops*.

[16] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Academic press.

[17] Kelley Cotter, Janghee Cho, and Emilee Rader. 2017. Explaining the News Feed Algorithm: An Analysis of the "News Feed FYI" Blog. In *ACM CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1553–1560.

[18] Robert Davies, Elly Vaughan, Graham Fraser, Robert Cook, Massimo Ciotti, and Jonathan E. Suk. 2019. Enhancing Reporting of After Action Reviews of Public Health Emergencies to Strengthen Preparedness: A Literature Review and Methodology Appraisal. *Disaster Medicine and Public Health Preparedness* 13, 3 (june 2019), 618–625. https://doi.org/10.1017/dmp.2018.82

[19] Jonathan Dodge, Roli Khanna, Jed Irvine, Kin-Ho Lam, Theresa Mai, Zhengxian Lin, Nicholas Kiddle, Evan Newman, Andrew Anderson, Sai Raja, Caleb Matthews, Christopher Perdriau, Margaret Burnett, and Alan Fern. 2021. After-Action Review for AI (AAR/AI). *ACM Transactions on Interactive Intelligent Systems* (2021). To Appear.

[20] Jonathan Dodge, Sean Penney, Claudia Hilderbrand, Andrew Anderson, and Margaret Burnett. 2018. How the Experts Do It: Assessing and Explaining Agent Behaviors in Real-Time Strategy Games. In *2018 CHI Conference on Human*

Factors in Computing Systems (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 562, 12 pages.

[21] Luciano Floridi, Josh Cowls, Monica Beltrametti, Raja Chatila, Patrice Chazerand, Virginia Dignum, Christoph Luetge, Robert Madelin, Ugo Pagallo, Francesca Rossi, et al. 2018. AI4People—an ethical framework for a good AI society: opportunities, risks, principles, and recommendations. *Minds and Machines* 28, 4 (2018), 689–707.

[22] David Freedman, Robert Pisani, and Roger Purves. 2007. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York* (2007).

[23] A. Groce, T. Kulesza, C. Zhang, S. Shamasunder, M. Burnett, W. Wong, S. Stumpf, S. Das, A. Shinsel, F. Bice, and K. McIntosh. 2014. You Are the Only Possible Oracle: Effective Test Selection for End Users of Interactive Machine Learning Systems. *IEEE Transactions on Software Engineering* 40, 03 (mar 2014), 307–323. https://doi.org/10.1109/TSE.2013.59

[24] Samer Hanoun and Saeid Nahavandi. 2018. Current and Future Methodologies of After Action Review in Simulation-based Training. In *2018 Annual IEEE International Systems Conference (SysCon)* (Vancouver, BC, CAN) *(SysCon '18)*. IEEE, New York, NY, USA, 1–6.

[25] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139 – 183. https://doi.org/10.1016/S0166-4115(08)62386-9

[26] Robert Hoffman, Gary Klein, and Shane Mueller. 2018. Explaining Explanation For "Explainable Ai". *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 62 (09 2018), 197–201. https://doi.org/10.1177/1541931218621047

[27] Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for Explainable AI: Challenges and Prospects. *CoRR* abs/1812.04608 (2018). arXiv:1812.04608 http://arxiv.org/abs/1812.04608

[28] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2019. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics* 25, 8 (2019), 2674–2693. https://doi.org/10.1109/TVCG.2018.2843369

[29] Hsiu-Fang Hsieh and Sarah E Shannon. 2005. Three approaches to qualitative content analysis. *Qualitative health research* 15, 9 (2005), 1277–1288.

[30] Andrew Ishak and Elizabeth Williams. 2017. Slides in the Tray: How Fire Crews Enable Members to Borrow Experiences. *Small Group Research* 48, 3 (March 2017), 336–364. https://doi.org/10.1177/1046496417697148

[31] Paul Jaccard. [n.d.]. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.* 44 ([n. d.]), 223–270.

[32] Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. 2019. Explainable reinforcement learning via reward decomposition. In *Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence*. 47–53.

[33] Nathanael Keiser and Winfred Arthur, Jr. 2020. A meta-analysis of the effectiveness of the after-action review (or debrief) and factors that influence its effectiveness. *Journal of Applied Psychology* (08 2020). https://doi.org/10.1037/apl0000821

[34] Man-Je Kim, Kyung-Joong Kim, SeungJun Kim, and Anind Dey. 2016. Evaluation of StarCraft Artificial Intelligence Competition Bots by Experienced Human Players. In *2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, CA, USA) *(CHI EA '16)*. ACM, New York, NY, USA, 1915–1921.

[35] Man-Je Kim, Kyung-Joong Kim, SeungJun Kim, and Anind K Dey. 2016. Evaluation of StarCraft Artificial Intelligence Competition Bots by Experienced Human Players. In *ACM CHI Conference Extended Abstracts*. ACM, 1915–1921.

[36] A J Ko, T D Latoza, and M M Burnett. 2015. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering* 20, 1 (2015), 110–141.

[37] Cliff Kuang. 2017. Can AI be taught to explain itself? New York Times. (2017). Retrieved December 26, 2017 from https://www.nytimes.com/2017/11/21/magazine/can-ai-be-taught-to-explain-itself.html.

[38] T. Kulesza, M. Burnett, W. Wong, and S. Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *ACM International Conference on Intelligent User Interfaces*. ACM, 126–137.

[39] Todd Kulesza, Simone Stumpf, Margaret Burnett, Weng-Keen Wong, Yann Riche, Travis Moore, Ian Oberst, Amber Shinsel, and Kevin McIntosh. 2010. Explanatory debugging: Supporting end-user debugging of machine-learned programs. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 41–48.

[40] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W. K. Wong. 2013. Too much, too little, or just right? Ways explanations impact end users' mental models. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing (VL/HCC)*. 3–10. https://doi.org/10.1109/VLHCC.2013.6645235

[41] Kin-Ho Lam, Zhengxian Lin, Jed Irvine, Jonathan Dodge, Zeyad T Shureih, Roli Khanna, Minsuk Kahng, and Alan Fern. 2020. Identifying Reasoning Flaws in Planning-Based RL Using Tree Explanations. In *IJCAI-PRICAI 2020 Workshop on XAI*. https://drive.google.com/file/d/1ihT39-S2SFpCsarJfzDsnMOUT85YjCXK/view?usp=sharing

[42] Adam Lareau and Brice Long. 2018. The Art of the After-Action Review. *Fire Engineering* 171, 5 (May 2018), 61–64. http://search.proquest.com/docview/2157468757/

[43] Brian Y. Lim. 2012. *Improving understanding and trust with intelligibility in context-aware applications*. Ph.D. Dissertation. Carnegie Mellon University.

[44] Brian Y. Lim and Anind K. Dey. 2009. Assessing demand for intelligibility in context-aware applications. In *ACM International Conference on Ubiquitous Computing*. ACM, 195–204.

[45] Sandra Deacon Lloyd Baird, Phil Holland. 1999. Learning from action: Imbedding more learning into the performance fast enough to make a difference. 27 (1999), 19–32. https://doi.org/10.1016/S0090-2616(99)90027-X

[46] Theresa Mai, Roli Khanna, Jonathan Dodge, Jed Irvine, Kin-Ho Lam, Zhengxian Lin, Nicholas Kiddle, Evan Newman, Sai Raja, Caleb Matthews, Christopher Perdriau, Margaret Burnett, and Alan Fern. 2020. Keeping It "Organized and Logical": After-Action Review for AI (AAR/AI). In *25th International Conference on Intelligent User Interfaces (IUI '20)*. ACM. https://doi.org/10.1145/3377325.3377525

[47] Ronald Metoyer, Simone Stumpf, Christoph Neumann, Jonathan Dodge, Jill Cao, and Aaron Schnabel. 2010. Explaining how to play real-time strategy games. *Knowledge-Based Systems* 23, 4 (2010), 295–301.

[48] Nicole Mirnig, Gerald Stollnberger, Markus Miksch, Susanne Stadler, Manuel Giuliani, and Manfred Tscheligi. 2017. To err is robot: How humans assess and act toward an erroneous social robot. *Frontiers in Robotics and AI* 4 (2017), 21.

[49] Donald A Norman. 1983. Some observations on mental models. *Mental Models* 7, 112 (1983), 7–14.

[50] European Group on Ethics in Science and New Technologies. 2018. Statement on Artificial Intelligence, Robotics and 'Autonomous' Systems. (2018). https://ec.europa.eu/info/news/ethics-artificial-intelligence-statement-ege-released-2018-apr-24_en

[51] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. 2013. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games* 5, 4 (Dec 2013), 293–311. https://doi.org/10.1109/TCIAIG.2013.2286295

[52] Sean Penney, Jonathan Dodge, Andrew Anderson, Claudia Hilderbrand, Logan Simpson, and Margaret Burnett. [n.d.]. The Shoutcasters, the Game Enthusiasts, and the AI: Foraging for Explanations of Real-Time Strategy Players. 0, ja ([n. d.]). https://doi.org/10.1145/3396047 (To Appear).

[53] Sean Penney, Jonathan Dodge, Claudia Hilderbrand, Andrew Anderson, Logan Simpson, and Margaret Burnett. 2018. Toward Foraging for Understanding of StarCraft Agents: An Empirical Study. In *23rd International Conference on Intelligent User Interfaces* (Tokyo, Japan) *(IUI '18)*. ACM, New York, NY, USA, 225–237. https://doi.org/10.1145/3172944.3172946

[54] John Quarles, Samsun Lampotang, Ira Fischler, Paul Fishwick, and Benjamin Lok. 2013. Experiences in mixed reality-based collocated after action review. *Virtual Reality* 17, 3 (Sept. 2013), 239–252. https://doi.org/10.1007/s10055-013-0229-6

[55] Fred Ramsey and Daniel Schafer. 2012. *The statistical sleuth: a course in methods of data analysis*. Cengage Learning.

[56] Alexander Renkl, Robin Stark, Hans Gruber, and Heinz Mandl. 1998. Learning from Worked-Out Examples: The Effects of Example Variability and Elicited Self-Explanations. *Contemporary Educational Psychology* 23, 1 (Jan. 1998), 90–108. https://doi.org/10.1006/ceps.1997.0959

[57] Robert Rosenthal and Donald B Rubin. 1978. Interpersonal expectancy effects: The first 345 studies. *Behavioral and Brain Sciences* 1, 3 (1978), 377–386.

[58] Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

[59] Taylor Lee Sawyer and Shad Deering. 2013. Adaptation of the US Army's After-Action Review for Simulation Debriefing in Healthcare. *Simulation in Healthcare* 8, 6 (Dec. 2013), 388–397. https://doi.org/10.1097/SIH.0b013e31829ac85c

[60] James Schaffer, John O'Donovan, James Michaelis, Adrienne Raglin, and Tobias Höllerer. 2019. I Can Do Better Than Your AI: Expertise and Explanations. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. ACM, New York, NY, USA, 240–251. https://doi.org/10.1145/3301275.3302308

[61] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander Rush. 2019. Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-Sequence Models. *IEEE Transactions on Visualization and Computer Graphics* 25 (2019), 353–363. Issue 1. https://doi.org/10.1109/TVCG.2018.2865044

[62] John Sweller, Jeroen J. G. Van Merrienboer, and Fred Paas. 1998. Cognitive Architecture and Instructional Design. *Educational Psychology Review* 10 (09 1998), 251–. https://doi.org/10.1023/a:1022193728205

[63] J. Tullio, A. Dey, J. Chalecki, and J. Fogarty. 2007. How it works: A field study of non-technical users interacting with an intelligent system. In *ACM Conference on Human Factors in Computing Systems*. ACM, 31–40.

[64] U.S. Army. 1993. *Training Circular 25-20: A Leader's Guide to After-Action Reviews*. Technical Report. Department of the Army, Washington D.C., USA.

[65] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing Theory-Driven User-Centric Explainable AI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '19)*.

[66] J. Wang, L. Gou, H. Shen, and H. Yang. 2019. DQNViz: A Visual Analytics Approach to Understand Deep Q-Networks. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 288–298.

[67] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.

[68] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL '19)*. 747–763. https://doi.org/10.18653/v1/P19-1073

[69] 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 (CHI '20)*. ACM, 13 pages. https://doi.org/10.1145/3313831.3376301

[70] Qian Yang, Aaron Steinfeld, and John Zimmerman. 2019. Unremarkable ai: Fitting intelligent decision support into critical, clinical decision-making processes. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. 11 pages.

[71] M. Zhang and Z. Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

A APPENDIX A: ANALYSIS MATH

Because our participants could select multiple nodes in the diagram, our analysis was faced with a multi-class, multi-label problem. In classification problems with two classes, recall and precision can be computed via familiar expressions like the following:

$$Recall_{Basic}(TP, FP, TN, FN) = \frac{TP}{TP + FN} \tag{1}$$

$$Precision_{Basic}(TP, FP, TN, FN) = \frac{TP}{TP + FP} \tag{2}$$

To find a multi-class, multi-label analog, we consulted a review by Zhang et al. [71], which describes two flavors of multi-label analysis. One equally weights each data *instance*, and another equally weights each *label*. Either one is well-defined for us, so we picked the former approach, using the recall/precision equations from their Section 2.2.2, as Equations (3) and (4) provided verbatim here, barring a single notational change, where:

- $h(\cdot)$ is the classification function, which returns labels given the ith data point as a feature vector $\vec{x}_i$;
- d is the number of data instances (Zhang et al. used p, but we will use that later. This notation swap is the only change from their equation.);
- and Y_i is the set of ground truth labels associated with the ith data point.

$$Recall_{Zhang}(h) = \frac{1}{d}\sum_{i=1}^{d}\frac{|Y_i \cap h(\vec{x}_i)|}{|Y_i|} \tag{3}$$

$$Precision_{Zhang}(h) = \frac{1}{d}\sum_{i=1}^{d}\frac{|Y_i \cap h(\vec{x}_i)|}{|h(\vec{x}_i)|} \tag{4}$$

We started from these expressions, and cast the summands into our situation via the following steps: First, $|Y_i|$ is just the number of bugs present (in this case 10). Second, since our bugs were all known to be present, the $\vec{Y}$ for a particular DP is a 1-vector, so the intersection becomes a sum of the bugs a participant found in *all* their reports. Third, since $|h(\vec{x}_i)|$ is intended to model the "number of shots fired at targets", we use the number of problem reports that participant submitted as the denominator in precision (while one could consider 0 reports to be 0 precision because bugs were known to be present, participants provided 2 reports at minimum). Importantly, in this framing the denominator has no dependence on the summation, and so it can be pulled outside the summation.

Next, we need to manipulate the summation part to handle reports not being independent. In our case we do not get negative data instances[14] because we did not request any certifications that (regions of) the explanation were free of bugs, though a few participants saw fit to submit such reports. This means we need to interpret silence on a bug as a FN, but we can ONLY know if the participant was silent on a bug after they have finished with that subtask. Further, we will never get a TN because bugs were known to be present. The other kind of non-independence we need to handle is best illustrated by Table 4: many participants reported the same problem multiple times. To handle this without awarding excess credit, we create Table 5 so that it is sliced per *participant* by aggregating each participant's part of columns of Table 4 by taking a max over the reports from each participant.

Putting the pieces together yields Equations (5) and (6), for participant p's recall/precision where:

[14]If confused by this terminology, consult the confusion matrix at (https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative).

		Bug A		Bug B		Bug C	
ReportID	**PID**	**Describe**	**Select**	**Describe**	**Select**	**Describe**	**Select**
1	Alice	0	0	1	1	0	1
2	Alice	0	0	0	0	1	1
3	Alice	0	0	0	0	0	0
4	Bob	0	0	0	0	0	0
5	Bob	1	0	0	0	0	0
6	Cindy	0	0	0	1	0	0
7	Cindy	0	1	1	1	0	0
8	Cindy	0	0	1	1	0	0
9	Cindy	0	0	0	1	0	0

Table 4. Mockup of data post-labeling, presented per *problem report*. In our labelling, each bug could be selected and/or described properly, so 2 points available per bug. Notably, as illustrated here, participants often found the same issues repeatedly, so we devised Equations 5 and 6 to handle not awarding additional credit for these repeat finds. In the example provided, we used binary indicator variables for simplicity of presentation, though our formulation naturally handles partial credit with no modification.

		Bug A		Bug B		Bug C	
PID	**#Reports**	**Describe**	**Select**	**Describe**	**Select**	**Describe**	**Select**
Alice	3	0	0	1	1	1	1
Bob	2	1	0	0	0	0	0
Cindy	4	0	1	1	1	0	0

Table 5. Mockup of data post labeling, per participant (i.e. after taking the max across reports from that person). To compute recall/precision, one can take the sum across rows of this table, normalize by 2 (because there are 2 points per bug), then divide by the number of bugs or problem reports, respectively.

- B is the set of bugs (in our case, there are 10);
- M is the maximum score available per bug (in our case, 2);
- j indexes particular labels (of which there are BM in each report, because each bug could be described and/or labelled correctly);
- and $R(\cdot)$ is the report function, which returns the set of problem reports from a given participant.

$$Recall_{Participant}(p) = \frac{1}{|B|M} \sum_{j}^{|B|M} \max_{r \in R(p)} r_j \tag{5}$$

$$Precision_{Participant}(p) = \frac{1}{|R(p)|M} \sum_{j}^{|B|M} \max_{r \in R(p)} r_j \tag{6}$$

Calculating recall per *bug* is very similar, with the summation going down columns instead of across rows, but precision becomes a bit more complicated. Because some reports could be attributed to bugs while others could not, determining the correct number of reports to divide by requires checking to see if a report was attributed to a *different* bug. Mathematically, we use Equations (7) and (8) to compute the recall of a label b (e.g. "Bug A Select"), where:

- P is the total number of participants;
- p indexes participants;
- and $R(\cdot,\cdot)$ is an overloaded function returning the set of problem reports for a participant that could possibly be attributed to a particular bug (either by actually being attributed to that bug OR by being unable to be attributed to any bug).

$$Recall_{Label}(b) = \frac{1}{P}\sum_{p}^{P} \max_{r\in R(p)} r_b \tag{7}$$

$$Precision_{Label}(b) = \frac{1}{\sum_{p}^{P}|R(p,b)|}\sum_{p}^{P} \max_{r\in R(p)} r_b \tag{8}$$

Using these as the basis, we can analyze a bug (or kind of bug, e.g. Leaf Evaluation Function bugs) by running this on multiple columns and normalizing results appropriately.

To illustrate usage of Equations (5) and (6), one can compute the *recall* of a participant using Table 5; we can sum across the row and divide by the number of bugs (the max was already incorporated moving from Table 4 to Table 5). Similarly, for *precision*, we use the same sum across the row, but divide by the number of problem reports the participant provided. In the concrete example, $|B| = 3$ and $M = 2$, so Alice's recall is a whopping $\frac{4}{6} \approx 67\%$ with the same precision (Alice submitted 3 bug reports). Meanwhile, Bob's recall was only $\frac{1}{6} \approx 17\%$ with precision a bit higher at $\frac{1}{2*M} = 25\%$ because he did not provide many reports. Cindy, on the other hand, had recall at $\frac{3}{6} = 50\%$, but with lower precision ($\frac{3}{4*M} \approx 38\%$) because of the higher report volume.

Next, to illustrate usage of Equation (7), 1 of the 3 participants (Bob) found "Bug A Describe", so the recall for that bug would be 33%. Similarly, 1 of the 3 participants Selected Bug A, leading to a 33% recall overall for Bug A.

Finally, to illustrate usage of Equation (8), we compute the precision for "Bug B Describe". Alice went 1-for-2 (report 1 hit, report 2 was attributable to bug C, and report 3 was unable to be attributed and so possibly intended for that target). Meanwhile, Bob went 0-for-1 (report 4 was possibly intended for that target, while report 5 was attributable to Bug A). And last, Cindy went 1-for-4 (report 6 missed but was intended for Bug B, report 7 hit, report 8 hit but was duplicate, report 9 missed but was intended for Bug B). Combining these results yields $\frac{2}{7} \approx 29\%$.

B APPENDIX B: STUDY DESIGN DETAILS

B.1 Details of the StarCraft II game

In our StarCraft II setup, both players were RL powered agents. We call them the "Friendly AI" (referring back to Figure 1's left) and the "Enemy AI" (right). Participants were shown the game from the Friendly AI's perspective.

As Figure 1 shows, the game has a top and a bottom lane, each of which is a separate battlefield between the AI players. Each AI player has two Nexuses, which is the AI's base in this game. A Nexus is represented by the golden/yellow star structure at the corner of each lane. Next to each Nexus is a health bar with that Nexus's corresponding health points.

The two ways an AI player can win are by: (1) destroying one of the opposing Nexuses before 40 rounds, by bringing the Nexus health to 0, or (2) having the lowest Nexus health if all Nexuses are standing at the end of 40 rounds. Thus, in trying to win, throughout the game the AIs generate troops behind their respective Nexus to cause damage to the opposing Nexus in their lane.

The bar with diamonds at the bottom of the screen in Figure 1 is the game timeline. StarCraft II is played in rounds: each new round starts at one of the black diamonds (marked as D1, D2, etc). These are called "decision points", each marking a point where the AI decides on an action before the next round begins. At a decision point, the AI decides: (1) what troops to buy, if any, and (2) which lane to place them in (top or bottom).

To engage in battle, the AIs need minerals, because minerals are akin to money: the AI can use them to buy troops or invest in Pylons. At the start of the game, each AI is given 150 minerals, and then receives 100 minerals in every subsequent round. A Pylon generates an additional 75 minerals per round, and each AI can buy up to 3 pylons in the game. The Pylons for the Friendly and Enemy AIs are represented by the three diamonds in the center of the column on each side. The AIs spend their minerals to buy troop production buildings, which produce troops. Once an AI buys a troop production building, one unit of that type is generated in each subsequent round. After being generated, these troops will run towards the opposing nexus, and fight any units in their way. The AI cannot control what units attack the other, or troop formations; it can only buy the troop production buildings to generate troops.

The three types of troop production buildings in this game are: Marines, Banelings, and Immortals, all of which are shown in Figure 1. Marines are small, low cost units. They have the lowest health of the three troops, and attack with small, quick shots. Banelings are explosive bug units with a moderate cost. They have medium health, and they explode upon contact with an enemy unit. Immortals are large and are also the most expensive unit. They have the highest health, since they have a shield. They attack with slow shots that are effective against Nexuses. There is a rock-papers-scissors relationship between the Marines, Banelings and Immortals. Marines are effective against Immortals, Immortals against Banelings, and Banelings against Marines.

B.2 Experiment session walkthrough

Participants were given a tutorial detailing the game rules (specified above). Next, they were given access to the web interface with a unique ID and password. After entering their credentials, they were prompted by the facilitator to click on "Play" and start watching the game.

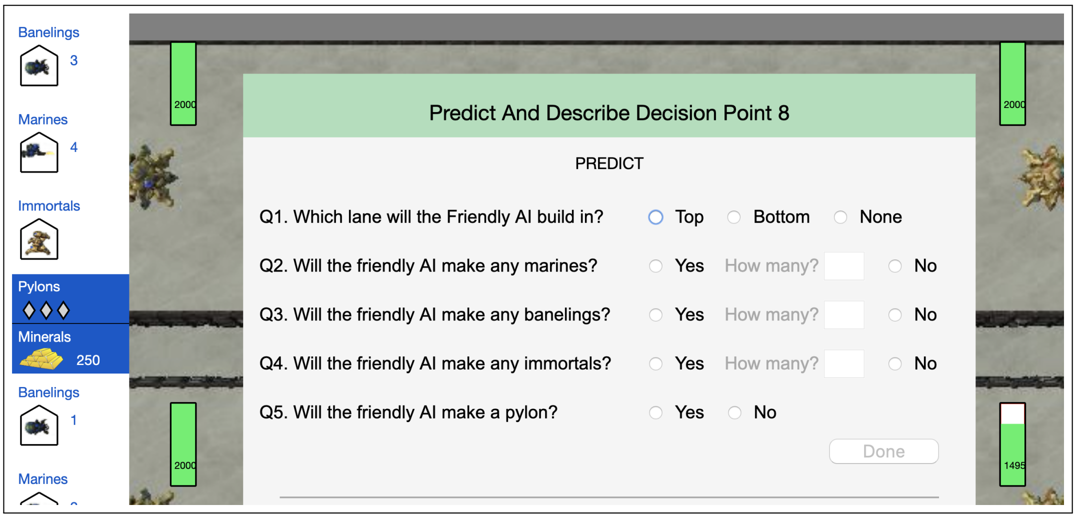
Participants in both treatments saw identical AI agents, game replays and explanations. Their agents also exhibited identical bugs, which are detailed in Table 6.

The game automatically paused at DP 8, and participants in both treatments were shown identical "prediction" questions (Figure 14). After answering these questions, participants watched the game round. After the game round ended, participants answered a "description" questions, where they described the Friendly AI's actions in the round they had just watched. The AAR/AI group was given a guided process (Figure 15), whereas the Non-AAR/AI group was given observational questions (Figure 16).

After answering the prediction and description questions, participants in both treatments then saw the Friendly AI's explanation for its actions. Figure 17 shows the AAR/AI interface, and Figure 18 shows the Non-AAR/AI interface. All participants saw the same explanations, but as these figures show, the different treatments asked different questions about the bugs participants found. Also, AAR/AI participants had to finish the row they had selected to work on before moving onto another row as part of the AAR/AI process, whereas Non-AAR/AI participants were allowed to navigate freely among rows.

ID-Type	**Description** ***(and how we engineered them)***
1-LEF	*Why a bug:* because the actions preceding state at DP 10 (i.e., predicted future states from DP 8) differ by 1 marine from its sibling actions, yet the expected outcome is radically different (flipped) than all other sibling actions. A correct state would have similar outcome expectations to the sibling states. *Exaggerated by changing:* Friendly AI's win by destroying top enemy nexus **to** friendly AI's loss by enemy AI destroying friendly AI's bottom nexus.
2-TF	*Why a bug:* because the Friendly agent has 4 marine-producing buildings in the top lane, but there are 21 total Friendly marines expected to be in the top lane, State at DP 10, row 1C. A correct state would have 4 or fewer marines in the top lane. *Exaggerated by changing:* Friendly marines top grid #1 to 19. Friendly marines top grid #2 to 2.
3-TF	*Why a bug:* because base (Nexus) health cannot heal. In expected state DP 9 row 2A, the enemy bottom base is expected to incur a significant amount of damage as shown by the red bar. However in predicted child state DP 10 row 2C, that damage is not reflected as the base's health in DP 10 row 2C is greater than its health in DP 9 row 2A. A correct state would not have greater friendly Base HP in DP 10 row 2C. *Exaggerated by changing:* Friendly Base HP D9 row 2A to 20; Friendly Base HP D10 row 2B to 100.
4-TF	*Why a bug:* because the enemy has built 1 immortal-producing building in the preceding action, and can therefore have at most 1 immortal troop on the field. This state depicts 2 immortal troops in adjacent game grid. A correct state would be 1 or fewer enemy Immortals in the bottom lane. *Exaggerated by changing:* Enemy immortal bottom grid #4 to 1; Enemy immortal grid #3 to 1.
5-LEF	*Why a bug:* because the Friendly base is expected to have 0 HP meaning it has been destroyed; therefore per game rules, the friendly agent has lost the game at DP 10. However the expected outcome shows that the friendly agent expects to destroy the enemy's top and bottom base. A correct state would have been for the enemy win 100% by destroying a friendly bottom base. *Exaggerated by changing:* Friendly bottom base HP to 0.
6-TF	*Why a bug:* recall the game rule that the top and bottom lanes are independent; troops built in one lane will not affect the outcome of what happens in the other. From DP 16 row 1A to all child actions and states, the enemy builds 3 marine-producing buildings in the top lane and the friendly does not build anything in the top lane for all 3 actions from DP 16 to DP 17 (Row 1A, 1B, and 1C). Since the action for the top lane is the same for all 3 sibling predictions, the expected state in the top lane in DP 17 in all 3 sibling states should all be the same. However state DP 17, row 1C differs from its siblings, 1 fewer marine is expected in DP 17, row 1C. *No exaggeration needed.*
7-LEF	*Why a bug:* because all sibling child states reflect a likely win in the top lane with a less likely win by purchasing troops in the bottom lane, but the bars in row 2B depict a high expectation to win in the bottom lane with no purchases in the bottom lane. *Exaggerated by changing:* Friendly win by destroying enemy bottom to 76%, Friendly win by destroying enemy top to 20%
8-LEF	*Why a bug:* because the enemy agent has 0 immortal-producing buildings in state DP 17, yet it is expected to have 1 immortal troop in the bottom lane. *Exaggerated by changing:* Enemy immortals bottom grid #2 to 1.
9-LEF	*Why a bug:* because in this predicted state the game is guaranteed to end with the enemy top base getting destroyed, but outcome bars show the friendly expecting to lose. *Exaggerated by changing:* Enemy top base HP to 0 in row 3C.
10-TF	*Why a bug:* because the game is guaranteed to end with the friendly bottom base getting destroyed sibling states in row 5B and 5C both have the correct outcome expectations (0% win, loose by friendly bottom base destory) however in row 5A the friendly expects to win by destroying enemy top. *Exaggerated by changing:* Friendly win by destroying enemy top to 23%. State D10, Row 5A, 5B, 5C set Friendly bottom base HP to 0.

Table 6. The 10 bugs; 5 in DP 8 (above the line), and 5 in DP 15 (below the line). LEF=in Leaf Evaluation Function outputs; TF=in Transition Function outputs (Section 3.3). More information about the bugs and their locations can be found in the Supplemental docs. All bugs occurred naturally, but we exaggerated some as marked.



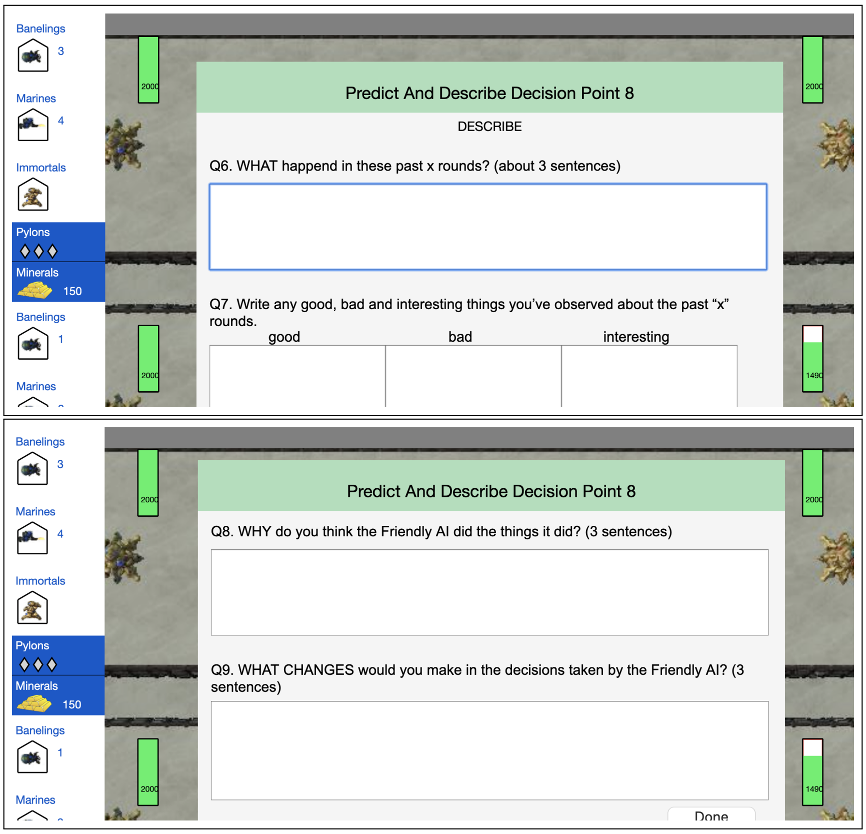
Predict And Describe Decision Point 8

PREDICT

Q1. Which lane will the Friendly AI build in?	Top	Bottom	None
Q2. Will the friendly AI make any marines?	Yes	How many?	No
Q3. Will the friendly AI make any banelings?	Yes	How many?	No
Q4. Will the friendly AI make any immortals?	Yes	How many?	No
Q5. Will the friendly AI make a pylon?	Yes	No	

Done

Fig. 14. Participants watch the game unfold until round 8, at which point the game pauses and the prediction questions pop up. The participant has to predict what they think the Friendly AI will do in round 9.



Predict And Describe Decision Point 8

DESCRIBE

Q6. WHAT happend in these past x rounds? (about 3 sentences)

Q7. Write any good, bad and interesting things you've observed about the past "x" rounds.

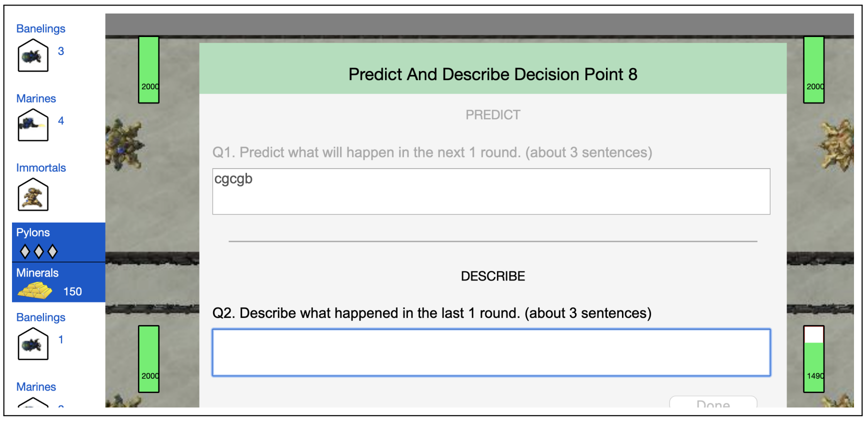
good	bad	interesting

Predict And Describe Decision Point 8

Q8. WHY do you think the Friendly AI did the things it did? (3 sentences)

Q9. WHAT CHANGES would you make in the decisions taken by the Friendly AI? (3 sentences)

Fig. 15. Having predicted what the agent would do (Figure 14), participants then see what it really did. Here, AAR/AI participants watch the game until round 9, at which point the game pauses again and the description questions pop up. The participant first describes what happened in round 9 (top), and then why (bottom).



Predict And Describe Decision Point 8

PREDICT

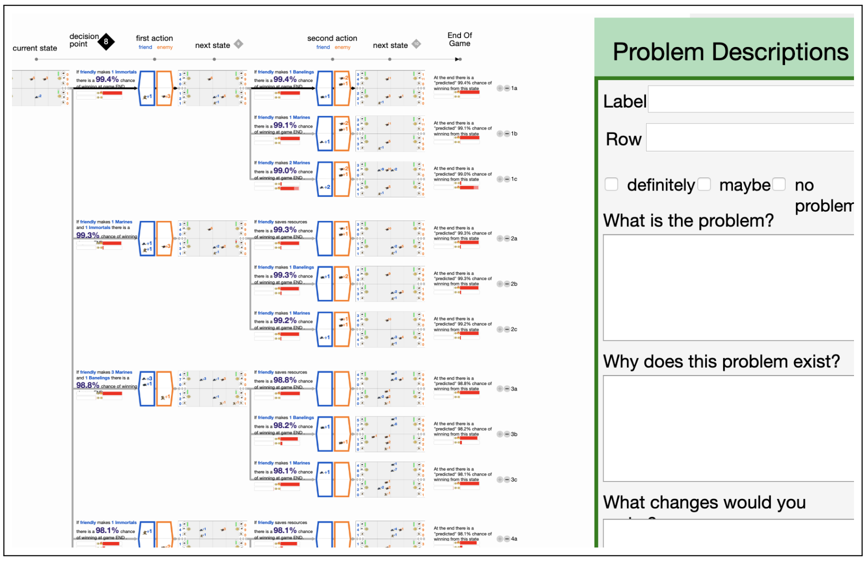
Q1. Predict what will happen in the next 1 round. (about 3 sentences)

cgcgb

DESCRIBE

Q2. Describe what happened in the last 1 round. (about 3 sentences)

Fig. 16. Having predicted what the agent would do (Figure 14), participants then see what it really did. Here, Non-AAR/AI participants watch the game until round 9, at which point the game paused again and the description questions pop up. The participant has to describe what happened in round 9.



Problem Descriptions

Label

Row

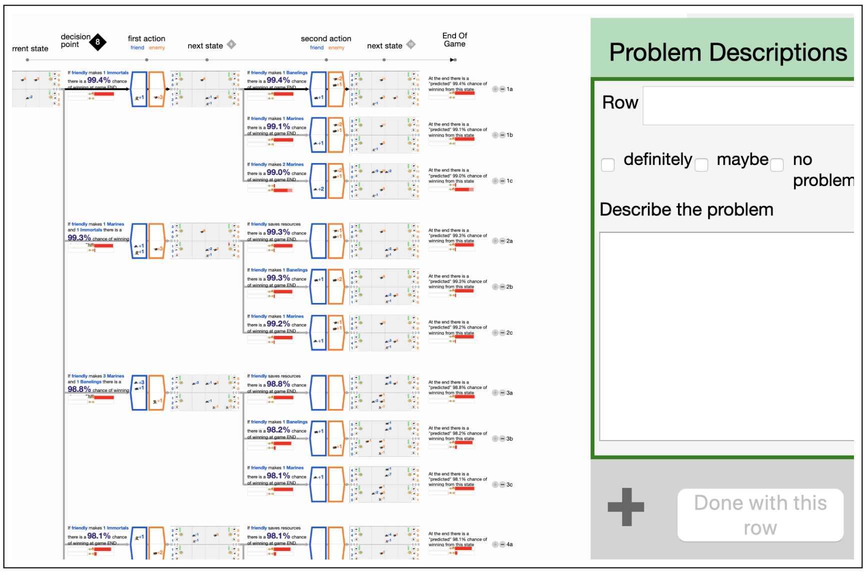
definitely maybe no problem

What is the problem?

Why does this problem exist?

What changes would you

Fig. 17. AAR/AI participants answered the AAR/AI questions of "What-Why-What changes" for each problem they found, in addition to giving their problem descriptions labels.



Problem Descriptions

Row

definitely maybe no problem

Describe the problem

Done with this row

Fig. 18. Non-AAR/AI participants had less structure than their AAR/AI counterparts, simply indicating the location and severity, then offering a description.

C APPENDIX C: MODEL-BASED AGENT ARCHITECTURE

This section describes details about the model-based agent we used for the studies. Figure 19 illustrates the overall architecture of the agent. We constructed a Minimax search tree by combining a *decomposed reward* deep Q-network (drDQN [32]), used for *action ranking* and *leaf evaluation*, and a transition model [41]. We describe the details of both in the next two subsections.

Model training details. Both the drDQN and transition models are neural networks that were pre-trained. They were trained separately and frozen while the agent plays. The training process continued until the agent achieves a high win percentage against a pool of agents or until resources were expended. This took around three days on a consumer desktop machine. In other words, the model-based agent in its entirety does *not* have a training process. Both networks have three fully connected layers and each hidden layer uses ReLU as its activation function. They have different numbers of neurons and output sizes, as indicated at the bottom of each layer in Figure 19. We used *mean squared errors* as the loss functions for both models. The learning rates of the drDQN and transition model were 10^{-4} and 5^{-4}, respectively.

C.1 Decomposed Reward Deep Q-Network (for Action Ranking and Leaf Evaluation)

The purpose of using *decomposed reward* deep Q-network (drDQN) agent instead of a standard deep Q-network (DQN) agent is that rather than only a single Q-value, it provides a more explanatory *vector* at the leaf nodes. In our case, we decomposed the Q-value (which is a scalar win probability) into an 8-element vector composed of the probability of each Nexus being destroyed and the probability of each Nexus having the lowest Health Points (HP) if the game reaches the tie-breaker. Therefore, we can compute the win probability for a single player by taking the sum of winning by destroying each opponent Nexus (2 elements) and by tiebreaking each opponent Nexus (2 elements). Further, the sum of all 8 elements should be 1.0, since it represents a probability distribution.

The drDQN model was pre-trained via pool-based self-play learning to achieve a reasonable high win-probability, which provides a meaningful decomposed Q-value vector for the leaf nodes of the Minimax tree. Because the size of the Minimax tree grows exponentially in its depth, we cannot expand it to the end of the game. To cope with this, we use the drDQN to prune the tree,

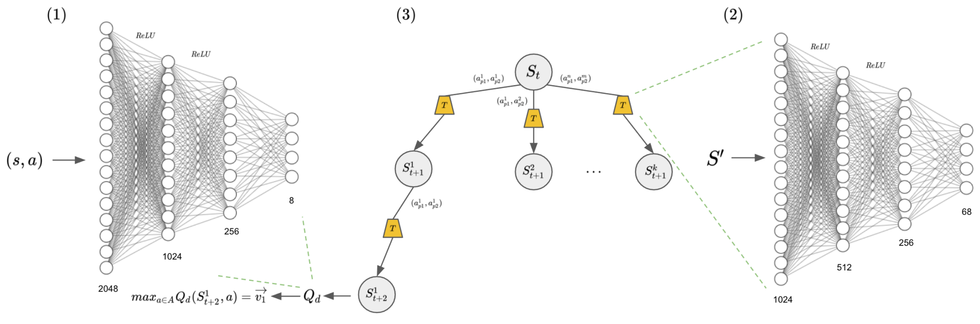


Fig. 19. The model-based agent consists of three parts. **(1, left)** The decomposed reward DQN (drDQN) model, which takes a state-action pair (s, a) as input and outputs a decomposed Q-value vector. **(2, right)** The transition model, which outputs the estimated next game state by taking the after-state S' as input. **(3, middle)** The tree structure that utilize (1) and (2) together. Here, the Minimax algorithm assigns the Q-value vectors computed at the leaf all the way to the root.

declining to expand actions that do not look promising (this is the *Action Ranking Function* referred to in Section 3.3). Therefore, evaluating leaf nodes with a neural network is important because it predicts the value of the future based on the leaf states—*without* expanding the tree further (this is the *Leaf Evaluation Function* referred to in Section 3.3). The decomposed Q-value function provides a discounted accumulation value vector predicting the future based on a state-action pair. The leaf node value vector $v = max_{a \in A} Q(s, a)$ is back-propagated back to the root node, where A is the action space and (s, a) is the state-action pair. Thus, we use the *same* drDQN twice in same agent, for both ranking actions and evaluating leaf nodes.

C.2 Transition Model

The transition model was also pre-trained by supervised learning based on the dataset from running the drDQN agent playing against an opponent pool which includes several types of agent. It takes an after-state S' as input which combines: the current state S, Player 1 (Friendly AI)'s action a_{p1}, and Player 2 (Enemy AI)'s action a_{p2}. Since actions in Tug-of-War correspond to an integer vector corresponding to the buildings the player is going to create, the action is deterministic and can be simply *added* up with the current state to produce an after-state corresponding to a tuple (s, a_{p1}, a_{p2}). It outputs an estimated state that describes the game at the next decision point. The estimated state has exactly elements as the input state has, which includes: mineral resources, number of buildings, number of troops in different regions for both lanes, Nexus HP, and the wave number.