AN ABSTRACT OF THE THESIS OF

Roli Khanna for the degree of Master of Science in Computer Science presented on June 11, 2021.

Title: Assessing and Finding Faults in AI: Two Empirical Studies

Abstract approved: ______________________________

Minsuk Kahng

With the advent of Artificial Intelligence (AI) in every sphere of life in today's day and age, it has become increasingly important for non-AI experts to be able to comprehend the underlying logic of how AI systems work, assess them and find faults in these systems, particularly when they are used in high risk scenarios such as in military strategies and medical applications. Recent developments to address the need to open the black boxes of these AI-powered systems have led to the emergence of AI explanations. There now exist myriad successful explanation methods and tools that attempt to explore and explain how AI systems work. However, a key problem with such work is the lack of a process that users can follow to navigate AI systems along with their explanation. This problem becomes increasingly evident with non-AI experts, due to their lack of context and depth of knowledge of the subject. To address this challenging problem, my colleagues and I propose a new process called AAR/AI or After-Action Review for Artificial Intelligence that aims to bridge this gap between AI systems and non-AI experts. AAR/AI, inspired by the US Defense debriefing strategy called AAR, is a process for understanding, analyzing and navigating sequential decision making environments. This thesis details two human-subjects studies my colleagues and I conducted, one qualitatively and the other quantitatively, to evaluate the effectiveness of AAR/AI in assessing an AI system and in identifying and localizing faults in it. The studies recommend that not only does AAR/AI assist non-AI experts to effectively navigate an AI system and keep their thoughts organized and logical, it also helps them identify and localize faults in it. Participants that used AAR/AI to localize faults did so with far more precision and recall than those that did not. I believe that this is a crucial step towards building democratic and explainable AI systems, and making them accessible to a larger audience that is not familiar with them.

Assessing and Finding Faults in AI: Two Empirical Studies

by

Roli Khanna

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 11, 2021
Commencement June 2021

Master of Science thesis of Roli Khanna presented on June 11, 2021.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Roli Khanna, Author

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Minsuk Kahng for guiding me through every step of my journey in graduate school. I will forever will be grateful for your support and knowledge. Next, I would like to thank the faculty members in the XAI research group, particularly Dr. Margaret Burnett, Dr. Weng Keen Wong and Dr. Alan Fern. I am grateful for the pearls of wisdom and the brainstorming sessions that helped me evolve as a researcher. A massive thank you is in order for my family, my friends and my support circle, thank you mother and father, for looking out for me and showering me with words of love and motivation from miles away, thank you Damanpreet Kaur and Gayathri Garimella for being the best roommates I could ever ask for, and thank you Amreeta Chatterjee for the constant support and encouragement, thank you Jonathan Dodge for being kind and helpful. Lastly, and most importantly, thank you Achint, my husband, for being my pillar and my cheerleader; I love you.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

Chapter 1: Introduction

1.1 Goal and Main ideas

Imagine “Jake”, a pilot in a big aviation company. There has recently been a plane crash, and Jake is a part of the debriefing committee that has to find out what led to the plane crash. A part of the plane’s control systems was powered by Artificial Intelligence or AI, and Jake is delegated the job to find out if the AI had a part in the decisions leading up to the crash. Jake is an expert and knows all about aeroplanes, but knows nothing about AI systems. This creates a peculiar problem where domain experts are expected to work with AI systems and reasonably understand them even though they have little to no context of how such systems work. This is one of the central problems that Explainable AI attempts to solve: how might we **explain** a complex AI system to domain experts that are non-AI experts, so that they may work with such systems?

Another primary premise that “XAI” or Explainable AI finds itself on is the need for accountability. An individual will likely “trust” or find an AI system accountable for its actions if they understand how it works, specifically in identifying where the system is usually right and where it usually makes mistakes.

For a domain expert like Jake, assessing an AI agent and finding faults in it is likely a complex task. In sequential decision making environments such as the one in Jake's case, specific tasks such as finding faults are multi-faceted and difficult to baseline. It is also hard for Jake to know if the fault he found is due to his lack of AI expertise or is genuinely the AI's fault. There is a lack of a proper process to guide such domain experts through an assessment task in sequential decision making environments such as the one Jake finds himself in.

Hence, my colleagues and I propose AAR/AI, or After Action Review for AI [60, 22]. AAR/AI is largely inspired by a debriefing method called AAR, or After-Action Review (AAR), formulated by the U.S. Army [63]. The military has been using AAR for decades to evaluate soldier performances and defense strategies. AAR was a success in many branches of military, and has also been adapted for other domains such as fire-fighting [39], medical treatments [80], and transportation services [58].

The traditional AAR debriefing process involves seven steps:

1. Define the rules;

2. Explain the objectives;

3. What was supposed to happen?;

4. What happened?;

5. Why did it happen?;

6. Formalize learning: How did it happen, or what changes could we make to fix it?;

7. Summarize.

Owing to the adaptable nature of AAR, and its support for sequential and complex decision making domains, we adapted AAR's seven-fold process into an AI assessment strategy. The adapted AAR for AI or AAR/AI process consists of seven steps:

1. Introduce the rules of the AI's environment;
2. Explain the AI's objective;
3. What did the evaluator think the AI would do?;
4. What did the AI do?;
5. Why did it do it?;
6. What changes would the evaluator recommend to fix it?;
7. Summarize.

My colleagues and I conducted two experiments to find if the AAR/AI process is indeed useful; the first is a qualitative think-aloud interview study that involves a reinforcement learning agent, an RTS (Real Time Strategy) game for the AI's environment, and paper prototypes for the AI's explanation. This experiment then informed a large scale quantitative study which delved deeper into the assessment task, and enquired if the AAR/AI process helped domain experts that are also non-AI experts localize an AI agent's faults.

This thesis outlines these two experiments in an attempt to understand the effectiveness of AAR/AI, and if it is the right assessment strategy for AI agents.

1.2 Thesis Overview

1.2.1 Qualitative study: Is AAR/AI helpful in the assessment process?

My colleagues and I designed a qualitative study that involved think-aloud interviews with paper prototypes and feedback forms to assess if AAR/AI was helpful for our domain experts that are also non-AI experts. It makes sense to dive into the motivations, intentions and information processing styles of people like Jake to understand the nuances behind assessing an AI agent for non-experts/domain experts.

Specifically, the research questions that we were looking to answer through this experiment were:

1. RQ1: What are the strengths and weaknesses of guiding Human assessment with AAR/AI?
2. RQ1a: When using AAR/AI for assessment, what do people need to make good assessment decisions?

For the purpose of this study, we chose a Real time strategy or RTS game called StarCraft II as the domain. RTS games are one of the most difficult to navigate AI domains owing to the complex and sequential decision making space, and are gaining popularity as an evaluative playground for AI agents. The target audience consisted of users that were not experts in Artificial Intelligence or Machine Learning (had not taken any course in AI/ML), and had expertise in RTS games. We recruited 11 such participants, and a facilitator conducted independent think-aloud interview sessions with each participant. In each session, the facilitator introduced the participant to the rules of the game (AAR/AI step 1-2), and made them watch a StarCraft II game where two AIs competed against each other. As part of the assessment task, we asked each participant the AAR/AI questions throughout the game (AAR/AI steps 3-6): 1. What did the AI just do? 2. Why do they think the AI did what it did? 3. What changes would they make in the decisions taken by the AI? We also asked the participant if they would allow the AI to make those decisions on their behalf, if they were playing the game. We showed the participants the AI's explanation for its actions in the form of paper prototypes, and gauged if the explanation component assisted the participant in successfully understanding and assessing the AI's actions.

The procedure of the study has been detailed below:

1. Explain the study and game rules (AAR: Step 1)

2. Have participant get used to user interface for video replay

3. Think-aloud tutorial

4. Explain the participant's objective (AAR Step 2)

5. Start the replay.

6. Think-aloud during the video (AAR: Steps 3, 4, 5): this will happen during the decision points in the video, not through-out the video.

7. Tutorializer will take notes on participants throughout. Tutorializer will say that "You can replay the game as much as you want to answer this question." at some point.

8. Post task questionnaire at the end of the study (Step 6)

We found that participants responded positively to the AAR/AI process, and self reported to finding it useful and helping them keep their thoughts "organized and logical". We also found that AAR/AI in conjunction with the AI's explanations was found to be particularly useful by the participants. These results recommend that AAR/AI is indeed an effective AI assessment strategy that can be incorporated into AI systems with explanations.

According to Bloom's taxonomy, which is a framework used by educators to categorize different levels of learning, there exist 6 levels of learning, with level 1 corresponding to basic understanding of a concept, and level 6 corresponding to a fairly advanced understanding. The results from the qualitative study recommended that participants reached up level 5 of Bloom's taxonomy in assessing the AI, and sometimes even level 6. The results (detailed below) were encouraging, and formed the basis for the forthcoming large scale quantitative study:

- The AAR/AI process helped participants in building their mental models about the AI, and contributed in keeping their thinking organized, structured and logical.

- Participants self reported to find the specific combination of the AAR/AI process, the explanations and the encouragement of active participation helpful in understanding the AI.
- The AAR/AI process encouraged participants to reason with the AI, and reasonably falsify its predictions.

1.2.2 Quantitative study: Is AAR/AI helpful in localizing an AI agent's faults?

In the qualitative study, we concluded that AAR/AI helped domain experts in assessing an AI agent. However, assessment can mean a lot of things: it can mean analyzing the accuracy of the agent, it can also mean its behavioral analysis, it can also mean finding all the instances where it went wrong.

We conducted formative small scale quantitative and qualitative experiments to inform the forthcoming large scale study to answer questions pertaining to the right assessment task and the right evaluation strategy. These included walking participants through our interface, giving them tasks that ranged from predicting the AI's actions to describing them, multiple post-hoc analysis of their feedback, and an entire overhaul of the experiment setup from in-person to an online process. All of these design changes and insights lead to the study design of the quantitative study, some of these major changes included:

1. Fine tuning the Research question (and corresponding major task) to identify and localize faults of the AI.
2. Testing the online setup, making it quantitative study ready
3. Embedding the AAR/AI process deeper in the explanations, and using a "guided search" mechanism to tackle the challenges we faced in the November study.
4. Including labelling faults.

Specifically, in one of formative experiments mentioned above, we found that participants largely categorized assessment into three categories: inferring the AI's actions, agreeing or disagreeing with the AI's actions, and finding flaws with the AI's actions. While directly interpreting and judging the AI's actions were common trends that we observed in recurring experiments, it was interesting to observe participants actively attempting to find faults in the AI and believing that there must be something fundamentally wrong with the AI's decision making process.

Finding faults in an AI is an important, and often overlooked, component of assessing an AI. Domain experts such as Jake actively look for places where the AI may have made the wrong decision, such as in the case of the plane crash. It then becomes essential to find a process that works specifically for finding faults in an AI. Hence, we fine-tuned the research question the forthcoming large scale quantitative experiment would address in order to solve a specific problem case and be able to reasonably quantify it:

RQ: How can we facilitate users to localize faults?

The quantitative study was designed such that all the participants had access to the same explanation interface, the same AI agents, and the same game. The study was conducted in a similar fashion as the previous qualitative experiment in that the target audience were domain experts that are non-AI experts, and the domain was StarCraft II. The study was designed by creating two randomly assigned groups, where the only difference (or the independent variables) was the presence or absence of the AAR/AI artefacts. The design was then mapped into a between-subjects and within-subjects experiment such that:

1. RQ1: Treatment 1: (Half the participants) use AAR/AI, with the same explanations. Treatment 2: (Half the participants) use Ad-hoc or non AAR/AI (control group), with the same explanations.

2. RQ2: Treatment 1: (All the participants) find Fault x, with the same explanations. Treatment 2: (All the participants) find Fault Y, with the same explanations.

The study was conducted entirely online over Zoom calls and a website, with one facilitator and 1-5 participants per session. The study contained pop-up questions (differing according to being in the AAR/AI or the control group), a game and explanation interface that participants could use to answer questions about the AI, and mark and describe the AI's faults in the explanation. The facilitator kept track of the actions taken by the participants through a virtual Dashboard.

The purpose of the experiment was to determine if AAR/AI helped participants in identifying and consequently localizing faults. Specifically, the experiment seeks to find how helpful AAR/AI is and what kind of faults (if any) does it help in finding. The procedure of the experiment is detailed below:

1. Pre-task-questionnaire for cognitive diversity.

2. Game and explanation tutorial.

3. Do a Practice Task, where the participants look for potential problems.

4. Start the main task (Loop over "x" DPs:)

    - Step 1 (Prediction): The participant completes the "Prediction sheet" to predict the AI's actions in the next 1 round.
    - Step 2: Watch the replay (1 round).
    - Step 3 (Describe): The participant completes the "Description sheet" (AAR/AI: "What/Why/How", Ad-hoc: "Summarise the Friendly AI's actions") to summarize what the AI actually did.
    - Step 4: The participant now has access to the AI's explanation, and begins finding faults as follows:

        (a) *Problem Localization:* Localize the "problem" in the AI's explanaiton that possibly caused the failure. The participant circles problem in the Explanation, and uses arrows for the cause of the problem.

   (b) *Problem Description:* The participant then completes a "summarising" questionnaire (AAR/AI: "What-Why-How", Ad-hoc: "Summarise the problems in the AI's reasoning") to explain the problem.

5. Post task questionnaires

The results from the experiment recommended that not only did the AAR/AI participants find more faults, they also did so more precisely. A summary of the results:

- AAR/AI participants had a higher recall of the bugs that they found than the control group.
- AAR/AI participants significantly outperformed the control group in all the types of bugs in the AI's explanation.
- AAR/AI participants also self reported more bugs, and did so with far more precision than the control group.
- AAR/AI helped participants abstract concepts about the AI.

1.3 Research Contributions

The contributions of this thesis, while specific to the Explainable AI community, can also be applied to similar decision making spaces. The experiments conducted recommend that AAR/AI is a powerful and useful assessment process that helped non-AI experts (that are domain experts) in finding and localizing faults in an AI agent. Contributions of these studies are multifold:

- *A new AI assessment process for non-AI experts:* AAR/AI is a debriefing process adapted to suit the needs of domain experts assessing AI systems, and it has proven to be useful in assessing and finding faults in such systems. The variety of advantages that AAR/AI offers in identifying and localizing faults can be attributed to its flexibility, simplicity and alignment with human information processing strategies. Additionally, AAR/AI is a democratic process that can be used to find faults in diverse AI explanation interfaces successfully.

- *System for integrating facilitation processes with explanation interfaces:* This thesis details the combination of AAR/AI embedded with consistent search and explanations for assessment and fault finding tasks in AI systems. Participants consistently fared better in assessment tasks with AAR/AI across different explanation interfaces, and self reported that AAR/AI helped them align their thoughts and think more logically.

- *Fault finding paradigm:* AAR/AI, coupled with consistent search and labelling was found to be particularly useful for finding faults, and can be used in the future to suit similar debugging needs in various AI explanation models. Notably, participants that used AAR/AI were 6 times more likely to find faults in an AI's explanation than those that didn't, and even did so more accurately.

- *Empirical experiment design demonstrating the effectiveness of a Human Debriefing process such as AAR/AI:* The between subjects experiment with and without AAR/AI supports recommended that participants that used AAR/AI found significantly more faults and did so more accurately. Moreover, empirical evidence of the within subjects experiment with different types of faults also recommended that participants with AAR/AI supports fared significantly better on all the types of faults. Such experiment designs offer methods and in-depth techniques to evaluate human debriefing processes such as AAR/AI.

1.4 Prior Publications

I note that this research is a result of multiple collaborations detailed in two publications. The qualitative experiment detailed in Chapter 2 is adapted from Mai et al [60].[1] I co-authored this paper with 12 other students and faculty. I was listed as a second author. For the quantitative experiment detailed in Chapter 3, it is adapted from a co-authored journal paper that is under review as of May 2021 [43].[2] I am the first author of this manuscript and collaborated with 11 other students and faculty.

[1] [60]: 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. Keeping It “Organized and Logical”: After-Action Review for AI (AAR/AI). Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI). ACM, 2020.

[2] [43]: Roli Khanna, Jonathan Dodge, Andrew Anderson, Rupika Dikkala, Jed Irvine, Zeyad Shureih, Kin-Ho Lam, Caleb R. Matthews, Minsuk Kahng, Alan Fern, and Margaret Burnett. Finding AI’s Faults with AAR/AI: An Empirical Study. 2021. (Under Review)

Chapter 2: Keeping It “Organized and Logical”: After-Action Review for AI (AAR/AI)

Explainable AI (XAI) is growing in importance as AI pervades modern society, but few have studied how XAI can directly support people trying to *assess* an AI agent. Without a rigorous process, people may approach assessment in ad hoc ways—leading to the possibility of wide variations in assessment of the same agent due only to variations in their processes. AAR, or After-Action Review, is a method some military organizations use to assess human agents, and it has been validated in many domains. Drawing upon this strategy, we derived an AAR for AI, to organize ways people assess reinforcement learning (RL) agents in a sequential decision-making environment. The results of our qualitative study revealed several strengths and weaknesses of the AAR/AI process and the explanations embedded within it.

2.1 Introduction

Consider people tasked with assessing AI systems—specifically those responsible for asserting that the technology is safe and regulation-compliant. An example of such a technology is a self-driving car, where the importance of evaluating its safety is paramount, especially since failures have such grave consequence that they are likely to wind up in court [17]. Assessing accidents caused by self driving cars increasingly tread into legal grey areas. Who is held liable? The driver for failing to react in time, or the company for delivering defective code? [65].

When considering the question of *how* to do assessment, we note that an intelligent agent interacts with the world in ways analogous to those of a human agent. Thus, perhaps we could adapt established techniques for evaluating the quality of *human agents* for use on AI. The technique we specifically refer to is the After-Action Review (AAR), devised by the U.S. Army in the mid-70's [63]. The AAR was a success in various branches of military, and has also been adapted for other domains including medical treatments [80], transportation services [58], and fire-fighting [39].

We term our adaptation AAR/AI ("AAR for AI"). AAR/AI is a *process* for *domain experts* to use in assessing whether and under what circumstances to rely upon an AI agent. We envision AAR/AI to be suitable for sequential domains, such as real-time strategy (RTS) games. It contains a series of steps the human takes to evaluate an AI agent and the explanations it provides of its behaviors.

To investigate AAR/AI, we created a custom game in StarCraft II (Section 2.4.1). Then, we created a reinforcement learning (RL) agent that yielded high-quality actions in the domain (Section 2.4.2). For this agent, we also devised an explanation for the model-based agent to show its search tree (Section 2.3.3). To evaluate the AAI/AR process in the context of this domain, explanation, and agent, we conducted a qualitative study designed to investigate these RQs:

RQ1 When using AAR/AI for assessment, what do people need to make good assessment decisions?

RQ2 What are the strengths and weaknesses of guiding human assessment in this way?

RQ3 What are strengths and weaknesses of search tree explanations, as we have designed them?

2.2 Background & Related Work

There are many papers describing the challenges of evaluating AI systems' quality (e.g. [12, 29]), including specific attacks (e.g. [24]). Rising to meet these challenges, approaches like DeepTest [90] attempt to utilize concepts from software engineering to improve testing of deep neural networks. In particular, they seek to measure and improve "neuron coverage" (proposed by Pei et al. [69], similar to code coverage). To accomplish this, they apply a series of transformations to the input, a form of data augmentation conceptually similar to fuzzing. Other approaches have transported software engineering concepts, such as test selection [30, 38] and formal verification [73]. However, these approaches are *system*-oriented in terms of exposing problems, not *human*-oriented by giving an assessor the tools to determine appropriate use for the AI.

2.2.1 People Analyzing AI

Human-oriented evaluation of AI is an active area of research, though much of it is at a different granularity than we needed. For example, Lim et al. researched how their participants' sought information in context-aware systems powered by decision trees. The result of their research was a code set of several "intelligibility types" describing the information. They discovered that their participants demanded *Why* and *Why Not* information, especially when the system behaved unexpectedly [54]. Using Lim's code set, Penney et al. studied how experienced RTS players looked for information when understanding and evaluating an "AI," but they found that participants preferred *What* information over *Why* information and that the large action space of StarCraft II led to high navigation costs, which meant missing important game events [71]. Dodge et al. analyzed how shoutcasters (human expert explainers, like sports commentators) assessed competitive StarCraft II players. They showed the ways that shoutcasters present information that they thought their human audiences needed [21]. Kim et al. gathered 20 experienced StarCraft II players to play against competition bots and rank them based on performance criteria. They noted how human evaluations of the AI bots differ from the evaluations used for AI competitions and that the human player's ability plays a huge role in their evaluations of the AI's overall performance and human-likeness [44]. These studies found how people evaluate an AI, but they did not present a *structured process* for assessment.

There are several models which consider system assessment in a human-oriented way; however, these works do not provide an assessment process for AI, but rather on whether humans will *adopt* systems or not. One such framework is Technology Acceptance Modeling (TAM) [20]. TAM can predict how well a system will be accepted by a user group and explain differences between individuals or subgroups. More recently, the UTAUT (Unified Theory of Acceptance and Use of Technology) model was proposed as an acceptance evaluation model [33]. These approaches could be used to examine the quality of AI systems, but they do not offer a concrete *process* for human assessors to enact.

2.2.2 People Explaining AI

Our process has an explanation explicitly embedded within it, so we briefly survey explanation strategies for AI. The primary purpose of explanations is in their ability to improve the mental models of the AI systems' users. Mental models are *"internal representations that people build based on their experiences in the real world"* that assist users to predict system behavior [64].

Explanations are also a powerful tool for shaping the attitudes and skills of users. One such example is Kulesza et al.'s proposed principles for explaining (in a "white box" way) machine learning based systems, wherein the system made its predictions more transparent to the user [48], which in turn improved the quality of their participants' mental models. Another study by Anderson et al. [4] provided insights on the varying changes in the mental models of participants with different explanation strategies of an AI agent.

US Army AAR Process
Introduction and rules.
Review of training objectives.
Commander's mission and intent (what was supposed to happen).
Opposing force commander's mission and intent (when appropriate).
Relevant doctrine and tactics, techniques, and procedures (TTPs).
Summary of recent events (what happened).
Discussion of key issues (why it happened and how to improve).
Discussion of optional issues.
Discussion of force protection issues (discussed throughout).
Closing comments (summary).

Table 2.1: Steps of the US Army AAR process [92].

Another direct consequence of altering the mental models of users is the improvement in their ability to command the system. According to a study by Kulesza et al. [50], participants with the most improved mental models were able to customize the system's recommendations the best, accommodating for the explanations that the researchers provided.

Explanations in the domain of AI agents in RTS games have been gaining traction over the years. In a study by Metoyer et al. [61], they present a format where experienced players played while providing explanations to non-RTS players. The strategy that expert players used while demonstrating how to play the game was found to be key to the explanation process. The study by Kim et al. [45] had experienced players play against AI bots in order to assess the bot's skill levels and overall performance. However, despite the existing research mentioned above, there is a dearth in literature concerning what humans really *need* in order to understand and assess such systems [68].

	AAR Debrief Steps	AAR/AI Questions Answered	AAR/AI Empirical Context
	1. Define the rules	How are we going to do this evaluation? What are the details regarding the situation?	We established the rules of evaluation and the domain (see Supplemental Materials).
	2. Explain the agent's objectives	What is the AI's objective or objectives for this situation?	We explained the AI's objectives for the situation (see Supplemental Materials).
AAR/AI Inner Loop	3. Review what was supposed to happen	What did the evaluator intend to happen?	We asked, *"What do you think should happen in the next three rounds?"*.
	4. Identify what happened	What actually happened?	The participant watched three rounds. Then, we asked, *"Could you briefly explain what actually happened in these past three rounds?"*.
	5. Examine why it happened	Why did things happen the way they did?	We asked, *"Why do you think the rounds happened the way they did?"*. Next, the participant summarized anything good, bad, or interesting on an index card. Last, we provided the participant the agent's explanation for that decision (Figure 2.1), and requested they, *"Think aloud about why the Friendly AI did the things it did."*.
	6. Formalize learning (end inner loop)	Would the evaluator allow the AI to make these decisions on their behalf? What changes would they make in the decisions made by the AI to improve it?	We asked three questions: *"Would you allow the AI to make these decisions on your behalf?"*, *"What changes would you make in the decisions made by the AI to improve it?"*, *"Would you allow the Friendly AI to make this category of decisions on your behalf?"*.
	7. Formalize learning	What went well, what did not go well, and what could be done differently next time?	The participant completed a post-task questionnaire (see Supplemental Materials).

Table 2.2: How AAR/AI (right two columns) adapts the original After-Action Review steps (left column). The "Empirical Context" column explains how we realized it in our empirical study. Note that steps 3-6 form an "inner loop" that we repeated every three decisions. The parts outside the inner loop are documented in our Supplemental Materials (tutorials, questionnaires, etc), so we describe them only briefly here.

2.2.3 After-Action Review

To structure our assessment method, we turned to processes that have been used for humans to assess *other humans*, including Post-control, Post-Project Appraisal and After-Action Review (AAR) [82]. Our criteria for the process to use as our basis included: (1) have a structured and logical flow, (2) be well established, and (3) be suitable for evaluation *during* a task, not just useful at the end of a task. We selected the AAR method as the one that best fulfilled these criteria.

AAR is a debriefing method created by the United States Army, and it has been used by military and civilian organizations for decades [79], to encourage objectivity [58]. The purpose is to understand what happened in a situation and give feedback, so people can meet or exceed their performance standards by going through a structured series of steps shown in Table 2.1.

The AAR process was primarily used as a method to provide performance feedback after soldier training sessions. Before starting an evaluation session, the leader (a designated individual across all sessions) performs groundwork to collect and aggregate data from the session for further analysis. The leader enters the session with a pre-planned mechanism to collect data and begins the session by reiterating the objectives of the analyzed exercise. From there, the leader asks a series of open-ended and leading questions about what happened during the training session, making sure to encourage a diverse range of perspectives. These responses are then filtered into a recapitulation that the group collectively agrees on, and the discussion is shifted to scrutinizing any shortcomings in performance. This is followed by brainstorming solutions to avoid or improve responses to problematic outcomes. The session concludes by delineating an action plan to adhere to for future training [92].

AAR showed effectiveness for combat training centers [79], and the military still uses it, with a recent investigation of current methodologies for simulation-based training [31]. Outside military applications, AAR has been used in other domains, from medical treatment [74, 80], emergency preparedness [19], and response [39, 53]. The closest research to ours discusses how AAR will be different for manned-unmanned teams, but focused on the technologies needed to support the AAR process, not the process itself [10].

2.3 The AAR/AI Process

Our After-Action Review for AI (AAR/AI) is an assessment method for a human assessor to judge an AI. We base the steps of our method from Sawyer et al's DEBRIEF adaptation from the Army's AAR [80]. In their adaptation, they Define rules, Explain objectives, Benchmark performance, Review what was supposed to happen, Identify what happened, Examine why, and Formalize learning. Table 2.2 outlines our AAR/AI adaptation.

The original AAR method is a facilitated, team-based approach, but our AAR/AI method is for an individual reviewing, learning the AI's behavior, and assessing its suitability [82]. The outcomes are different for the approaches: AAR aims for transfer of knowledge within a team, and AAR/AI aims for individual acquisition of knowledge and assessment of an AI. These two primary differences between AAR and AAR/AI are what generated the specific ways AAR/AI (Table 2.2's columns 2 and 3) carries out the original method's steps (Table 2.2's column 1).

2.3.1 AAR/AI: Defining Rules & Objectives

A facilitator starts each session with a tutorial on the user interface, domain, explanations, and the objectives of the assessment (Steps 1-2, Table 2.2). This contextualizes the discussion in terms of what the assessor is supposed to do and the agent that they are assessing. After that, the facilitator begins the AAR/AI "inner loop" (discussed next), and after every loop is done, the assessor completes a questionnaire.

2.3.2 AAR/AI's Inner-Loop: What, Why, How

During each iteration of the inner loop, the facilitator asks the assessor, "*What was supposed to happen?*", "*What happened?*", "*Why did it happen?*", "*How can it be improved?*" (Steps 3-6, Table 2.2). The assessor also summarizes what happened in the past three rounds and writes down anything they observed that was good, bad, or interesting on an index card. At Step 5, we provided the assessor with the AI's explanation for the most recent round, and asked to explain why the AI did the things it did, according to the process in Table 2.2. Following this, to formalize learning about this particular decision, the facilitator asks the assessor the questions listed in Table 2.2 step 6, (e.g. whether they would allow the AI to make these decisions on their behalf). Thus ends the inner loop, which would repeat until the end of that analysis session.

2.3.3 AAR/AI: Explanation Component

AAR/AI evaluators, like the AAR equivalent, require information on what happened, so our process requires an Explanations Component, since the evaluators not only must they know *what* happened, but the agent must be able to explain *why* it performed an action. In our evaluation study, we used a model-based agent, so we prototyped a model-based explanation.

A model-based agent (and its explanation) offers the benefit of explicitly representing the future states the agent is trying to reach or avoid. Our model-based explanation captures the agent's search tree, shown in Figure 2.1. We described the search tree to participants as, *"...a diagram of decisions, where the Friendly AI decides what actions or decisions it must take to complete a round in the game."*

The explanation lays out the agent's "explanatory theory" [85] of how the game could play out in different situations. In essence, the theory's "constructs" of that theory are: game states, roles (e.g. friends or enemies), actions available to various roles, and (estimated) values of different states and actions.

In Figure 2.1, the root node (region 1) shows the current game state and its estimated value. One layer down (region 2) shows the 4 best actions available to the friendly AI in the current game state–and their values, as estimated by the

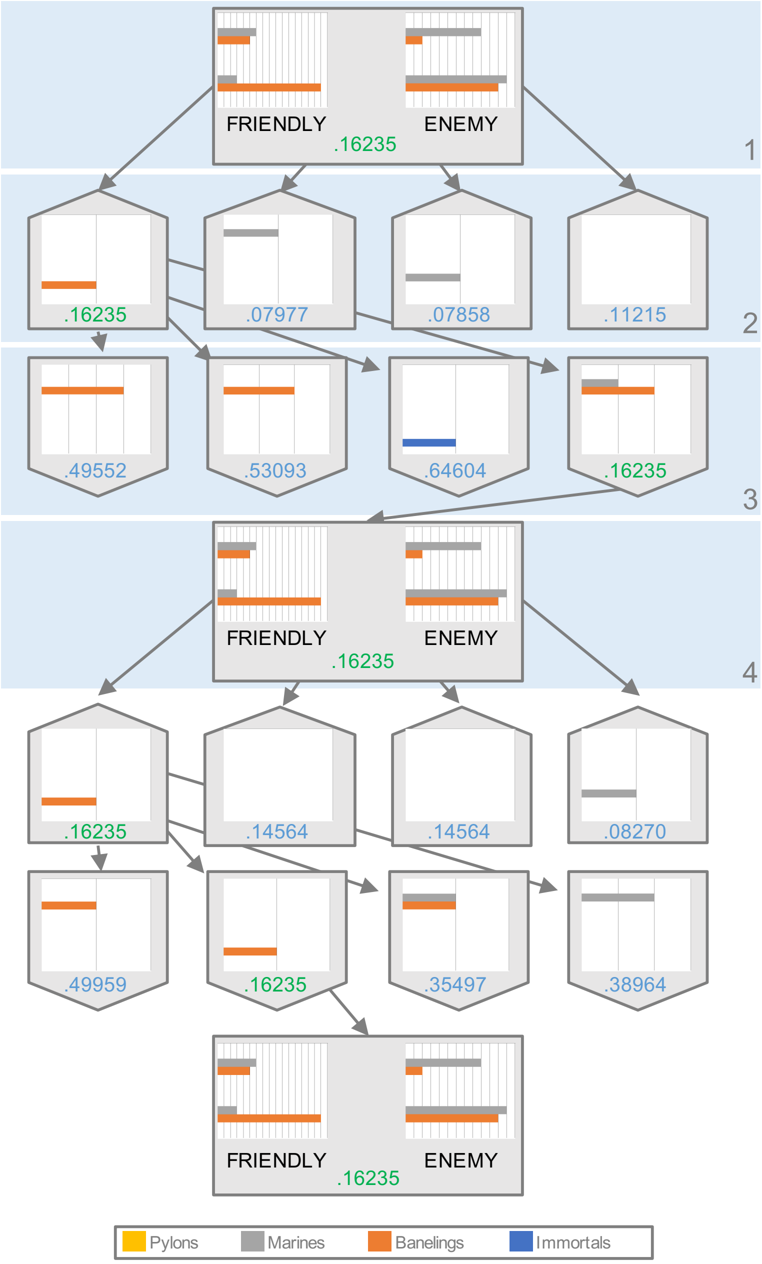


Figure 2.1: Search tree explanation for decision point 22. Blue background boxes show: (1) game state at decision point 22, (2) top 4 most rewarding actions, as estimated by the AI, (3) top 4 most rewarding actions *for the enemy* in response to its "best" action, as estimated by the AI, and (4) predicted game state at decision point 23. Our agent searches to depth 2, so the explanation includes another turn of search from the *predicted* state (box 4). Note that all states below the root (box 1) are predicted by the agent. Green highlighted numbers indicate parts of the principal variation.

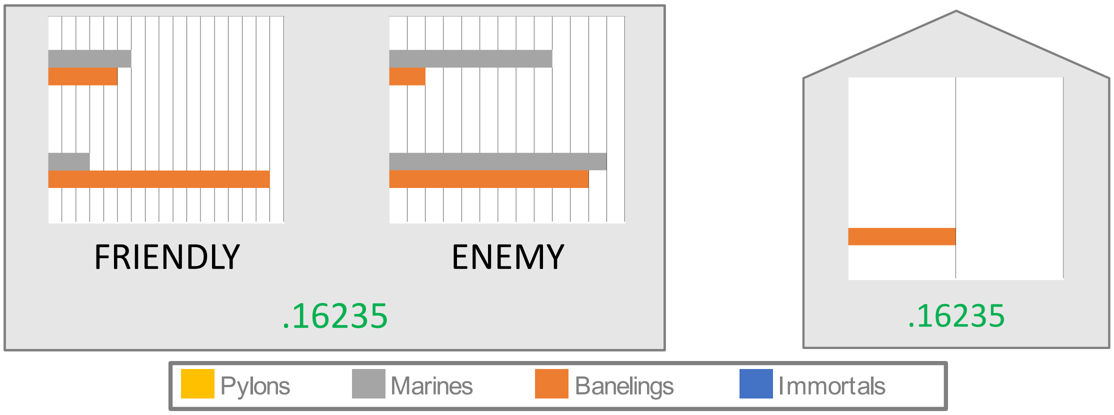


Figure 2.2: *Left:* An example of State node presentation. Each bar shows a number of unit production facilities for each lane and type. Here, the Friendly AI has 6 marines and 5 banelings in the top lane—with 3 marines and 16 banelings bottom. *Right:* An example of Action node presentation. Similar to the state, bars are split by lane and by unit. Each node is given with the agent's estimate of the win probability associated with that action (number at the bottom.)

agent based on the tree expansion. The third level of the tree (region 3) shows actions available to the opponent—again, the 4 best actions and their values as estimated by the agent. The fourth level of the tree (region 4) shows the *predicted* state that the agent thinks will ensue based on the current state, taken together with the simultaneous actions from itself and the opponent. From that level, the agent performs another round of search in the same way, resulting in an agent that looks ahead 2 rounds. Each node is shown with the state or action that node depicts, alongside the estimated value of that state/action, shown with more detail in Figure 2.2. If that value is part of the principal variation (colloquially, the most likely trajectory given "optimal" play from both sides), its value is shown in green instead of blue.

2.3.4 AAR/AI's Artifacts

Part of AAR/AI involves creating materials to help keep everyone on task during the assessment. The US Army AAR uses cards in order to log observations [92], though the information collected is largely focused on personnel and their positioning. Since the AI performs within the RTS domain, we turned to how professional shoutcasters analyze AI, like AlphaStar [86]. They used formatted text for actions that they found "good," "bad," or "interesting," which we replicated in the AAR/AI's index cards. This prevents assessors, regardless of the AI's use, from relying on memorizing when a decision is good or not. By using such written artifacts, the AAR/AI process has the benefit of gaining retrospective feedback on process or explanations. Further, artifacts like these can assist in comparing the assessment results from multiple different individuals.

2.4 Empirical Study: Methodology

To inform our design of AAR/AI, we ran an in-lab think-aloud qualitative study. One goal was to investigate what participants needed when doing AI assessment, alongside strengths and weaknesses of our process. Additionally, since the AAR/AI process embeds an explanation, our other goal was to obtain feedback about the model-based explanation strategy we described in Section 2.3.3.

We recruited 11 students at Oregon State University who had not taken classes in AI or ML. Since our game is based on StarCraft II, we recruited those familiar with real-time strategy games, to ensure that participants could understand the game sufficiently to assess the AI.

A researcher facilitated for the participant (assessor) during the AAR/AI process, starting with a tutorial about the interface, domain, and task (Steps 1/2). Since each session was limited to 2 hours, we wanted to ensure that each participant reached the end of the replay and had time for our post-task questionnaire. Thus, we decided to have them analyze every third decision point out of the 22 available, including the last one (e.g. 3,6,...,21,22). This allowed up to 5-7 minutes for each iteration of the AAR/AI inner loop—though it was rarely necessary to enforce limits during the study.

At each iteration, the researcher asked the assessor a structured series of open-ended questions to elicit their thoughts as they performed their assessment of the AI's actions (Steps 3-6). Additionally, the participant wrote on index cards (Section 2.3.4) to help them formalize thoughts and offer the option to refer back to previous ones.

Upon completion of the task (Step 7), we asked: *"Did the process of the questions I asked you help you understand and assess the AI better?"*, *"Do you think the AI's diagrams have enough detail?"*, *"Would you prefer the width of the diagram to be narrower or wider? Or do you like the way it is?"*, *"What kind of actions would you have liked to see on the diagram?"*, and *"In the main task, did you find these cards useful?"*. Finally, we compensated participants $20.

Each session spent ˜30 minutes for the briefing/tutorial (pre-task), ˜50 minutes on the inner-loop (the main-task), and ˜25 minutes on the post-task questionnaire. This timing was consistent with Sawyer et al.'s recommendations (25/50/25%, respectively) [80].

2.4.1 The Domain

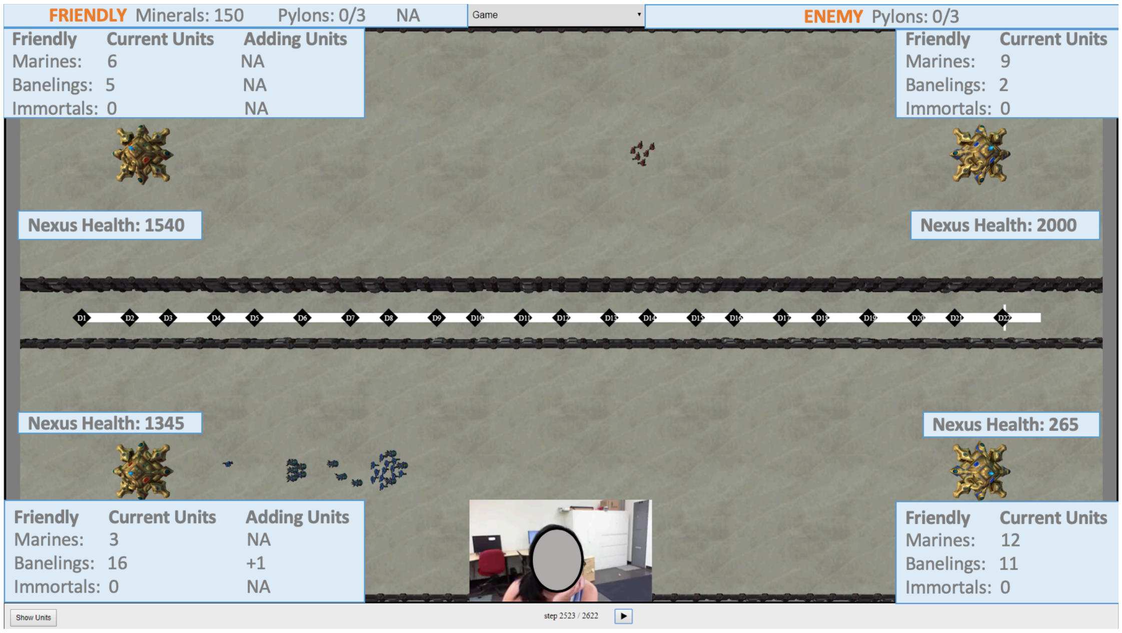
StarCraft II is a popular Real-Time Strategy (RTS) game that offers hooks for AI development ([93, 94]) and a flexible engine for map creation[1]. The game used for this study is a tug-of-war like customized game based on StarCraft II, shown[2] in Figure 2.3. The objective of the game was to destroy either of the opponent's Nexus in the top lane or bottom lane . If no Nexus is destroyed after 40 rounds, the player whose Nexus has the lowest health will lose.

At every round of the game:

- Each player receives income (100 minerals, +75 per pylon)
- The player chooses to build any combination of unit production facilities (i.e. barracks) which will exist for the next round, subject to the following constraints:
    1. Total cost cannot exceed current mineral count
    2. Players are only allowed to build in *one* lane at a time
    3. Players do not know the opponent's action until both actions are finalized

[1] Many map creation resources are available at places such as [89].

[2] Materials to replicate this state are freely available in our Supplementary Materials.



FRIENDLY Minerals: 150 Pylons: 0/3 NA

ENEMY Pylons: 0/3

Friendly	Current Units	Adding Units
Marines:	6	NA
Banelings:	5	NA
Immortals:	0	NA

Friendly	Current Units
Marines:	9
Banelings:	2
Immortals:	0

Nexus Health: 1540

Nexus Health: 2000

Nexus Health: 1345

Nexus Health: 265

Friendly	Current Units	Adding Units
Marines:	3	NA
Banelings:	16	+1
Immortals:	0	NA

Friendly	Current Units
Marines:	12
Banelings:	11
Immortals:	0

Figure 2.3: Game screen at decision point 22. Note the text boxes offering state information (current units, nexus health, etc) as well as action information (adding units).

- Players spawn units equal to the total number of unit production facilities currently held (i.e., 5 barracks implies 5 marines)

Each round, both players choose which lane to build in and the number of unit-producing buildings to spend resources on for each of 3 unit types, who share a rock-paper-scissors relationship. **Marines** (50 minerals) are low health units that attack in small quick shots. They are effective against immortals. **Banelings** (75 minerals) are medium health units that attack by exploding on contact. Banelings are effective against marines. Lastly, **Immortals** (200 minerals) are high health units that attack in large slow shots. Immortals can inflict significant damage on a Nexus. Players may also choose to build a pylon to increase their income per round. The maximum number of pylons they can build is 3, and the cost of a pylon increases each time one is purchased. Note that an action in this context is essentially an integer vector, meaning the branching factor is combinatorial with respect to minerals possessed.

Once a unit spawns, the players can no longer control it; they will move toward the enemy Nexus and attack any enemies along the way. Also, units *always* spawn at the same location each wave.

2.4.2 The Agent Implementation

The agent is model-based, so it has access to a transition function that maps a state-action tuple to the successive state. Applying the transition function allows the agent to expand a move tree, and perform minimax search[3] on it. The system uses three learned components (all represented by neural networks): the transition model, the heuristic evaluation performed at leaf nodes, and the action ranking at the top level.

The heuristic evaluation function estimates the value, or quality, of non-terminal leaf nodes in the search tree. This function is necessary to address the depth of the full game tree, since the search will rarely be able to expand the tree until all leaf nodes are terminals. The action ranking function provides a fast estimate of the value associated with taking each action in a state. This function is necessary to address the large action-branching factor by only performing the more

[3] For more information on game tree search, see Russell and Norvig, Chapter 5 [78].

Code: Description	**Example**	#
Explanation Overall Quality: Participant found the explanation useless or helpful in a general sense (very vague), or in determining reasons for actions in the decision process (clarity, or lack thereof).	P2: *"I think it's pretty easy to understand, like, after looking at for a little while."*	8
Diagram Color Coding: Participant comments on the manner in which an explanation object is colored.	P17: *"The color coding is okay. Um, it's pretty distinctive. Um, I don't know if the background is gray or- and even the marines are gray... it was confusing because if it was different color"*	4
Changing Diagram Data Contents: Participant talks about changing data in the diagram (such as changing the node definitions, changing the key, etc). This is NOT about showing an action/state node that is not present.	P18: *"How much minerals it has, something like that. I would like that to be represented on the diagram."*	7
Diagram Node Contents: Participant wants the diagram to contain more/fewer nodes, (e.g. wishes to interactively expand a node, request a specific action be examined, or have a "wider/narrower" tree) OR thinks it contains the right amount.	P11: *"I would just have more options available, you know. ... So sometimes, there are missing... missing options which should be taken."*	16
Diagram Glyph Presentation: Participant comments on the glyphs for the action or state nodes, referring to the way the state information is presented in the glyph	P10: *"As the number of units goes on increasing, the line goes on increasing. And that is why it's short. That's clear, but vertical lines are... if it would have been 1, it would have been great. Just 1 line."*	6

Table 2.3: Helpful/Problematic code set for the *explanations*. Frequencies are from three post task questions centered on the explanation and its contents.

expensive tree expansion under some number of top-ranked actions to improve estimates (similar to AlphaGo and AlphaZero [83, 84]). A big difference, however, is that our system uses a learned transition model, due to the stochastic and complex nature of the transitions between states, whereas Silver et al.'s used a perfect move-transition model (e.g., Chess's deterministic rules).

2.4.3 Analysis Methods

To answer **RQ2** and **RQ3**, two researchers applied content analysis [37] to the coded statements from the post-task questions about helpful or problematic elements of the process or explanations, resulting in the code set in Table 2.3. The two researchers coded 21% of the data corpus separately, achieving inter-rater reliability (IRR) of 82.4%, computed via Jaccard Index [40]. Given this level of reliability, they then split up the remaining coding.

To answer **RQ1**, we drew from a code set that Dodge et al. used in their StarCraft II study, who had adapted from Lim et al.'s work [55, 21]. Dodge et al. also added in a "judgment" code, which the AAR/AI needed because of the nature of assessment. Individually, the two researchers coded 20% of the data corpus, achieving an inter-rater reliability (IRR) of 76.4%. Given this level of reliability, they then split up the remaining coding.

2.5 Results

Our explanation strategy consists of three components: the AAR/AI process itself, the specific explanation content and presentation, and a "keep the user active" tactic to facilitate their learning of the agent's behaviors. Accordingly, this section has three parts: 1) how the AAR/AI process affected participants' understanding, 2) how the Explanation (tree diagram) content and presentation affected participants' understanding, and 3) how the integration of all three elements of our strategy affected participants' understanding.

2.5.1 Results: The AAR/AI Process

The goal of our project's explanation strategy is to enable participants to understand how the AI agent is "thinking" well enough to evaluate how suitable the AI agent is for different situations that arise. In essence, our explanation strategy aims to help people build mental models. In this subsection, we consider what the AAR/AI process itself brought to our participants' mental-model building.

Many of the participants commented on how AAR/AI's "structuredness" helped their understanding by keeping their thinking organized, structured, and/or logical. (Only one participant said it was not helpful, but this was because they believed that with their experience in RTS games, they already understood the AI's behavior without the need of any assistance.) For example:

P8, PTQ 2: *Uh, yes, I would say AAR/AI was helpfu. It definitely directed me towards what I should be paying attention to.*

Level: [9]'s Description	**How it applies to understanding the AI**	**Examples from our participants**
1. **Remembering**: Have students acquired the ability to correctly recall information?	Participants *recall* domain information, such as game rule(s), what an agent can do with particular game units, etc. (Supported by AAR/AI's questions about the game.)	+P20: *"It'd probably buy another baneling... to counter the marines..."*
2. **Understanding**: Can students understand information they have learned to recall?	Participants *understand* the domain information provided. (Supported by AAR/AI's "What" and first "Why" question.)	+P8: *"...you (the AI) don't necessarily know which lane they're coming through... it's not much of an informed decision until the first round happens."*
3. **Applying**: Can students apply their newly learned knowledge?	Participants *apply* the explanation of the AI to the game. (Supported by second "Why" question.)	+P2: *"I ...like it how (the explanation diagram) is, because like I could try to draw my own conclusions from it rather than just like 'oh this is just what happened'."*
4. **Analyzing**: Can students see patterns and make inferences about a problem?	Participants *analyze* the AI's problems in the game, and reason about solving them. (Supported by the prediction task and the "What changes would you make" question.)	+P2: *"So the bottom one did pretty well like overpowering the enemy AI and even attacking nexus, lowering its health while the top one, the enemy AI did a better job sending more marines and the friendly AI sent banelings which got overpowered by the marines."* +P19: *"So we have almost same health on top and bottom. So, to defeat us, they have to focus on either one. So I guess they will focus bottom, because they have to save them at the time. I guess we have to use minerals to buy immortal here, so that we can save ourselves and at the same time, kill the enemy."*
5. **Evaluating**: Can students take a stand or decision, and justify it?	Participants *evaluate* the AI agent, and judge if they would allow the agent to make decisions on their behalf in this or similar situations. (Supported by the "Would you allow..." question series.)	+P5: *"Producing these banelings (in both) lanes allowed nexus damage bottom lane, and then having the one or two marines do consistent damage on the nexus really took down the nexus health, so that was actually a really good decision."* +P20: *"This is gonna be sad. Yep. It's all downhill from here. (after watching the replay) Uh, the friendly AI lost, uh, due to their misinvestment in the top row, and only increasing their baneling count, which only works at melee range which is ineffective to marines if there's already a baneling wall in front of them."*
6. **Creating**: Can students create a new point of view?	Participants *create* new points of view by generalizing upon, abstracting above, or recommending differences in the AI's behaviors.	+P14: *"Well, the enemies will invest in banelings, and I feel that the friendly's will invest in marines, especially more in the top row, since it is more damage..."* +P21: *"I would consistently save a small quantity of minerals each round, rather than trying to save them all in a single round."*

Table 2.4: Bloom's taxonomy levels participants achieved in learning the agent's behavior.

P18: *I could think what it should improve on and why the previous round happened the way it did. So, when those questions were broken down... Really helped in following the game.*

P14, PTQ 2: *...it categorized the flow of logic that we should've had in analyzing the prediction and what actually happened, so it kept it more organized, and therefore, more logical.*

P17, PTQ 2: *I know it was too much information ... it helped me understand it better. ...it just helps me ... to understand it better, and makes it more logical.*
To understand the level of our participants' mastery of understanding the agent, we applied Bloom's Taxonomy [9], which is a framework used by educators to categorize the different levels of learning. The taxonomy has six levels [6], ranging from basic understanding of a concept (level 1), through a fairly advanced understanding (level 6). Each level requires learners to engage with a higher level of abstraction than the last. The application of Bloom's taxonomy to our context is detailed in Table 2.4.

As Table 2.4 shows, subsets of participants showed mastery of every Bloom's level. In fact, all participants achieved Bloom's Level 5 at least once during the study. Further, all except one of the participants achieved Bloom's Level 6 at some point.

Bloom's Level 5 is of particular interest to our project: it is the level of understanding that allows evaluation. Evaluation is a form of problem-solving—working out whether the AI agent is "capable enough" for a particular situation—and problem-solving greatly benefits from diversity of thought [28]. Although most research into diversity of thought is in the context of team problem-solving, at an abstract level it amounts to bringing diverse perspectives to a problem (e.g. [28]).

To consider whether the AAR/AI process was able to elicit diverse perspectives from our participants, we turned to the Lim-Dey intelligibility types, which we used as a codeset for our qualitative coding (Table 2.5). As the results show, each of AAR/AI steps guided participants' thinking (according to their self-reports) toward different Lim/Dey perspectives [54]. For example, the first question guided most participants to focus on "What Could Happen," the second on "Input" and "Output" types of information, and the last on "How To" information. Since other research has shown each intelligibility types has its own advantages and disadvantages, we see the diversity of perspectives that AAR/AI seemed to elicit as a particular strength of AAR/AI [6].

2.5.2 Results: Explanation Content and Presentation

The tree diagrams provided participants with a more global view of the agent's decision process, supplementing the local-only "right now" view provided by the game state. As two participants put it aptly:

P2, PTQ 7: *I kinda of like it how it (explanation diagram) is, because like I could try to draw my own conclusions from it rather than just like 'oh this is just what happened'.*

P14, Artifact PTQ: *(In the game state)... difficult to grasp the whole situation, so having the graph gave me a chance to get my footing on overall trends* *and options.*

This way of using the explanation was a theme which was in a post-task response from another participant:

P17, Artifact PTQ: *The diagrams used to make it easier also helped to understand the predictions. To look at one thing from many angles and make appropriate predictions.*

However, a pitfall some participants fell into was extrapolating too much information from the tree diagrams. Several participants seemed *certain* about the agent's long-term plan, which was troubling because the explanation did not make such a plan explicit—if the agent even had one.

P21, DP18 Why 1: *At this point, I feel certain that the friendly's trying to destroy the bottom nexus of the enemy.*

P10, DP21 Why 2: *I think it's because it was a whole game plan from the beginning. ... like from the beginning of the bottom lane, the friendly AI started increasing the troop numbers.*

However, the explanation could not possibly have shown a many-step game plan, because the agent was only looking head two states.

Another participant also expressed difficulty in seeing long term strategies, but for a different reason—granularity mismatches between moves, tactics, and strategies:

P20, Artifact PTQ: There are subtasks and decisions that go into making a strategy and not being able to see this had me make less informed assumptions about the future decisions.

Part of P20's complaint above also was a desire for more information, and this issue arose in multiple ways. One participant wanted the explanation to show an estimation of the resources available to both the friendly AI and its opponent: **P20, PTQ 3:** *I would enjoy to see ... the AI's, calculation of their minerals. ...further extrapolation of getting this many more minerals allows you to buy these units. ...Because in RTS games you think about is the enemy's resources as well and how to manage those as well as your own.*

2.5.2.1 How Much More/Less/Different to Show?

Addressing the previously described requests for more or different information is not straightforward. With the agent considering combinatorial action spaces, showing the full search tree all at once would have been too large for humans to process. Thus, we needed to choose a smaller set of noteworthy actions to show—but which ones and how many?

To situate the "which" question, the explanations participants saw showed only four actions (recall Figure 2.1). Some participants thought there should be more and/or different ones. For example:

P5, PTQ 3: *... since there are only four options ... if it was a possibility for more options 'cause there was definitely more possibilities.*

However, these four options were only "top" as per the agent's estimations, which may not have been the right four:

P5, PTQ 3: *I would think the AI would have the best four, which it didn't* *have the best four.*

One participant proposed also showing the *worst* possible choice:

P20: *I'd like to see ... what the friendly AI thinks is the ... choice that would give them the least chance of winning as well as their greatest chance of winning...*

As to how many actions to show, seven of the participants indicated that they liked the tree—but one wanted a smaller one, and three wanted a larger tree.

P8: *I liked the way it is. It's easy to read.*

P21, DP18 Why 1: *I do not have any problem with narrow diagram...*

P11: *I would just have more options available...*

Finally, one wanted everything—which is of course an infeasibly large amount of information to present statically, but might be possible to at least navigate via dynamic mechanisms:

P5: *All the possible actions and all possible outcomes.*

2.5.2.2 Explanations as Axioms and Theorems

In the explanation trees, leaf nodes used a neural network to evaluate the quality of states. These estimates were, in essence, axiomatic and the minimax search that proceeds atop those values are akin to theorems. Thus, if the axioms hold true, then the theorems were true. Some participants were open to "grant the axioms."

P14, DP21 Why 2: *I mean because, those are the ones with greater scores. So I guess that is why it chose those decisions.*

Others did not grant them and found themselves not understanding or possibly disbelieving parts of the diagram.

P10, PTQ 3: *I think diagram needs improvement, because those are not that clear at some times. ...It does have enough details, but the decisions were, not made... according to the diagram.*

In fact, one participant identified the issue quite well: that the win probabilities have no clear provenance.

P8, PTQ 7: *... If there's any easy way to say why it came up with these numbers... there were several steps that I just didn't know why it was taking that action...*

We found that RTS experience seemed to be a potential driver for rejecting the heuristic evaluation function, with P5 and P20 being particularly critical of the agent's decisions:

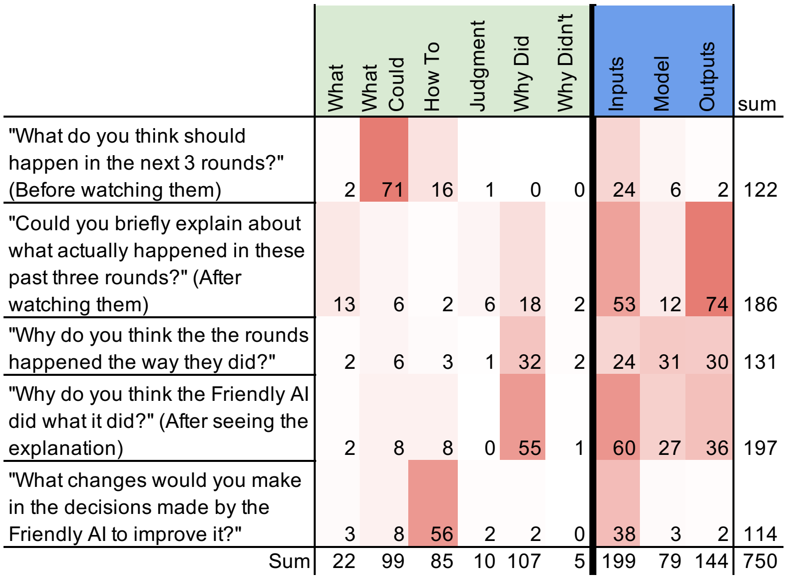
P5: *Wow, rewards went down... A baneling is better than a marine by rewards points, but there's clearly a better answer.*

Those with less RTS experience seemed less critical of the agent's explanation, but they still compared the agent's actions to the tree:

P14, Artifact PTQ: *Information didn't always line up with what occurred. Therefore, it gives a false belief on what/how the AI is doing.*

2.5.3 Results: Combined Explanation Strategy

Some results seemed directly tied to the integration of all three aspects of our explanation strategy: the AAR/AI process to provide structure, the tree diagrams



	What	What Could	How To	Judgment	Why Did	Why Didn't	Inputs	Model	Outputs	sum
"What do you think should happen in the next 3 rounds?" (Before watching them)	2	71	16	1	0	0	24	6	2	122
"Could you briefly explain about what actually happened in these past three rounds?" (After watching them)	13	6	2	6	18	2	53	12	74	186
"Why do you think the the rounds happened the way they did?"	2	6	3	1	32	2	24	31	30	131
"Why do you think the Friendly AI did what it did?" (After seeing the explanation)	2	8	8	0	55	1	60	27	36	197
"What changes would you make in the decisions made by the Friendly AI to improve it?"	3	8	56	2	2	0	38	3	2	114
Sum	22	99	85	10	107	5	199	79	144	750

Table 2.5: Lim Dey coding of participant responses, sliced by question asked during the AAR/AI.

to provide content, and the tactic of keeping the user active along the way to encourage engagement.

2.5.3.1 Encouraging Metacognition

Researchers in the field of education have long pointed to the benefits of metacognition, in which learners evaluate the success of their own learning/understanding processes [26]. Metacognitive activity is well-established as an important influence on learning and understanding [97].

Our participants showed several instances of metacognition that seemed to come from the integration of AAR/AI, the tree explanation, and the “active user”. For example:

	"The degree to which..." [85]	**Applicable to...**	**Evidence to date for or against**
Testability	...empirical refutation is *possible*: constructs and $< predictions >$ are understandable, internally consistent, free of ambiguity	...this *explanation* of the agent's model of the world.	*Empirical*: The agent's explanations were found to be understandable by several participants, as described in Section 2.5. The diagrams were clear and explicit in their information, from most, but not all, participants' reports.
Falsifiability /Empirical Support	...is supported by empirical studies that confirm its validity	...this *explanation* of the agent's model of the world.	*Empirical*: Our explanations explicitly represented the agent's predictions about likely future states and their values, which participants could falsify.
		...this *style* of model-based explanation.	*Empirical*: AAR/AI evaluators (one instance: our participants).
Explanatory Power	...accounts for and predicts all known observations within its scope	...this *explanation* of the agent's model of the world.	*Empirical*: One measure is whether the agent's theory and explanation correctly predicted everything, in our study, the agent did not achieve this. *Criteria-based*: Whether its constructs are sufficient to express every possible action and state, i.e. completeness. In this study, the constructs have full explanatory power—but our explanation limited the number, so the actual explanation was not complete.
		...this *style* of model-based explanation.	
		...*all* model-based explanations.	
Parsimony	...<has> a minimum of concepts and propositions	...this *explanation* of the agent's model of the world.	*Criteria-based*: This explanation had 4 constructs/concepts that do not overlap, so cannot be reduced further.
Generality	...breadth of scope... and independent of specific settings	...this *explanation* of the agent's model of the world.	*Criteria-based*: This explanation's scope is limited to explaining this particular domain.
		...this *style* of model-based explanation.	*Criteria-based*: The style of explanation is not restricted to games, and should be usable for any sequential setting of model-based AI.
		...*all* model-based explanations.	Model-based explanations are restricted to model-based agents.
Utility	...supports the relevant areas	...this *explanation* of the agent's model of the world.	*Empirical*: Most, but not all, participants reported the agent's explanations to be useful to understanding its actions.
		...this *style* of model-based explanation.	*Empirical*: AAR/AI evaluators (one instance: our participants).

Table 2.6: Applying Sjøberg et al.'s Evaluation Criteria for Theories [85] to the agent's model-based explanation

P5, PTQ 2: *It made me think of it like how the AI is thinking. Is it thinking long term? Is it thinking short term? Thinking about the two different lanes each time? what the best decision would be or what I would make as the decision, so you asking that question made me think was my own decision better.*

P8, PTQ 1: *...it was good to kind of evaluate myself where I was at when thinking about what decisions the AI was doing, so I can better evaluate the next stage.*

One form of metacognition is self-explanation, and our approach encouraged some participants to generate their own explanations:

P10: *I think the aim of the AI is to increase the number of minerals, and then go to the last one that is immortals, so that they can make a great damage to the nexus.*

Finally, while our process promoted thinking about the future, the cards also supported participants' ability to reflect on the *past*:

P19, PTQ 1: *These cards? It's good to write good points and bad points for every three rounds, so that we can go back and see what mistakes we did from the bad.*

2.5.3.2 Falsifying the Agent's Predictions

One of the strengths of the model-based explanations was that it made part of the search tree explicit and that the agent made concrete predictions about the future. However, we observed that this allowed participants to falsify [72] those predictions:

P14, DP3 Why 2: *So the friendly had ... two banelings, so one baneling and some marines. Yes, that seems right. ... it predicted that the enemy would buy two more marines, and it ended up being so. Yep, it was right ... it was predicted that they would buy a baneling, and they did ... so far, it's going as predicted.*

We explicitly crafted parts of the process to allow the human to reflect on *their* past thoughts, but this participant focused on the accuracy of the *agent's* predictions about the future. Notably, this type of assessment was made possible by the model-based agent, and our explanations revealed relevant information to be able compare different time slices.

2.6 Discussion

2.6.1 Future AAR/AI implementations

AAR/AI is highly adaptable, and this provides leeway to iteratively improve it. Two areas for improvement that we observed were that participants thought they could remember what happened in the past, and that participants found questions/artifacts repetitive and burdensome at times. For example:

P20, PTQ 1: *... I am fairly confident in my ability to remember what occurred.*

P5, PTQ 1: *Some of this stuff kind of repeats...*

An alternative might be to instead enable people to decide where to pause, in an approach similar to the empirical mechanism used by Penney et al. [71]. In that study, participants watched a replay until they came to a decision that seemed important, at which point they could pause, consider our questions, and write down their thoughts. In essence, blending this device with our inner loop would give more control to the evaluators as to how often and exactly where the evaluation questions need to be answered.

2.6.2 Prediction as Explanation

2.6.2.1 Trend 1: People used explanations as prediction tools.

Reed et al. suggested that explaining a solution to a problem helps people to solve similar problems [75]. Our strategy followed a similar approach, where participants predicted the agent's action (i.e., the problem), saw the action (i.e., the solution), and then provided an explanation to the action (i.e., explanation of the solution). Some participants even began using the explanations as the basis for their prediction:

P8, Artifact PTQ: *Understanding the diagram gave some insight into how the AI thought, which made predicting its next move easier.*

Participants engaging with the model-based explanation reported attitudes consistent with a series of studies Kelleher and Hnin observed, *"suggest that learners who attempt to understand the steps of a problem solution may have higher germane load but improved ability to apply these elements in novel situations."* [42].

2.6.2.2 Trend 2: The process of predicting the actions, and then showing the actions, was powerful.

Another trend we observed is that predicting the AI agent's decisions *prior* to observing the AI agent's actual actions turned out to be part of our *explanation strategy.* One of the pillars of learning effectively is self-explaining [15]. "Good" students learn with understanding the material and forming self-explanations on their own, while "poor" students rely heavily on examples to learn and struggle to generate explanations on their own. Positioning the prediction task before the observation task effectively caused participants to create self-explanations for the AI agent's actions. Participants used the process and the explanation, to generate their own explanation for predicting the agent's actions:

P10, Why DP6: *I think the aim of the AI is to increase the number of minerals, and then go to the last one that is immortals, so that they can make a great damage to the nexus.*

Participants who answered AAR/AI questions perform a "rationale generation" [23] task, which appears to offer some benefits as an AI evaluation strategy.

Renkl et al. found that acquisition of transferable knowledge can be supported by eliciting self-explanations [76]. Learners with low levels of prior topic knowledge profit from such an elicitation procedure. We observed this effect in our study, as participants with little experience in RTS comfortably navigated through the process of assessing the AI's actions—even forming their own explanations.

2.6.3 The agent's explanations as theory

Recall from Section 2.3.3 that the agent's explanations are its explanatory theory of the game. Since it is a theory, we draw upon criteria that can be used to evaluate theories [85]. In Table 2.6, we consider how to apply these criteria to evaluate *this* agent's model-based explanation, this *style* of model-based explanations, and in some ways, even *all* model-based explanations.

2.7 Threats to Validity

Any study has threats to validity, which can skew results towards particular conclusions [98].

One such threat was the participants' amount of domain expertise. Evaluators of an AI system need domain knowledge to evaluate the AI's performance in the domain, and some of the participants may not have had enough RTS experience. 46% of participants had at least 10 hours of RTS gaming experience. It is possible that these participants' experience levels may have impacted their ability to evaluate an AI in that domain. Also, it was not clear how to interpret large decreases in the number of clarifications a participant requested early vs. late in the process. It could have meant that the participants understood the explanations over time, or alternatively that they simply gave up. The question wording could also have influenced participants' responses. Many were written and uniformly worded in a balanced set of positive, negative, and neutral wording, but the verbal post-task interview wording was informal, so more subject to individual variation.

The reliability of qualitative coding rests upon inter-rater reliability (IRR) measures. We used Jaccard [40], and 80% is considered good agreement, but for one code set we achieved only 76%. Other hindrances to the generalizability of our findings include the small size of our study and circumscribed design.

Also, qualitative studies are intended to reveal phenomena on approaches that have not been investigated before, and are not suitable for generalization. That said, we think our study helps inform model-based explanations for domains where the branching factor is small (or can be made small via pruning, as we have done).

2.8 Conclusion

In this paper, we have presented AAR/AI (After-Action Review for AI), a new assessment method to bring accountability to both AI agents and to the humans who must assess them. To inform the design of AAR/AI, we present results from a qualitative in-lab study to learn what people need when assessing an AI agent, as well as pros/cons of both the AAR/AI process and the explanations embedded in the process. Among the phenomena we found were:

- *"Organized," "Logical," and... "Repetitive":* Some participants remarked that AAR/AI process helped them think logically and stay organized. Some appreciated its support for reflection on past thoughts. Notably, the process helped participants generate rationale for events with long time lags. However, some bemoaned the repetitiveness of the AAR/AI questions.
- *Explanation complexity:* Our search tree explanations for a model-based

agent were approximately the right complexity for some of the participants to understand. They reported being able to *"draw their own conclusions"* from them, and appeared to be using them to align the agent's prediction with the actual future. Other participants did not fully understand the diagram. This mix of attitudes toward the same explanation corroborates other research reporting that explanations are not "one size fits all people" (e.g. [4]), and suggests allowing people to access different actions and/or explanation types on demand.

- *Diversity of perspectives:* As we observed and participants reported, AAR/AI's questions encouraged participants to consider their observations from multiple, different perspectives, which research suggests may produce problem-solving benefits [28].

- *How many and which:* To answer some of the AAR/AI questions, participants needed to compare items in the explanation from a very large set of options, the sheer quantity of which made them hard to co-locate. We provided the AI's four most promising options, but some participants wanted to see options the AI considered *bad* as well. Accommodating different people's comparison needs to answer the AAR/AI questions is an unresolved issue—so methods to support scalable comparisons of items in large datasets (e.g. [66]) is an active area of Info Viz research.

- *From whence:* Some participants needed to know the *provenance* of axiomatic values (value estimations at the leaf nodes). That said, if people are to be held accountable for relying on an AI agent, then the ability to "audit" its decision making by allowing the ability to trace provenance may be a requirement.

While AAR/AI was useful in guiding participants to think logically, adding explanations assisted participants in the overall assessment process. Notably, developing useful explanations and rigorously measuring their quality remains quite difficult. We hope that, by appealing to educational frameworks (e.g. Bloom's Taxonomy), we can help people like P14 see *"the flow of logic that we **should've** had"*, a benefit we hope our process will be able to extend to others tasked with assessing AI systems that impact us daily.

2.9 Acknowledgements

This work was supported by DARPA #N66001-17-2-4030. Any opinions, findings, conclusions, or recommendations expressed are the authors' and do not necessarily reflect the views of the DARPA, Army Research Office, or US government.

Chapter 3: Finding AI's Faults with AAR/AI: An Empirical Study

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. AAR/AI, an emerging 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). Further, these results were not dependent on advantages with any particular bug or type of bug; AAR/AI participants outperformed non-AAR/AI participants on every 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.

3.1 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., [62, 30, 48, 14, 87, 45, 81, 102]).

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 medical domain, Pat might be a doctor, assessing whether to believe an AI system's diagnosis of their patient's illness. In the judicial domain, Pat might be a judge, assessing whether to follow an AI system's recommendation for how long to sentence a defendant just convicted in their court. These particular situations matter to Pat and to the person 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" [67, 25].

In assessing domains like the above, no objective "ground truth" is available—only Pat knows this patient's or defendant's current situation. Even if Pat's next decision (to follow or ignore the AI system's recommendation) helps the patient to eventually recover or the defendant to eventually reform, Pat can never really know whether their own decision was correct[1].

Another domain in which this is the case is Real-Time Strategy (RTS) games, a popular domain for AI research. In RTS games, players attempt to strategically maneuver through many possible choices to hopefully win the game. Sometimes in AI research, the RTS player is an AI agent maneuvering on behalf of a human[2], and that is the case in this paper. Using the RTS context, we present an empirical study to investigate domain-knowledgeable users' effectiveness when using AAR/AI (described in Section 3.1.3), an XAI-based process to help human users assess an AI agent [60].

3.1.1 The Domain

For our study, we used a model-based reinforcement learning (RL) agent that played an RTS game. The game was the StarCraft 2 "Tug-of-War" custom game that was used in the first AAR/AI publication [60]. 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 3.1), 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

[1] Groce et al. also pointed out [30] 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.

purchasing troop production buildings and/or purchasing Pylons to increase income. Figure 3.1 shows a screenshot of the game replay as it appeared to our study's participants.

In this game, troop types have 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.

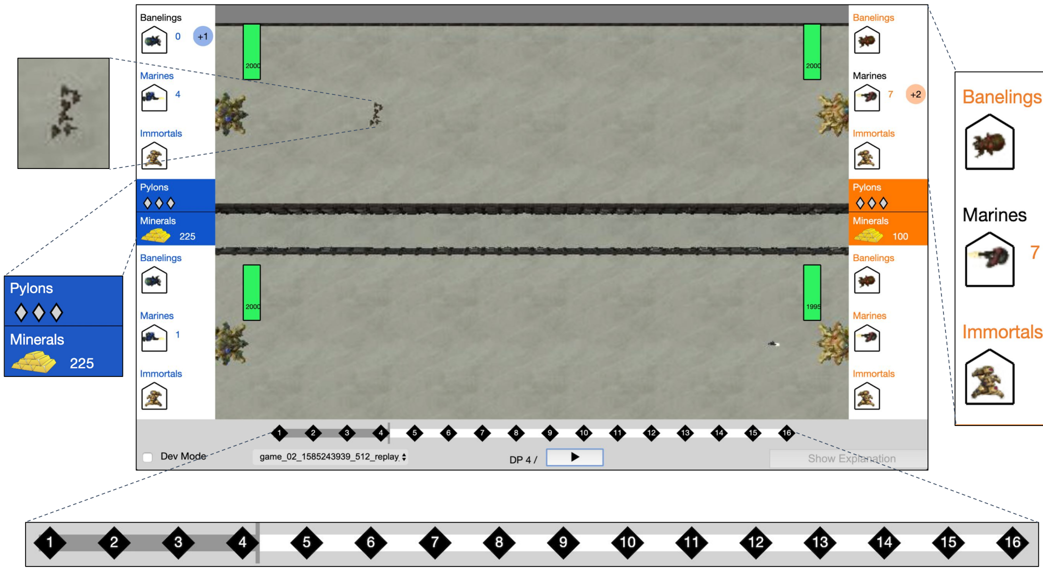


Figure 3.1: The participants' replay view of the game just past Decision Point 4 (fourth diamond; see callout at bottom). The game board has two lanes where action takes place: a top lane and a bottom lane. The (blue) friendly AI agent's "home" is the left side, and the (orange) enemy is the right side. 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).

Troops spawn behind the Nexus (the player's base, represented as gold star-shaped objects on the gameboard in Figure 3.1). Once spawned, they march down the lane and attack enemies in pre-programmed fashion, as with the marines shown in Figure 3.1. 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.

3.1.2 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

[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.

guesses", ward off ethical concerns, or defend against potential legal challenges (e.g., a malpractice suit) [47, 25].

In medicine, the causes of symptoms are diseases; in software engineering literature, the causes of symptoms (failures) are termed "faults". Avizienis et al. [7] 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".

3.1.3 How well can human users assess an AI RTS player's failures and faults with AAR/AI?

In this paper, we empirically evaluate a *process* known as After-Action Review for AI (AAR/AI) [60], by which XAI users can localize such faults (bugs) in an AI agent. We derived AAR/AI from After-Action Reviews (AAR), originally devised by the U.S. Army [92] for assessing human decisions. AAR has been used for decades to assess human decisions in the military (e.g. [31]), and has also been adapted to manned-unmanned teams [10]. Civilians have also used AAR processes

[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.

in transportation [59], medical treatment [74, 80] and emergency response [19, 39, 53]. Further, deploying AAR in a wide variety of domains has proved beneficial, with a recent meta-analysis by Keiser et al. [41] finding that, AAR yielded medium-sized practical effects, on average across 61 studies. AAR/AI is the first use of AAR in AI.

The original AAR/AI publication [60] describes AAR/AI in 7 steps. 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. (7) Finally, 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 [60], 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 3.3.2), 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?

3.2 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, 48, 51]—representations people construct in their heads about how something works from whatever they have experienced with it [64]. 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 [91]. 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" [101].

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 [76]. A user might generate self-explanations as a result of being prompted to do so, or might do so on their own accord [34].

As an example of XAI work involving self-explanations, Chi et al. showed that learners relying more heavily on examples had worse outcomes [15], which they credited to inability to engage in self-explanation. Another example occurred in our early qualitative AAR/AI results [60], in which participants' uses of self-explanation, in combination with other factors, produced high levels in Bloom's learning taxonomy [9]. 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 [11].

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.") [88]. 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. [35] 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 [95]. 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. [49, 48], 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 [16]. 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) [56, 57]. 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 [13, 70]. Research has reported that supporting the "right" intelligibility types for a particular situation or domain improved users' confidence in the system [18].

Although most current XAI research focuses on helping people interpret models' inner workings (e.g. [36, 96]), 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 [100], clustering around user-selected example-based "anchors" [14], or "covering" different input/output combinations [30]. Others support fault localization ("visual debugging" [87]) 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 [68], 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 [61], Dodge and Penney investigated how expert broadcasters explain RTS [21, 70], Kim et al. investigated human responses to Human vs. AI battles [45, 44], and Penney et al. investigated pairs of AI players making sense of “simulated AI” behavior [71, 70]. Although some of these RTS investigations included participants *noticing AI failures* (symptoms of faults), none except the AAR/AI work [60] 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.

3.3 Methodology

To investigate the effectiveness of the AAR/AI process for localizing AI’s faults/bugs, we conducted an empirical study with domain-knowledgeable users 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.

3.3.1 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 3.1). 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 3.1.1. Participants' task was to localize the AI agent's bugs. Participants in both the treatments saw the same explanation (which we describe in Section 3.3.2)—the only difference between the treatments 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 3.2); a

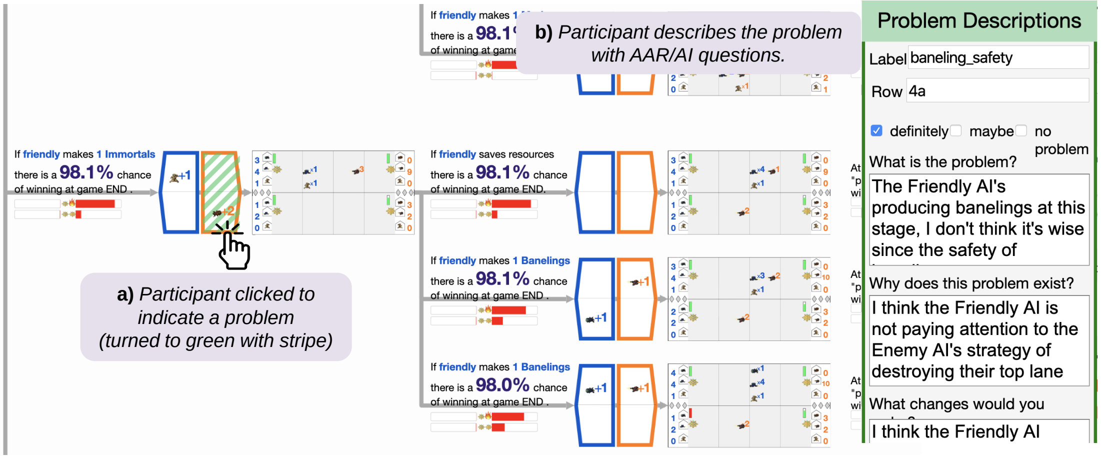
	AAR/AI	**Non-AAR/AI**	Total
Man	25	24	49
Woman	7	8	15
Transgender	1	0	1
Undergrad	13	14	27
Grad	10	7	17
Non-student	10	11	21
Total	33	32	65

Table 3.1: 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.

click log of their interactions; and their responses to a post-task NASA/TLX questionnaire [32]. All questionnaires are included in the Supplemental Documents accompanying this paper.

The study proceeded as follows. The participants agreed to an informed consent form, then filled out the pre-task demographic questionnaire, then performed Steps 1-3 below (also illustrated in Figure 3.3), 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


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.

b) *Participant describes the problem with AAR/AI questions.*

a) *Participant clicked to indicate a problem (turned to green with stripe)*

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

Figure 3.2: 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.

as defined in Section 3.1.)

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 3.4A), and they began watching a sped-up replay of a 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

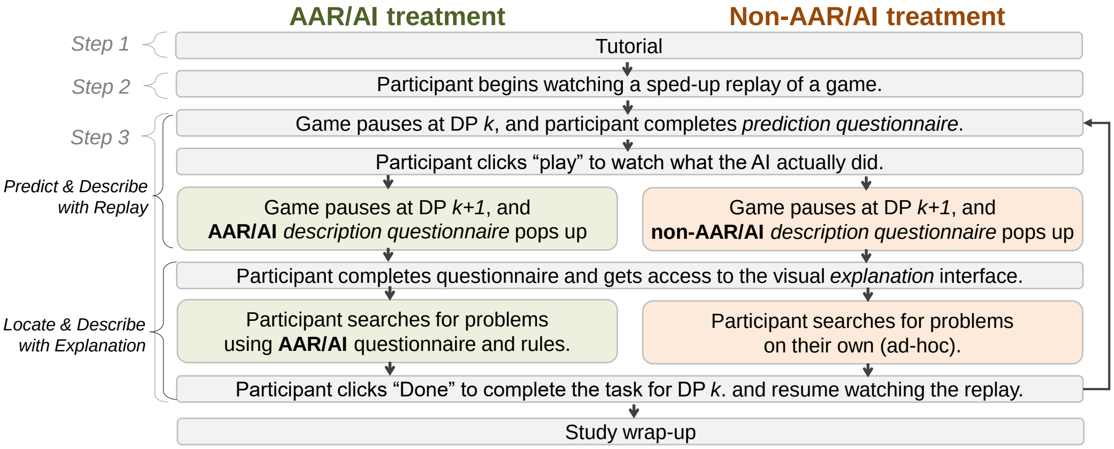

AAR/AI treatment
Non-AAR/AI treatment
Step 1
Tutorial
Step 2
Participant begins watching a sped-up replay of a game.
Step 3
Game pauses at DP k, and participant completes prediction questionnaire.
Participant clicks “play” to watch what the AI actually did.
Predict & Describe with Replay
Game pauses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

Figure 3.3: Summary of study procedure.

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 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.

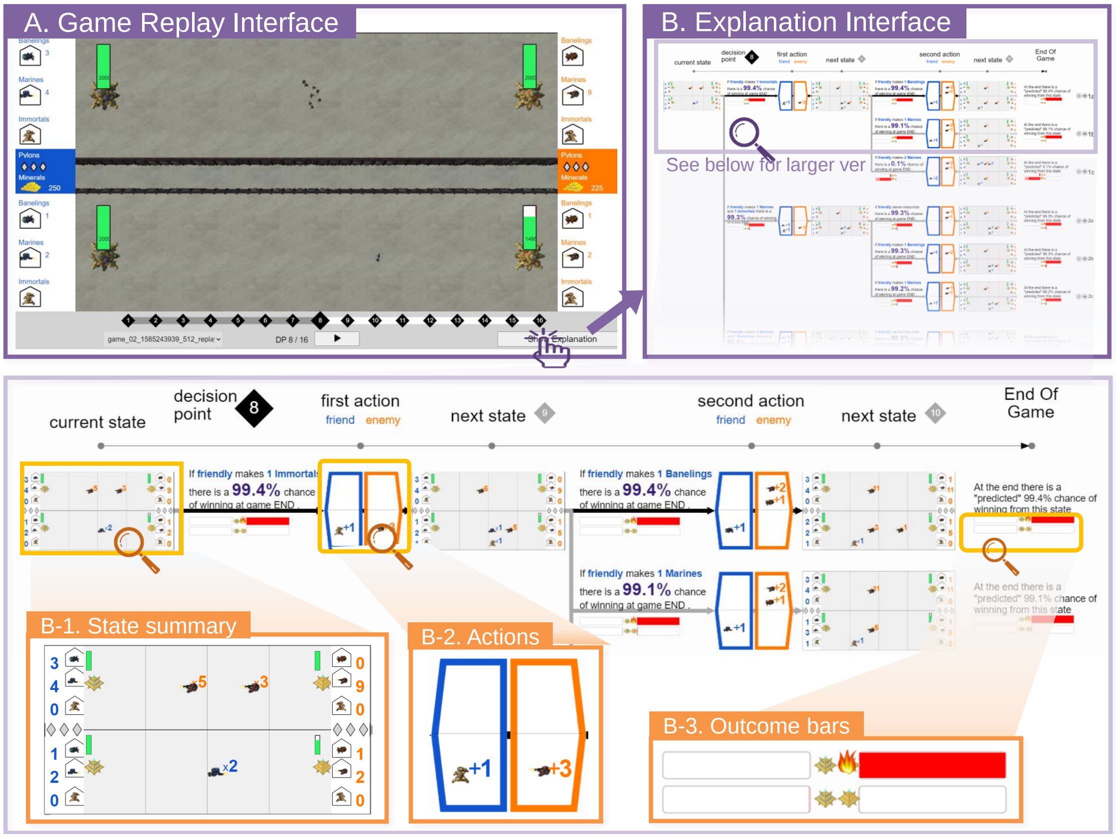
Locate and Describe with Explanation: After they had answered the initial questions, participants were able to see the explanations (Figure 3.4B). Participants were then told to locate problems in the AI's reasoning using the explanations.

In our formative investigations and pilot participants, AAR/AI seemed to benefit from "consistent search" practices [8], so we enforced a form of it in the AAR/AI treatment as follows. 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 3.2. 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.

3.3.2 Explanations

Figure 3.4B 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 3.4B-1) graphically represents a current state of the game (i.e., root of the search tree). Note that the state in Figure 3.4B-1 is an approximate thumbnail of the gameboard (Figure 3.4A). The tree expansion to the right of the current state shows different combinations of actions and states the agent predicts could happen next. Figure 3.4B-2 shows the Friendly AI's action in the blue box and the Enemy's action in orange. Next is the predicted "next state"



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

Figure 3.4: **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.

(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 3.4B).

Outcome predictions, shown in Figure 3.4B-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.

3.3.3 The Reinforcement Learning Agent

In our study, the Friendly AI "player" is powered by a model-based reinforcement learning agent[5], which determines the Friendly AI's next action by predicting future states using the following neural-network driven functions:

1. Action-Ranking Function (ARF) with type signature `(State, Action)` → `float`: answers "How good is taking this Action in this State?"

2. Transition Function (TF) with type signature `(State, Action1, Action2)` → `State`: answers "What State will arise if I (Friendly AI) take Action1 and

[5] This agent was also used in Mai et al [60].

the opponent (Enemy AI) takes Action2?"

3. Leaf Evaluation Function (LEF) with type signature `(State)` → `(outcome-probabilities)`: answers "How good is this state?"

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 3.4B). 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 3.4B). 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. While we can 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.[6]

3.3.4 The Bugs

The task for the participants was to identify the AI agent's bugs. We used naturally occurring bugs produced from the agent and additionally created new bugs based

[6] See Chapter 5 in [78] for more on minimax and game tree search.

on them. For instance, 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. 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, to harvest a set of bugs researchers agreed could be used as ground truth, and selected DPs that contained those bugs.

After rigorous analysis of the naturally occurring bugs [52] as well as trying them in our preliminary pilots, we harvested 10 of these bug instances, some of which we then exaggerated as shown in Table B.1. Five bug instances were in each of two decision points (DP). The bugs were of two types, based on which components of the reinforcement learning agent they were in: half were Transition Function (TF) bugs and half were Leaf Evaluation Function (LEF) bugs.

An example of a Leaf Evaluation Function (LEF) bug is Bug ID #1 at DP 8; this bug is shown in detail later in Figure 3.7. In row 1C, the agent predicts that the Friendly AI will *lose* with its *bottom* nexus being destroyed, while it consistently predicts that the Friendly AI will *win* by destroying the enemy AI's *top* lane nexus in other rows. It is suspicious that 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 3.8. 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.

3.4 Results

3.4.1 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 3.3.4. 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 [40]). 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 to 1 point, and if they correctly *described* the bug, they could earn another point. Table 3.2 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 3.5 shows the distributions of how many problems participants reported. AAR/AI participants reported significantly more problems than Non-AAR/AI participants (t-test, $t(63) = 5.7829$, $p < .0001$)[7]. In fact, even the AAR/AI participant with the fewest problem reports (9) still submitted more than

[7] 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.

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 B.1, 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[8]. 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 B.1 (perfect recall) and nothing else (perfect precision).

Using these measures, the AAR/AI participants had both significantly greater

[8] See Eqs. (A.5) and (A.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. [**?**] offer Eqs. (A.3) and (A.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 3.2: 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.)

average recall (Welch's t-test, t(55.666) = 4.5479, p < .0001) and precision (t-test, t(63) = 2.0358, p = .04598) than the Non-AAR/AI participants (Figure 3.6). 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[9].

Together, these three results suggest that the AAR/AI process not only encouraged participants to report significantly more problems (Figure 3.5), it also encouraged them to report problems that were indeed bugs, as measured by their significantly higher recall and precision (Figure 3.6). These results are especially encouraging given that none of the participants had backgrounds in AI/ML.

[9] 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 [**?**].

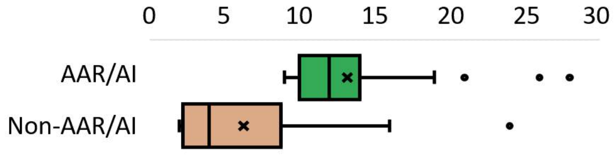


Figure 3.5: Problem report count per participant. The AAR/AI participants submitted significantly more problem reports than their Non-AAR/AI counterparts.

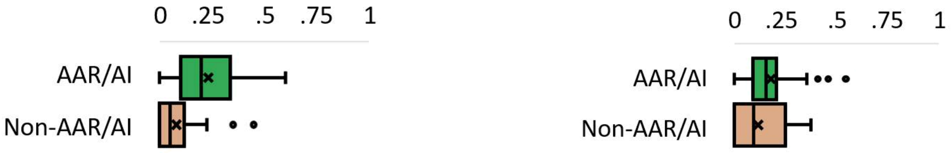


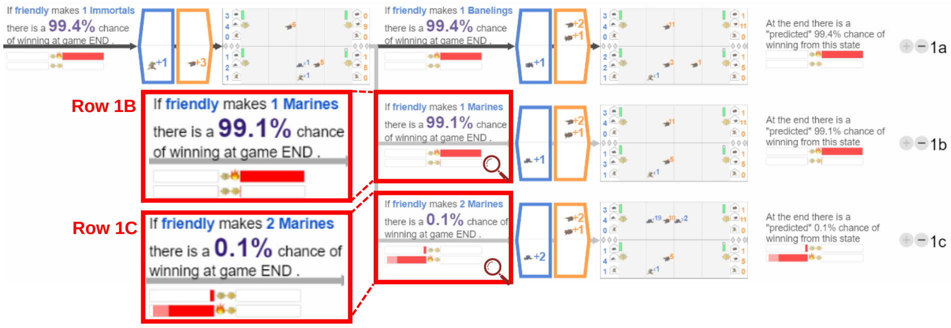
Figure 3.6: Participants' recall (left) and precision (right) (all bugs). AAR/AI participants performed significantly better than than Non-AAR/AI participants with both measures.

3.4.2 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 3.7. Transition Function (TF) bugs occur when the neural network predicts an inaccurate future state, given a current state and actions, such as in Figure 3.8. 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.

To answer this question, we analyzed recall and precision separately for LEF



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 .

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 .

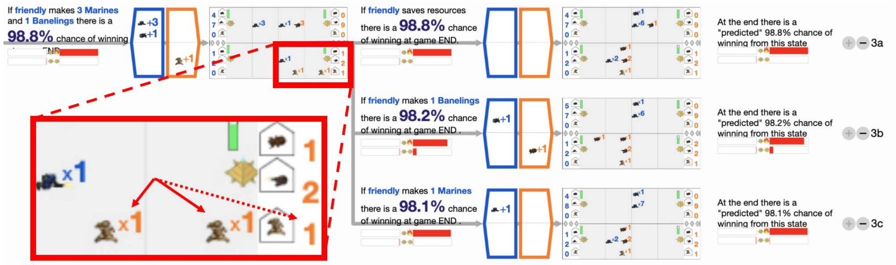
If friendly makes 2 Marines there is a 0.1% chance of winning at game END .

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

Figure 3.7: 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).

bugs and TF bugs. (Note that the number of target bugs is now split into two for analysis, which affects the distributions.) Figure 3.9 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 3.10 shows, AAR/AI participants outperformed non-AAR/AI participants on *every* bug.



If friendly makes 3 Marines and 1 Banelings there is a 98.8% chance of winning at game END.

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

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

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

Figure 3.8: 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).

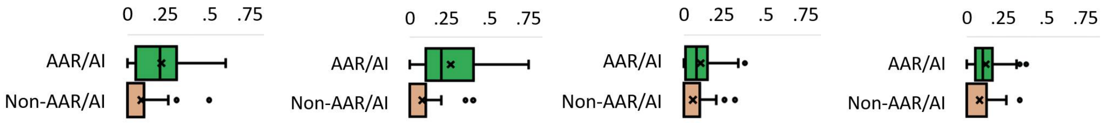
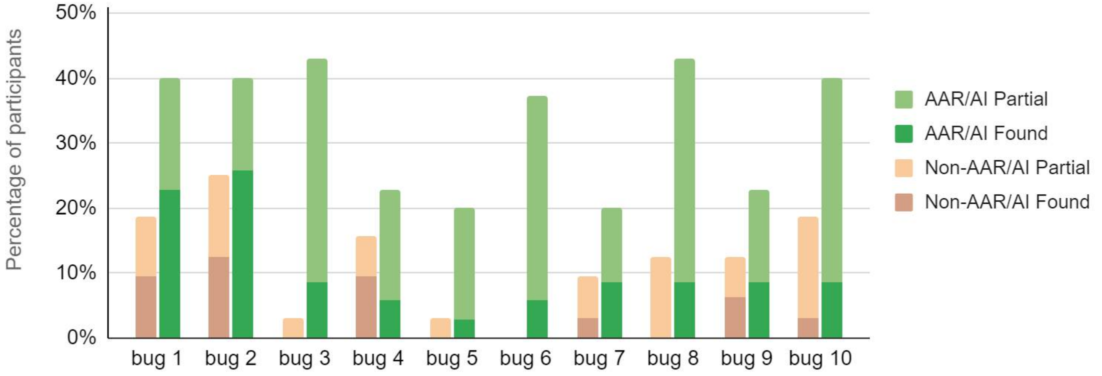


Figure 3.9: AAR/AI participants' vs. non-AAR/AI participants' (left to right): recall for leaf-evaluation function (LEF) bugs only, recall for transition function bugs (TF) only, precision for leaf-evaluation function (LEF) bugs only and precision for transition function (TF) bugs only.



Type	Leaf	Transition	Transition	Transition	Leaf	Transition	Leaf	Leaf	Leaf	Transition
DP	8	8	8	8	8	15	15	15	15	15

Figure 3.10: Percentage of participants who found each bug. For all 10 bugs, the AAR/AI participants outperformed their Non-AAR/AI counterparts.

3.4.3 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.

3.4.3.1 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 [37], where keywords are identified before and during data analysis to form the code set. This generated the categories of labels shown in Table 3.3. When a participant’s label was applicable to more than one category, we coded it in 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 [40]), coding the labels directly or in the context of the participants’ reports when necessary to disambiguate labels. Given this high level of agreement, one researcher completed the rest of the coding.

As Figure 3.11 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

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

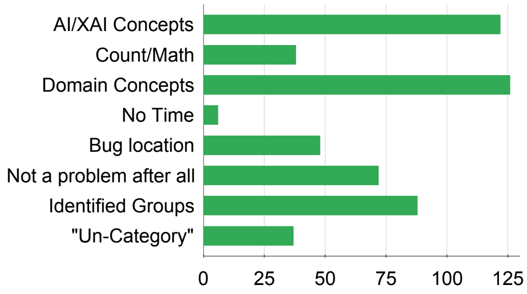


Figure 3.11: Frequencies of AAR/AI participants' ways of labeling faults into these categories. (Non-AAR/AI participants did not have a labeling feature.) Domain Concepts and AI/XAI Concepts were the most common.

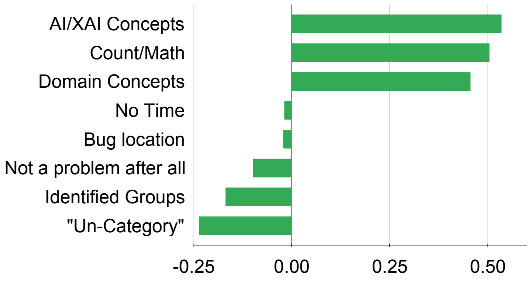


Figure 3.12: Correlation between the scores in each categorization scheme and the participant's total score.

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.

3.4.3.2 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 3.12). Each of these had a correlation coefficient r between $[0.45, 0.54]$, which is generally considered to be a moderate correlation [27]. 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, 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 3.13 illustrates one such walkthrough. Another example excerpted from a participant's report is:

120 (Labeled "Action-prediction incompatibility"), report 163, "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 la-

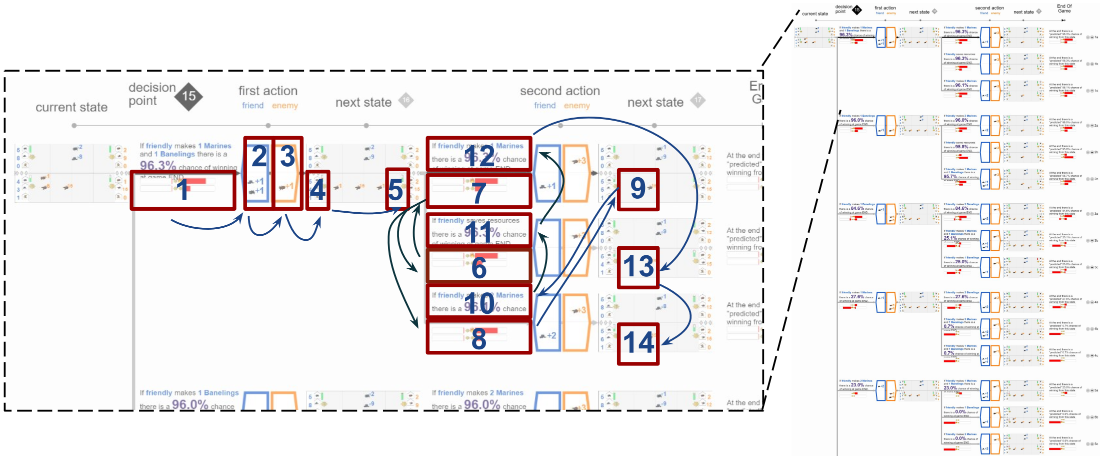


Figure 3.13: P119's search path leading to a problem report they labeled "bad decisions". To report this problem, P119 walked down to the second level, then perused each node at the second level, drilling down further if the node seemed potentially problematic (e.g., node 8), then returned to their progression through the second level until the need arose to drill down again (e.g., node 12).

bels seemed to suggest that participants were localizing bugs by "auditing" the counting/math via the explanation tree:

128 (Labeled "number"), report 206: *the numbers are inconsistent. the total marines are 9 but it only shows 6 in the game area.*

104 (Labelled "troop_calculation"), report 25: *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:

111 (Labeled "evaluated battlefield error"), report 65, Label: 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.

137 (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.

3.5 Discussion

3.5.1 Cognitive Costs Imposed by AAR/AI

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".

For each of the 5 NASA TLX dimensions gathered[10], there was insufficient evidence to suggest differences in the averages between the AAR/AI and Non-AAR/AI participants (all p-values fell between [.133, .878]). Thus, the question remains open as to whether adding AAR/AI to XAI adds cognitive load to domain-knowledgeable users' efforts to assess their AI agent's strengths and weaknesses.

[10] 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.

3.5.2 Limitations of the Study

Every empirical study has limitations and threats to validity [99, 46]. 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 study in the RTS domain, Penney et al. [71, 70] 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[11]. 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

[11] 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.

off to achieve the controls necessary to isolate an independent variable [46] (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 [46]. 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 [77], and no effects can be revealed by statistics. To address this issue, we exaggerated the size[12] of some of these naturally occurring bugs, as enumerated in Table B.1. 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

[12] 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 B.1). Participants' results did not reveal any patterns as to whether the size of the exaggeration mattered.

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.

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 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 3.2), 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.

Threats 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.

3.6 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.

- 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.

Acknowledgements

We thank Zhengxian Lin for his work on the agent powering the game and Thomas G. 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.

Chapter 4: Conclusion

In this thesis, I and my colleagues have presented a process called AAR/AI to make AI systems accessible and comprehensible to non-experts. AAR/AI aims to provide a workflow to navigate and assess such systems with ease and accuracy. To this end, we designed and conducted two human-subjects experiments. Each experiment was an inquiry into the effectiveness of the AAR/AI process in assessing and identifying and localizing faults in AI systems. We found that AAR/AI, coupled with explanation interfaces, is an efficient and noteworthy assessment process. Below, I summarize the four strengths of AAR/AI, and what makes this a great and useful process for XAI:

- *Helps in rationale and mental model building*: AAR/AI helped users form rationale for long time lags. Participants mentioned finding AAR/AI helpful in keeping their thoughts organized and logical. This effect is also observed in the accuracy and structure of bug reports submitted by AAR/AI users, where they followed an information processing pattern and developed abstractions of concepts denoting each bug on their own. AAR/AI also helped participants develop diverse perspectives to problem solve.
- *Navigating complex decision spaces*: AAR/AI helped participants navigate a real time strategy game such as StarCraft II as well as the AI's tree based

explanation in a complex decision space with ease and accuracy. The results from the quantitative study recommend that participants that used AAR/AI not only identified more faults, but they were also more precise in their assessment.

- *Explanation agnostic*: Participants fared well in the assessment process using AAR/AI, irrespective of the explanation interface. I found that participants found success with AAR/AI in all the studies conducted thus far: the qualitative study used a simplistic tree based explanation, a few preliminary studies employed model based and model free explanations with varying detail, and the quantitative study contained a complex tree based explanation that encompassed a wide variety of details not seen in the previous experiments.
- *Identifying and localizing faults accurately*: Participants particularly found AAR/AI in identifying and localizing faults. Results from the quantitative study recommend that participants that used AAR/AI had higher recall and precision of finding bugs. Moreover, labelling bugs may have further helped AAR/AI participants in finding bugs by abstracting higher level concepts into unique individual theories.

4.1 Future work

As a part of potential future work, an alternate human centered process may be developed, and consequently compared with AAR/AI. It would be interesting to evaluate the specific advantages and disadvantages of diverse AI intensive processes that align with different types of human information processing capabilities. Such questions encourage understanding and evaluating combinations of workflows and explanation interfaces that are suited to a certain intelligibility types or demographics. More research is required to find the types of assessment tasks that can be carried out to effectively evaluate such systems, and if success in a specific assessment task is equivalent of a well formed mental model of the AI system.

Bibliography

[1] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. 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, 2018.

[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. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19. ACM, 2019.

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

[4] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Burnett. Explaining reinforcement learning to mere mortals: An empirical study. In *International Joint Conference on Artificial Intelligence*, Macau, China, 10–18 August 2019. IJCAI.

[5] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Matthew Olson, Alan Fern, and Margaret Burnett. Mental models of mere mortals with explanations of reinforcement learning. *ACM Transactions on Interactive Intelligent Systems*, 10(2), May 2020.

[6] Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Pintrich, James Raths, and Merlin C. Wittrock. *A Taxonomy for Learning, Teaching, and Assessing: A revision of Bloom's Taxonomy of Educational Objectives.* Pearson, New York, NY, USA, 2001.

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

[8] Adam T. Biggs and Stephen R. Mitroff. 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):142–148, 2015.

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

[10] Ralph Brewer, Anthony Walker, E. Ray Pursel, Eduardo Cerame, Anthony Baker, and Kristin Schaefer. 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, pages 119–130, Cham, CHE, 2019. Springer Nature Switzerland AG.

[11] Ruth M. J. Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, IJCAI'19, pages 6276–6282. International Joint Conferences on Artificial Intelligence Organization, 2019.

[12] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

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

[14] Nan-Chen Chen, Jina Suh, Johan Verwey, Gonzalo Ramos, Steven Drucker, and Patrice Simard. Anchorviz: Facilitating classifier error discovery through interactive semantic data exploration. In *Proceedings of the 23th International Conference on Intelligent User Interfaces*, IUI '18, pages 269–280. ACM, 2018.

[15] Michelene T.H. Chi, Miriam Bassok, Matthew W. Lewis, Peter Reimann, and Robert Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13(2):145–182, 4 1989.

[16] Michael Chromik, Malin Eiband, Sarah Theres Völkel, and Daniel Buschek. Dark patterns of explainability, transparency, and user control for intelligent systems. In *IUI Workshops*, 2019.

[17] CNN. Who's responsible when an autonomous car crashes?, 2016.

[18] Kelley Cotter, Janghee Cho, and Emilee Rader. 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*, pages 1553–1560. ACM, 2017.

[19] Robert Davies, Elly Vaughan, Graham Fraser, Robert Cook, Massimo Ciotti, and Jonathan E. Suk. 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):618–625, june 2019.

[20] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13:319–340, 1989.

[21] Jonathan Dodge, Sean Penney, Claudia Hilderbrand, Andrew Anderson, and Margaret Burnett. 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*, CHI '18, pages 562:1–562:12, New York, NY, USA, 2018. ACM.

[22] Jonathan Dodge 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. After-action review for ai (AAR/AI). *ACM Transactions on Interactive Intelligent Systems (to appear)*, 2021.

[23] Upol Ehsan, Pradyumna Tambwekar, Larry Chan, Brent Harrison, and Mark O. Riedl. Automated rationale generation: A technique for explainable ai and its effects on human perceptions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, pages 263–274, New York, NY, USA, 2019. ACM.

[24] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Xiaodong Song.

Robust physical-world attacks on deep learning visual classification. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[25] Luciano Floridi, Josh Cowls, Monica Beltrametti, Raja Chatila, Patrice Chazerand, Virginia Dignum, Christoph Luetge, Robert Madelin, Ugo Pagallo, Francesca Rossi, et al. Ai4people—an ethical framework for a good ai society: opportunities, risks, principles, and recommendations. *Minds and Machines*, 28(4):689–707, 2018.

[26] Donna-Lynn Forrest-Pressley and GE MacKinnon. *Metacognition, Cognition, and Human Performance: Theoretical Perspectives*, volume 1. Academic Pr, 1985.

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

[28] Hershey H Friedman, Linda W Friedman, and Chaya Leverton. Increase diversity to boost creativity and enhance problem solving. *Psychosociological Issues in Human Resource Management*, 4(2):7, 2016.

[29] Ian Goodfellow and Nicolas Papernot. The challenge of verification and testing of machine learning, 2017.

[30] A. Groce, T. Kulesza, C. Zhang, S. Shamasunder, M. Burnett, W. Wong, S. Stumpf, S. Das, A. Shinsel, F. Bice, and K. McIntosh. You are the only possible oracle: Effective test selection for end users of interactive machine learning systems. *IEEE Transactions on Software Engineering*, 40(03):307–323, mar 2014.

[31] Samer Hanoun and Saeid Nahavandi. Current and future methodologies of after action review in simulation-based training. In *2018 Annual IEEE International Systems Conference (SysCon)*, SysCon '18, pages 1–6, New York, NY, USA, 2018. IEEE.

[32] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Peter A. Hancock and Najmedin Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139 – 183. North-Holland, 1988.

[33] Marcel Heerink, Ben Kröse, Vanessa Evers, and Bob Wielinga. Assessing acceptance of assistive social agent technology by older adults: the almere model. *International Journal of Social Robotics*, 2(4):361–375, Dec 2010.

[34] Robert Hoffman, Gary Klein, and Shane Mueller. Explaining explanation for "explainable ai". *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 62:197–201, 09 2018.

[35] Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. Metrics for explainable AI: challenges and prospects. *CoRR*, abs/1812.04608, 2018.

[36] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2019.

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

[38] Sandy H. Huang, Kush Bhatia, Pieter Abbeel, and Anca D. Dragan. Establishing appropriate trust via critical states. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3929–3936, 2018.

[39] Andrew Ishak and Elizabeth Williams. Slides in the tray: How fire crews enable members to borrow experiences. *Small Group Research*, 48(3):336–364, March 2017.

[40] Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, 44:223–270, 1908.

[41] Nathanael Keiser and Winfred Arthur, Jr. 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.

[42] Caitlin Kelleher and Wint Hnin. Predicting cognitive load in future code puzzles. In *2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 257:1–257:12, New York, NY, USA, 2019. ACM.

[43] Roli Khanna, Jonathan Dodge, Andrew Anderson, Rupika Dikkala, Jed Irvine, Zeyad Shureih, Kin-Ho Lam, Caleb R. Matthews, Minsuk Kahng, Alan Fern, and Margaret Burnett. Finding AI's faults with AAR/AI: An empirical study. *(Under Review)*, 2021.

[44] Man-Je Kim, Kyung-Joong Kim, SeungJun Kim, and Anind Dey. Evaluation of starcraft artificial intelligence competition bots by experienced human players. In *2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1915–1921, New York, NY, USA, 2016. ACM.

[45] Man-Je Kim, Kyung-Joong Kim, SeungJun Kim, and Anind K Dey. Evaluation of starcraft artificial intelligence competition bots by experienced human players. In *ACM CHI Conference Extended Abstracts*, pages 1915–1921. ACM, 2016.

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

[47] Cliff Kuang. Can ai be taught to explain itself? 2017. Retrieved December 26, 2017 from `https://www.nytimes.com/2017/11/21/magazine/can-ai-be-taught-to-explain-itself.html`.

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

[49] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W. K. Wong. 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)*, pages 3–10, Sept 2013.

[50] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. Tell me more? the effects of mental model soundness on personalizing an intelligent agent. In *ACM Conference on Human Factors in Computing Systems*, pages 1–10. ACM, 2012.

[51] Todd Kulesza, Simone Stumpf, Margaret Burnett, Weng-Keen Wong, Yann Riche, Travis Moore, Ian Oberst, Amber Shinsel, and Kevin McIntosh. Explanatory debugging: Supporting end-user debugging of machine-learned

programs. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 41–48. IEEE, 2010.

[52] Kin-Ho Lam, Zhengxian Lin, Jed Irvine, Jonathan Dodge, Zeyad T Shureih, Roli Khanna, Minsuk Kahng, and Alan Fern. Identifying reasoning flaws in planning-based rl using tree explanations. In *IJCAI-PRICAI 2020 Workshop on XAI*, 2020.

[53] Adam Lareau and Brice Long. The art of the after-action review. *Fire Engineering*, 171(5):61–64, May 2018.

[54] Brian Lim, Anind Dey, and Daniel Avrahami. *Why* and *Why Not* explanations improve the intelligibility of context-aware intelligent systems. In *2009 SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2119–2128, New York, NY, USA, 2009. ACM.

[55] Brian Y Lim. *Improving understanding and trust with intelligibility in context-aware applications.* PhD thesis, figshare, 2012.

[56] Brian Y. Lim. *Improving understanding and trust with intelligibility in context-aware applications.* PhD thesis, Carnegie Mellon University, 2012.

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

[58] Sandra Deacon Lloyd Baird, Phil Holland. Learning from action: Imbedding more learning into the performance fast enough to make a difference. 27:19–32, 1999.

[59] Sandra Deacon Lloyd Baird, Phil Holland. Learning from action: Imbedding more learning into the performance fast enough to make a difference. 27:19–32, 1999.

[60] 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. Keeping it "organized and logical": After-action review for ai (AAR/AI). In *25th International Conference on Intelligent User Interfaces*, IUI '20. ACM, 2020.

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

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

[63] John E. Morrison and Larry L. Meliza. Foundations of the After Action Review Process. Technical report, Institute for Defense Analyses, 1999.

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

[65] N.Y. Times. Tesla's self-driving system cleared in deadly crash, 2017.

[66] Oluwakemi Ola and Kamran Sedig. Beyond simple charts: Design of visualizations for big health data. *Online journal of public health informatics*, 8, 12 2016.

[67] European Group on Ethics in Science and New Technologies. Statement on artificial intelligence, robotics and 'autonomous' systems. 2018.

[68] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(4):293–311, Dec 2013.

[69] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore. *Proceedings of the 26th Symposium on Operating Systems Principles - SOSP '17*, 2017.

[70] Sean Penney, Jonathan Dodge, Andrew Anderson, Claudia Hilderbrand, Logan Simpson, and Margaret Burnett. The shoutcasters, the game enthusiasts, and the ai: Foraging for explanations of real-time strategy players. 0(ja). (To Appear).

[71] Sean Penney, Jonathan Dodge, Claudia Hilderbrand, Andrew Anderson, Logan Simpson, and Margaret Burnett. Toward foraging for understanding of starcraft agents: An empirical study. In *23rd International Conference on Intelligent User Interfaces*, IUI '18, pages 225–237, New York, NY, USA, 2018. ACM.

[72] Karl R Popper. Science as falsification. *Conjectures and refutations*, 1:33–39, 1963.

[73] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *Proceedings of the 22Nd International Conference on Computer Aided Verification*, CAV'10, pages 243–257, Berlin, Heidelberg, 2010. Springer-Verlag.

[74] John Quarles, Samsun Lampotang, Ira Fischler, Paul Fishwick, and Benjamin Lok. Experiences in mixed reality-based collocated after action review. *Virtual Reality*, 17(3):239–252, September 2013.

[75] Stephen Reed, Alexandra Dempster, and Michael Ettinger. Usefulness of analogous solutions for solving algebra word problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(1):106–125, January 1985.

[76] Alexander Renkl, Robin Stark, Hans Gruber, and Heinz Mandl. Learning from worked-out examples: The effects of example variability and elicited self-explanations. *Contemporary Educational Psychology*, 23(1):90–108, January 1998.

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

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

[79] Margaret Salter and Gerald Klein. After action reviews: Current observations and recommendations. Technical report, U.S. Army Research Institute for the Behavioral and Social Sciences, 2007.

[80] Taylor Lee Sawyer and Shad Deering. Adaptation of the us army's after-action review for simulation debriefing in healthcare. *Simulation in Healthcare*, 8(6):388–397, December 2013.

[81] James Schaffer, John O'Donovan, James Michaelis, Adrienne Raglin, and Tobias Höllerer. I can do better than your ai: Expertise and explanations. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19, pages 240–251, New York, NY, USA, 2019. ACM.

[82] Martin Schindler and Martin J Eppler. Harvesting project knowledge: a review of project learning methods and success factors. *International Journal of Project Management*, 21(3):219 – 228, 2003.

[83] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[84] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[85] Dag IK Sjøberg, Tore Dybå, Bente CD Anda, and Jo E Hannay. Building theories in software engineering. In *Guide to advanced empirical software engineering*, pages 312–336. Springer, 2008.

[86] Dan "Artosis" Stemkoski. AlphaStar - Analysis by Artosis. `https://www.youtube.com/watch?v=_YWmU-E2WFc`, 2019.

[87] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander Rush. Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25:353–363, 2019.

[88] John Sweller, Jeroen J. G. Van Merrienboer, and Fred Paas. Cognitive architecture and instructional design. *Educational Psychology Review*, 10:251–, 09 1998.

[89] The StarCraft II Community. Tutorials - Sc2MapsterWiki. `https://sc2mapster.gamepedia.com/Tutorials`, 2019.

[90] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars, 2017.

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

[92] U.S. Army. Training circular 25-20: A leader's guide to after-action reviews. Technical report, Department of the Army, Washington D.C., USA, 1993.

[93] Oriol Vinyals. Deepmind and blizzard open starcraft ii as an ai research environment, 2017.

[94] Oriol Vinyals, David Silver, et al. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii`, 2019.

[95] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. Designing theory-driven user-centric explainable ai. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '19, 2019.

[96] J. Wang, L. Gou, H. Shen, and H. Yang. Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):288–298, 2019.

[97] Franz Emanuel Weinert and Rainer H Kluwe. Metacognition, motivation, and understanding. 1987.

[98] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

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

[100] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, ACL '19, pages 747–763, 2019.

[101] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 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, 2020.

[102] Qian Yang, Aaron Steinfeld, and John Zimmerman. 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, 2019.

APPENDICES

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:

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

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

To find a multi-class, multi-label analog, we consulted a review by Zhang et al. [**?**], 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 (A.3) and (A.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.

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

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

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[1] 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

[1] 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 A.1: 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 A.5 and A.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.

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 A.1: many participants reported the same problem multiple times. To handle this without awarding excess credit, we create Table A.2 so that it is sliced per *participant* by aggregating each participant's part of columns of Table A.1 by taking a max over the reports from each participant.

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

- B is the set of bugs (in our case, there are 10);

		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 A.2: 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.

- 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.

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

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

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 (A.7) and (A.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).

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

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

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 (A.5) and (A.6), one can compute the *recall* of a participant using Table A.2; we can sum across the row and divide by the number of bugs (the max was already incorporated moving from Table A.1 to Table A.2). 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 (A.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 (A.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 $\frac{2}{7} \approx 29\%$.

Appendix B: Study Design Details

B.0.1 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 3.1's left) and the "Enemy AI" (right). Participants were shown the game from the Friendly AI's perspective.

As Figure 3.1 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 3.1 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 3.1. 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.0.2 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.

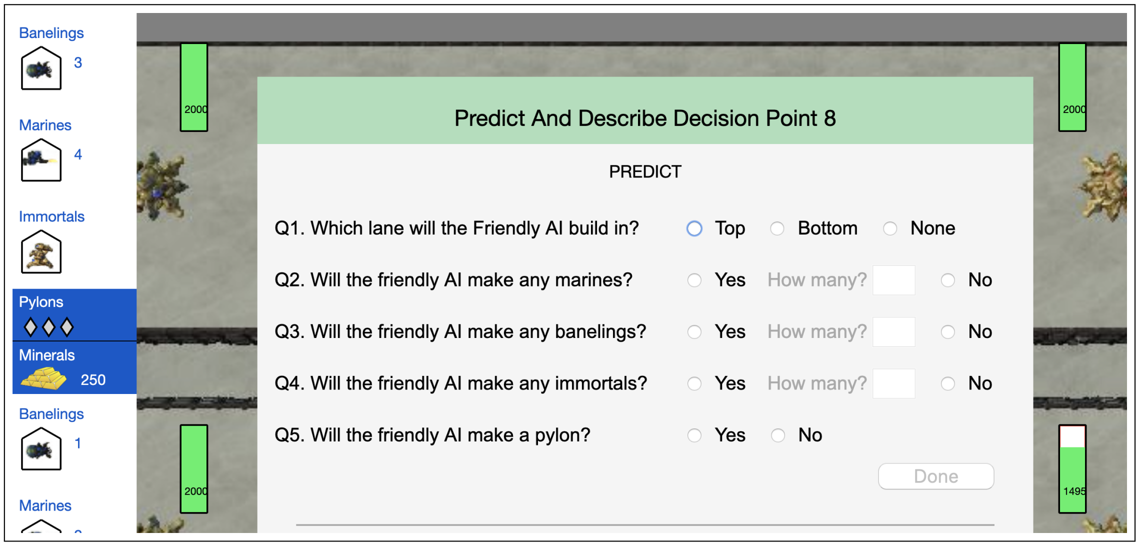
Participants in both treatments saw identical AI agents, game replays and explanations. Their agents also exhibited identical bugs, which are detailed in Table B.1.

The game automatically paused at DP 8, and participants in both treatments were shown identical “prediction” questions (Figure B.1). 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 B.2), whereas the Non-AAR/AI group was given observational questions (Figure B.3).

After answering the prediction and description questions, participants in both treatments then saw the Friendly AI’s explanation for its actions. Figure B.5 shows the AAR/AI interface, and Figure **??** 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,

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 B.1: 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 3.3.3). 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

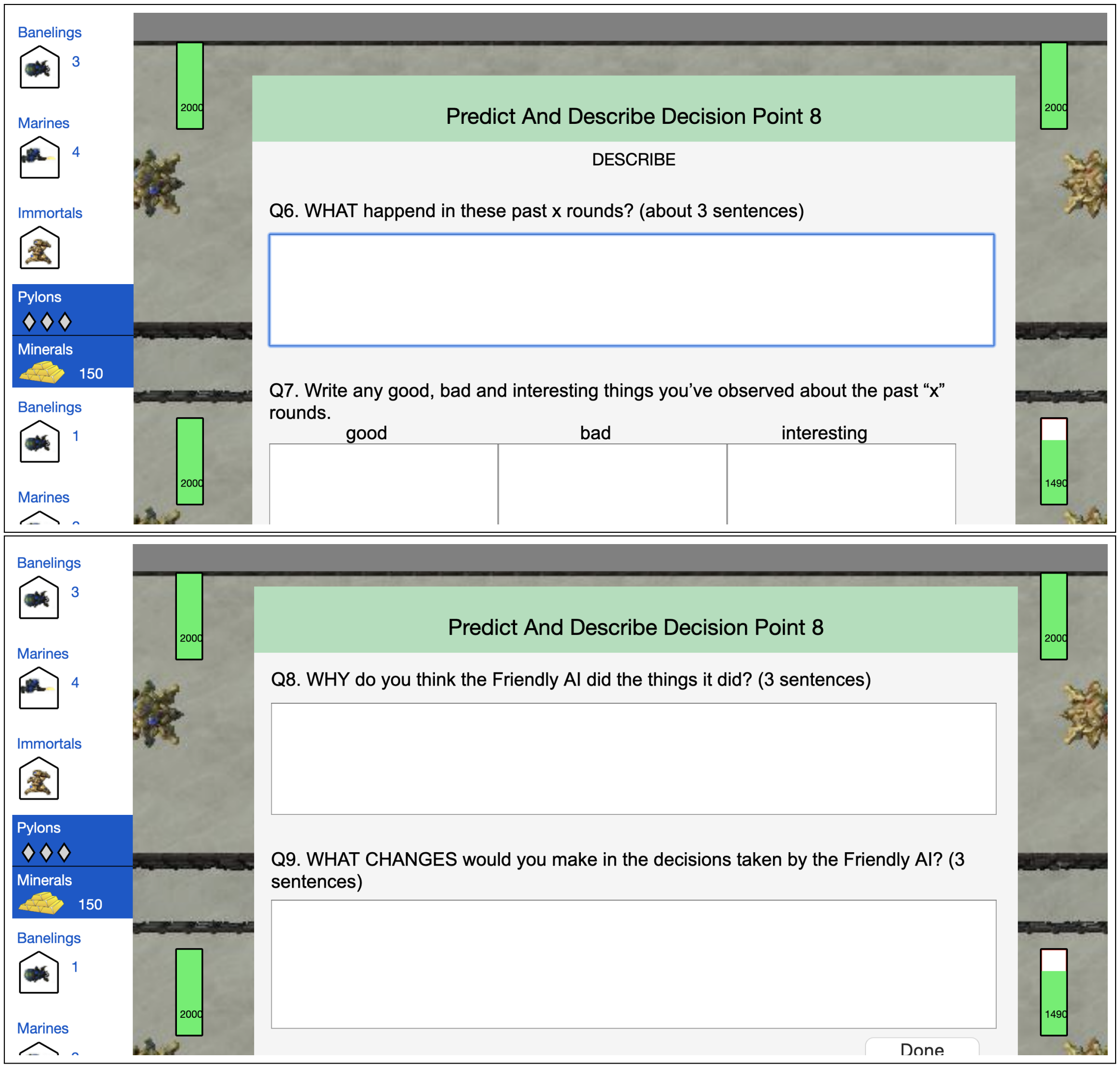
Q4. Will the friendly AI make any immortals? ○ Yes How many? ○ No

Q5. Will the friendly AI make a pylon? ○ Yes ○ No

Done

Figure B.1: 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.

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.



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.

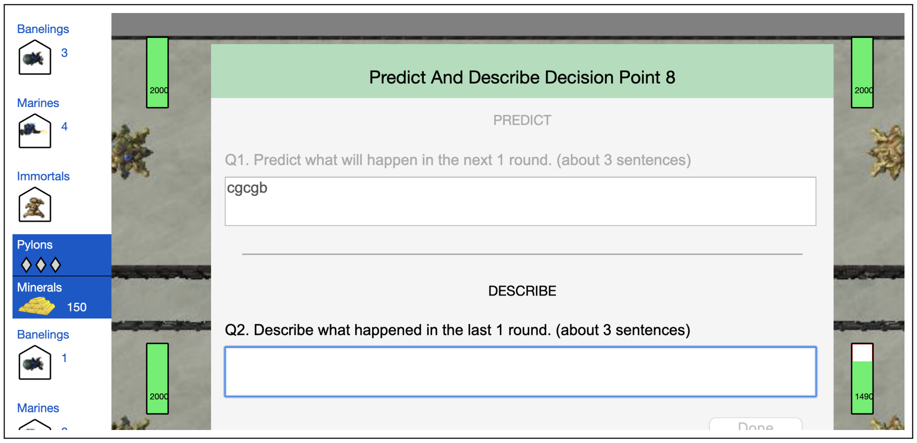
good	bad	interesting

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)

Figure B.2: Having predicted what the agent would do (Figure B.1), 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)

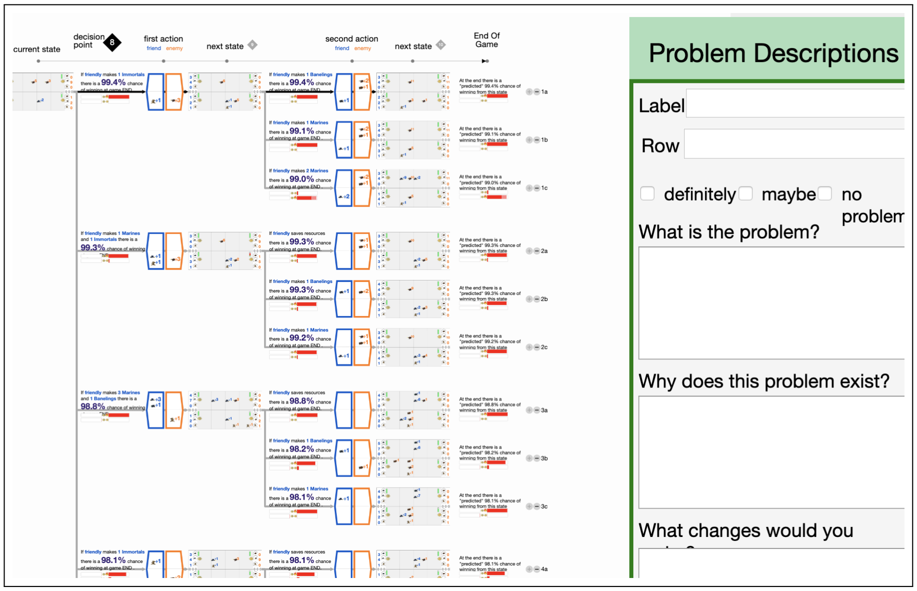
cgcgb

DESCRIBE

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

Done

Figure B.3: Having predicted what the agent would do (Figure B.1), 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

Figure B.4: 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.

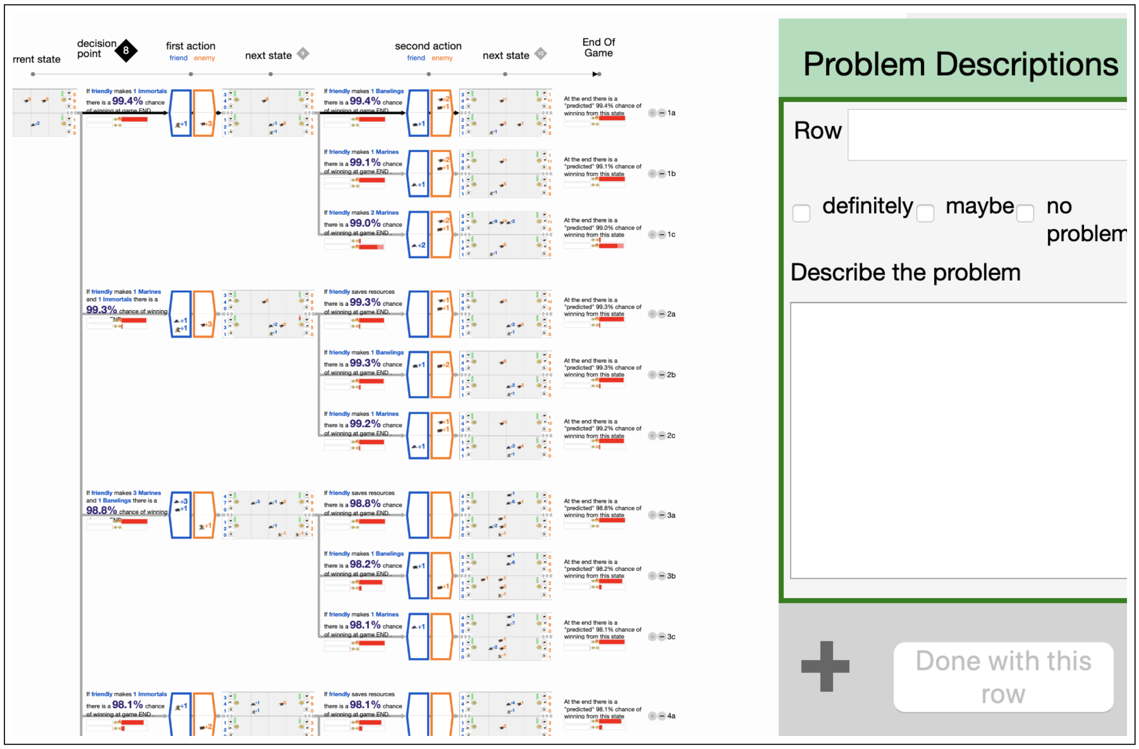


Figure B.5: Non-AAR/AI participants answered a simple "Describe the problem" question for each problem they found.

Appendix C: Model-Based Agent Architecture

This section describes details about the model-based agent we used for the studies. Figure C.1 illustrates the overall architecture of the agent. We constructed a Minimax search tree by combining a *decomposed reward* deep Q-network (drDQN [**?**]), used for *action ranking* and *leaf evaluation*, and a transition model [52]. 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

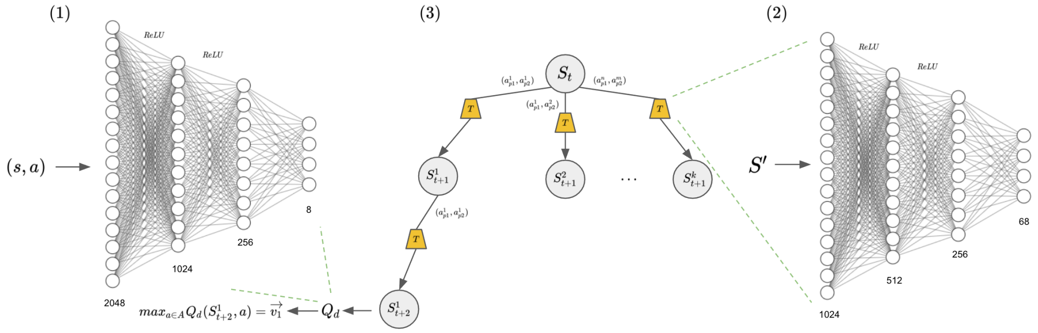


Figure C.1: The model-based agent consists of three parts. **(1, left)** The decomposed reward DQN (drDQN) model, which takes a state-action pair (s, a) as input and outputs a decomposed Q-value vector. **(2, right)** The transition model, which outputs the estimated next game state by taking the after-state 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.

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 C.1. 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.0.1 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, declining to expand actions that do not look promising (this is the *Action Ranking Function* referred to in Section 3.3.3). 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 3.3.3). 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.0.2 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.