How the Experts Do It: Assessing and Explaining Agent Behaviors in Real-Time Strategy Games

Jonathan Dodge, Sean Penney, Claudia Hilderbrand, Andrew Anderson, Margaret Burnett
Oregon State University
Corvallis, OR; USA
{ dodgej, penneys, minic, anderan2, burnett }@eecs.oregonstate.edu

ABSTRACT
How should an AI-based explanation system explain an agent's complex behavior to ordinary end users who have no background in AI? Answering this question is an active research area, for if an AI-based explanation system could effectively explain intelligent agents' behavior, it could enable the end users to understand, assess, and appropriately trust (or distrust) the agents attempting to help them. To provide insights into this question, we turned to human expert explainers in the real-time strategy domain —"shoutcasters" — to understand (1) how they foraged in an evolving strategy game in real time, (2) how they assessed the players' behaviors, and (3) how they constructed pertinent and timely explanations out of their insights and delivered them to their audience. The results provided insights into shoutcasters' foraging strategies for gleaning information necessary to assess and explain the players; a characterization of the types of implicit questions shoutcasters answered; and implications for creating explanations by using the patterns and abstraction levels these human experts revealed.

CCS Concepts
•**Human-centered computing** → **Empirical studies in HCI;**
•**Computing methodologies** → *Intelligent agents;*

Author Keywords
Explainable AI; Intelligent Agents; RTS Games; StarCraft; Information Foraging

INTRODUCTION
Real-time strategy (RTS) games are becoming popular artificial intelligence (AI) research platforms. A number of factors have contributed to this trend. First, RTS games are a challenge for AI because they involve real-time adversarial planning within sequential, dynamic, and partially observable environments [25]. Second, AI advancements made in the RTS domain can be mapped to real world combat mission planning and execution such as an AI system trained to control a fleet of drones for missions in simulated environments [36].

People without AI training will need to *understand* and ultimately *assess* the decisions of such a system, based on what such intelligent systems recommend or decide to do on their own. For example, imagine "Jake," a domain expert trying to make an educated decision about whether or not to use an intelligent agent. Ideally, an interactive explanation system could help Jake assess whether and when the AI is making its decisions "for the right reasons," so as to ward off "lucky guesses" and legal/ethical concerns (see [15]).

Scenarios like this are the motivation for a burgeoning area of research referred to as "Explainable AI," where an automated explanation device presents an AI system's decisions and actions in a form useful to the intended audience — here, Jake. There are recent research advances in explainable AI, as we discuss in the Related Work section, but only a few focus on explaining *complex strategy environments* like RTS games and fewer draw from expert explainers. To help fill this gap, we conducted an investigation in the setting of StarCraft II, a popular RTS game [25] available to AI researchers [38].

We looked to "shoutcasters," who are commentators for e-sports like RTS games. In StarCraft e-sports, two players compete while the shoutcasters provide real-time commentary. Shoutcasters are helpful to investigate for explaining AI agents in real time to people like Jake for two reasons. First, they face an *assessment* task — similar to Jake's. Specifically, they must 1) discover the actions of the player, 2) make sense of them and 3) assess them, particularly if they discover good, bad, or unorthodox behavior. They must do all this while simultaneously constructing an explanation of their discoveries.

Second, shoutcasters are *expert explainers*. As communication professionals, they are paid to inform an audience they cannot see or receive feedback/questions from. Hoffman & Klein [11] researched five stages of explanation, looking at how explanations are formed from observation of an event, generating one or more possible explanations, judging the plausibility of said explanations, and either resolving or extending the explanation. Their findings help to illustrate the complexity of shoutcasters' task, due to its abductive nature of explaining the past and anticipating the future. In short, shoutcasters must anticipate and *answer the questions the audience are not able to ask*, all while passively watching the video stream.

CHI 2018, April 21–26, 2018, Montréal, QC, Canada.

http://dx.doi.org/10.1145/3173574.3174136

Because shoutcasters explain in parallel to gathering their information, we guided part of our investigation using Information Foraging Theory (IFT) [29], which explains how people go about their information seeking activities. It is based on naturalistic predator-prey models, in which the *predator* (shoutcaster) searches *patches* (parts of the information environment) to find *prey* (evidence of players' decision process) by following the *cues* (signposts in the environment that seem to point toward prey) based on their *scent* (predator's guess at how related to the prey a cue is). IFT constructs have been used to explain and predict people's information-seeking behavior in several domains, such as understanding navigations through web sites or programming and software engineering environments [5, 8, 9, 18, 23, 26, 27, 28, 33]. However, to our knowledge, it has not been used before to investigate explaining RTS environments like StarCraft.

Using this framework, we investigated the following research questions (RQs). RQ1-RQ2 investigate the information needs of shoutcasters as human assessors, like our target user Jake. (Because shoutcasters may not have access to the players'/agents' internal reasoning, this perspective is especially pertinent to model-agnostic XAI, which ignores internals to achieve generality; e.g., [10, 14, 30]):

RQ1 *The What and the Where*: What information do shoutcasters seek to generate explanations, and where do they find it?

RQ2 *The How*: How do shoutcasters seek the information they seek?

We then used RQ3-RQ4 to investigate shoutcasters in their role as expert explainers:

RQ3 *The Questions*: What implicit questions do shoutcasters answer and how do they form their answers?

RQ4 *The Explanations*: What relationships and objects do shoutcasters use when building their explanations?

BACKGROUND AND RELATED WORK

Our work draws upon mental models in XAI. Mental models, defined as "internal representations that people build based on their experiences in the real world," enable users like "Jake" (our target user) to predict system behavior [24]. Kulesza et al. [17] found those who adjusted their mental models most in response to explanations of AI (a recommender system) were best able to customize recommendations. Further, participants who improved their mental models the most found debugging more worthwhile and engaging.

Building upon this finding, Kulesza et al. [16] then identified principles for explaining (in a "white box" fashion) to users how a machine learning system makes its predictions more transparent to the user. Participants' quality of mental models increased by up to 52% in user studies with a prototype following these principles, and along with these improvements came better ability to customize the intelligent agents. Kapoor et al. [12] also showed that explaining AI increased user satisfaction and interacting with the explanations enabled users to construct classifiers that were more aligned with target preferences. Bostandjiev et al.'s work on a music recommendation system [2] found that explanation led to a remarkable increase in user-satisfaction with their system.

Another important underpinning for our work is what people *want* explained. Lim & Dey [20] conducted an influential investigation into information demanded from context-aware intelligent systems. They categorized users' information needs into various "intelligibility types," and investigated which types provided the most benefit to user understanding. Among these types were "What" questions (What did the system do?), "Why" questions (Why did the system do X?), and so on. We draw upon these results in this paper to categorize the kinds of questions that shoutcasters' explanations answered.

Other research confirms that explanations containing certain intelligibility types make a difference in user attitude towards the system. For example, findings by Cotter et al. [6] showed that justifying *why* an algorithm works (but not on *how* it works) were helpful for increasing users' confidence in the system — but not for improving their trust. Other work shows that the relative importance of the intellibility types may vary with the domain; for example, findings by Castelli et al. [3] in the domain of smart homes showed a strong interest in "What" questions, but few of the other intellibility types.

Constructing effective explanations of AI is not straightforward, especially when the underlying AI system is complex. Both Kulesza et al. [16] and Guestrin et al. [30] point to a potential trade-off between faithfulness and interpretability in explanation. The latter group developed an algorithm that can explain (in a "black box" or "model-agnostic" fashion) predictions of any classifier in a faithful way, and also approximate it locally with an interpretable model. They described a fidelity-interpretability trade-off, in which making an explanation more faithful was likely to reduce its interpretability, and vice versa. However, humans manage this trade-off by accounting for many factors, such as the audience's current situation, their background, amount of time available, etc. One goal of the current study is to understand how expert human explainers, like our shoutcasters, manage this trade-off.

In the domain of assessing RTS intelligent agents, Kim et al. [13] invited 20 experienced players to assess the skill levels of AI bots playing StarCraft. They observed that human rankings were different in several ways to a ranking computed from the bots' competition win rate, because humans weighed certain factors like decision-making skill more heavily. The mismatch between empirical results and perception scores may be because AI bots that are effective against each other proved less effective against humans.

Cheung et al. [4] studied StarCraft from a different perspective, that of non-participant spectators. Their investigations produced a set of nine personas that helped to illuminate *who* these spectators are and *why* they watch. Since shoutcasters are one of the personas, they discussed how shoutcasters affect the spectator experience and how they judiciously decide how and when to reveal different types of information, both to entertain and inform the audience. Another contingent of researchers is working toward applying machine learning to automatically summarize different aspects of sports, potentially assisting sportscasters or shoutcasters in their work. Two examples are automatically generating e-sports statistics [35], and automatically extracting football play diagrams from raw video [34].

	Tournament	Shoutcasters	Players	Game
1	2017 IEM Katowice	ToD and PiG	Neeb vs Jjakji	2
2	2017 IEM Katowice	Rotterdam and Maynarde	Harstem vs TY	1
3	2017 GSL Season 1 Code S	Artosis and tasteless	Soo vs Dark	2
4	2016 WESG Finals	Tenshi and Zeweig	DeMuslim vs iGXY	1
5	2017 StarLeague S1 Premier	Wolf and Brendan	Innovation vs Dark	1
6	2016 KeSPA Cup	Wolf and Brendan	Maru vs Patience	1
7	2016 IEM Geonggi	Kaelaris and Funka	Byun vs Iasonu	2
8	2016 IEM Shanghai	Rotterdam and Nathanias	ShowTime vs Iasonu	3
9	2016 WCS Global Finals	iNcontroL and Rotterdam	Nerchio vs Elazer	2
10	2016 DreamHack Open Leipzig	Rifkin and ZombieGrub	Snute vs ShowTime	3

Table 1. Summary of StarCraft 2 games studied. Please consult our supplementary materials for transcripts and links to videos.

Finally, future attempts to generate automated shoutcasting from replays, using data such as the corpus reported in this paper, could use dialog evaluation systems (e.g., [21]) to assess the quality of the generated shoutcasters.

The closest work to our own is Metoyer et al.'s [22] investigation into the vocabulary and language structure of explaining RTS games. In their study, novices and experts acted in pairs; the novice watched the expert play and asked questions, while the expert thought aloud and answered them. They developed qualitative coding schemes of the content and structure of the explanations the expert players offered. In this paper, we drew upon these coding schemes, with slight modifications. The complete code sets are available in the on-line supplemental materials [7]. Our work differs from all of these works in that our explainers are *expert communicators* about the game (not participant players, programmers, or end users). Our work is also the first to apply IFT to XAI.

METHODOLOGY

In order to study high quality explanations and capable players, we considered only games from professional tournaments denoted as "Premier" by TeamLiquid[1] and also filtered out irrelevant utterances, as described later in this section. We selected 10 matches from the "Premier" pool available with video on demand from professional StarCraft 2 tournaments from 2016 and 2017 (Table 1). Professional matches have multiple games, so we randomly selected one game from each match for analysis. 16 distinct shoutcasters[2] appeared across the 10 videos, with two casters[3] commentating each time.

The rest of this section explains our qualitative coding methodology. In general, we measured rigor using an inter-rater reliability (IRR) rate of 80% over 20% of the data by 2 coders using the Jaccard index, which is the size of the intersection of the codes divided by the size of the union. We exceeded 20% of the data when we needed more test iterations for agreement or when we had to subset the coding. If a codeset was too complex to do in one pass, we coded subcategories. We simplified calculations with different data-subset sizes to ">20%."

First we coded for relevance. Shoutcasters should both inform and *entertain*, so they fill dead air time with jokes. To filter out irrelevant utterances, two researchers independently coded 32% of statements in the corpus as relevant or irrelevant to explaining the game. We achieved a 95% inter-rater reliability. Then, the researchers split up and coded the rest of the corpus.

Research questions RQ1 and RQ2 investigated how the casters seek information onscreen, so we used IFT constructs to discover the types of information casters sought and how they unearthed it. For RQ1 (the patches in which they sought information), we simply counted the casters' navigations among patches. Changes in the display screen identified most of these[4] for us automatically. For RQ2 (*how* they went about their information foraging), we coded the 110 instances of caster navigation by the context where it took place, based on player actions — Building, Fighting, Moving, Scouting — or simply caster navigation. Two researchers independently coded 21% of the data in this manner, with IRR of 80%. After achieving IRR, one researcher coded the remainder of the data.

For RQ3 (implicit questions the shoutcasters answered), we coded the casters' utterances by the Lim & Dey [20] questions they answered. We added a judgment code to capture caster evaluation on the *quality* of actions. The complete code set will be detailed in the RQ3 Results section. Using this code set, two researchers independently coded 34% of the 1024 explanations in the corpus, with 80% inter-rater reliability (Jaccard). After achieving IRR, the researchers split up the remainder of the coding.

For RQ4 (explanation content), we drew content coding rules from Metoyer et al. [22]'s analysis of explaining Wargus games and added some codes to account for differences in gameplay and study structure. (For ease of presentation, in this paper we use the terms "numeric quantity" and "indefinite quantity" instead of their terms "identified discrete" and "indefinite quantity", respectively.) Two researchers independently coded the corpus, one category at a time (e.g., Objects, Actions, ...), achieving an average of 78% IRR on more than 20% of the data in each category. One researcher then finished coding the rest of the corpus. Since all data sources are public, we have provided all data and coding rules in supplementary materials to enable replicability and support further research.

RESULTS

RQ1 Results: What information do shoutcasters seek to generate explanations, and where do they find it?

We used two frameworks to investigate casters' information seeking behaviors. We turned to the Performance, Environ-

[1] http://wiki.teamliquid.net/starcraft2/Premier_Tournaments

[2] Shoutcasters "confirmed" each others' quality in their consistency of utterance type and content (Figure 5).

[3] Here, caster pair (*caster* or *pair* for short) differentiates our observed individuals from the population of shoutcasters as a whole.

[4] But if a caster points at something to bring the other's attention to it—but does not have the mouse—the viewer can't see it.

Figure 1. A screenshot from an analyzed game, modified to highlight the patches available to our casters: *HUD* [1, bottom] (Information about current game state, e.g., resources held, income rate, supply, and upgrade status); *Minimap* [2, lower left] (Zoomed out version of the main window); *"Tab"* [3, top left] (Provides details on demand, currently set on "Production"); *Workers killed* [4, center left] (Shows that 9 Red workers have died recently); *Popup* [5, center] (visualizations that compare player performance, usually shown briefly). Regions 3 and 5 will be detailed in Figures 3 and 4.

ment, Actuators, Sensors (PEAS) model [31] to situate *what* information casters sought in a common framework for conceptualizing intelligent agents. We drew from Information Foraging Theory (IFT) to understand *where* casters did their information seeking, beginning with the places their desired information could be found. These places are called information "patches" in IFT terminology.

Table 2 columns 1 and 2 show the correspondence between PEAS constructs and patches in the game that the casters in our data actually used. **Performance** measures showed assets, resources, successes, and failures, e.g., Figure 1 region 4 (showing that Blue has killed 9 of Red's workers) and region 5 (showing that Blue has killed 19 units to Red's 3, etc.). Table 2 shows that casters rarely consulted performance measures, especially those that examined *past* game states. However, they discussed basic performance measures available in the HUD (Figure 1 region 1), which contained *present* state information, e.g., resources held or upgrade status.

The **Environment** where the agent is situated is the game state (map, structures, units, etc.) shown in Figure 1's main window. We label as Environment any of the patches in Table 2, such as the Units tab, that show *all* the game state data, regardless of whether the players or casters have observed it.

Sensors helped the agent collect information about the environment and corresponded to the local vision area provided by individual units themselves in our domain. Figure 1 region 2 (Minimap) shows a "bird's eye view" of the portion of the environment observable by the Sensors. Casters used patches containing information about Sensors very often, with Minimap and Vision Toggles being among the most used patches in Table 2. The casters had "superpowers" with respect to Sensors (and performance measures) — their interface allowed *full* observation of the environment, whereas players could only *partially* observe it. The casters extensively used the Minimap and the Vision Toggle as they were the only ways for casters to peer through the players' sensors.

Actuators were the means for the agents to interact with their environment, such as building a unit. Figure 1 region 3 (Production Tab) shows some of the Actuators the player was using, namely that Player Blue was building 5 types of objects, whereas Red was building 8. Casters almost always kept visualizations of actions *in progress* on display. RTS actions had a *duration*, meaning that when a player took an action, time passed before its consequence had been realized. The Production tab's popularity was likely due to the fact that it is the *only stable* view of information about actuators and their associated actions.

In fact, prior to the game in our corpus, Pair 3 had this exchange, which demonstrated their perception of the production tab's importance to doing their job,

> Pair 3a: *"What if we took someone who knows literally nothing about StarCraft, just teach them a few phrases and* **what everything is on the production tab***?"*
> Pair 3b: *"Oh, I would be out of a job."*

Implications for an interactive explainer

Abstracting beyond the StarCraft components to the PEAS model revealed a pattern of the casters' behaviors with implications for future explanation systems, which we characterize

	Patch Name	State	Agg.	Usage	Pair 1	Pair 2	Pair 3	Pair 4	Pair 5	Pair 6	Pair 7	Pair 8	Pair 9	Pair 10
Performance	**Units Lost popup**: Shows count and resource value of the units each player has lost.	Past	High	6						2	2	1	1	
	Units Lost tab: Same as above, but as a tab.	Past	High	5	1	1			1	1	1			
	Income tab: Provides resource gain rate.	Present	High	2				1		1				
	Income popup: Shows each player's resource gain rate and worker count.	Present	High	2					1	1				
	Army tab: Shows supply and resource value of currently held non-worker units.	Present	High	1				1						
	Income Advantage graph: Shows time series data comparing resource gain rate.	Past	High	1	1									
	Units Killed popup: Essentially the opposite of the Units Lost popup	Past	High	1	1									
Environment	**Units tab:** Shows currently possessed units.	Present	Low	51	1	2	2	10	1	13	20	2		
	Upgrades tab: Like Units tab, but for upgrades to units and buildings.	Present	Low	5				1	3			1		
	Structures tab: Like Units tab, but buildings.	Present	Low	2	1			1						
Actuator	**Production tab**: Shows the units, structures, and upgrades that are in progress, i.e. have been started, but are yet to finish.	Present	Low	Preferred (by choice) "always-on" tab, not counted										
Sensor	**Minimap:** Shows zoomed out map view	Present	Med	Too many to count										
	Vision Toggle: Shows only the vision available to *one* of the players.	Present	Low	36	5	8	1	2	1	5	1	7	5	1

Table 2. This table illustrates description, classification, and usage rates of the patches and enrichment operations we observed casters using. Each patch is classified by: 1. The part of the PEAS model that this patch illuminates best (column 1), 2. whether it examines past or present game states (column 3), and 3. degree to which the patch aggregates data in its visualization (column 4). The remaining columns show total usage counts, as well as per caster pair usage. Note that there are additional patches passively available (Main Window and HUD) which do not have navigation properties.

as: "keep your Sensors close, but your Actuators closer." This aligns with other research, showing that real-time visualization of agent actions can improve system transparency [39].

However, these results contrast with the explanation systems that tend to prioritize Performance measures. Our results instead suggest that an explanation system should prioritize useful, readily accessible information about what an agent did or can do (Actuators) and of what it can see or has seen (Sensors).

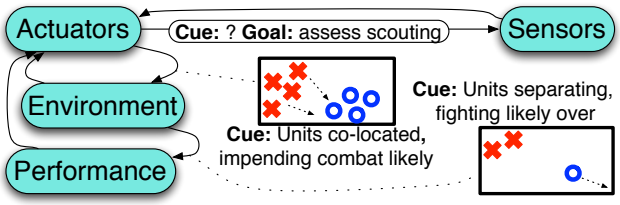

Figure 2. The A-E-P+S loop was a common information foraging strategy some casters used in foraging for agent behavior. It starts at the Actuators, and returns there throughout the foraging process. If a caster interrupted the loop, they usually did so to return to the Actuators.

RQ2 Results: The How: How do shoutcasters seek the information they seek?

Section RQ1 discussed the What and Where (i.e., the content casters sought and locations where they sought it.) We then considered how they decided to move among these places, summarized in Table 2.

The casters' foraging moves seemed to follow a common foraging "loop" through the available information patch types: an Actuators-Environment-Performance loop with occasional forays over to Sensors (Figure 2). Specifically, the casters tended to start at the "always-on" Actuator-related patches of current state's *actions in-progress*; then when something triggered a change in their focus, they checked the Environment for current game state information and occasionally Performance measures of past states. If they needed more information along the way, they went to the Sensors to see through a player's eyes. We will refer to this process as the "A-E-P+S loop."

Information Foraging Theory (IFT) explains why people (information predators) leave one patch to move to another, such when the casters left Actuator patches. According to IFT, predators choose navigations as cost/benefit decisions, based on the value of information in the patch a predator is already in versus the value per cost of going to another patch [29]. Staying in the same patch is generally the least expensive, but when there is less value to be gained by staying versus moving to another patch, the predator moves to the other patch. However, the predator is not omniscient: decisions are based upon the predator's *perception* of the cost and value that other patches will actually deliver. They form these perceptions from both their prior experience with different patch types [27] and from the cues (signposts in their information environment) that point toward content available in other patches.

Certain types of cues tended to trigger a move for the casters. Impending combat was the most common cue triggering a move from the Actuators type (Production tab) to the Environment type (Units tab) — i.e., from A to E in the A-E-P+S loop. In Figure 2, the co-location of opposing units served as the cue. This cue indicated imminent combat, which led to caster navigation to a new patch to illuminate the environment. In fact, combat cues triggered navigations to the Units tab most frequently, accounting for 30 of the 51 navigations there (Table 2).

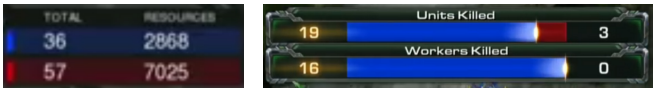

Figure 3. The Units Lost tab (left image) shows the number of units lost and their total value, in terms of resources spent, for both players. In this example from Pair 2, we see that Blue Player (top) has lost merely 2800 minerals worth of units so far in this game, while Red has lost more than 7000. The Units Killed popup (right image) allows shoutcasters to quickly compare player performance via a "tug-of-war" visualization. In this example from Pair 1, as we see that Blue Player (left) has killed 19 units, while Red has killed merely 3. The main difference between these two styles of visualization is that the tab offers more options and information depth to "drill down" into.

Figure 4. The Production tab, showing the build actions currently in progress for each player. Each unit/structure type is represented by a glyph (which serves as a link to navigate to that object), provided a progress bar for duration, and given the number of objects of that type. Thus, we can see that Blue Player (top row) is building 5 different types of things, while Red (bottom row) is building 4 types of things. The Structures, Upgrades, and Units tab look fairly similar to the Production tab.

Interestingly, this cue type was different from the static cues most prior IFT research has used. Cues tended to be static decorations (text or occasionally images) in previous IFT investigations that label a navigation device, like a hyperlink or button that leads to another information patch. In contrast, cues like the onset of combat are dynamic and often did not provide an affordable direct navigation. However, cues like this were considered cues because they "provide users with concise information about content that is not immediately available" [29]. They suggested high value in another location — in the case of combat, the Units tab.

Combat ending was a dynamic cue that triggered a move to a Performance measure. 10 of the 13 navigations to a past-facing Performance measure (via tab or popup), occurred shortly after combat ended as a form of "after-action review" (Table 2). Occasionally, the shoutcasters visited other Performance patches, such as the Income, Units Lost, and Army tabs, to demonstrate reasons why a player had accrued an in-game lead, or the magnitude of that lead (7 navigations). However, signs of completed fighting were the main cues for visiting a Performance patch.

The most common detour out of the A-E-P part of the loop to a Sensor patch was to enrich the information environment via the Vision Toggle (36 navigations, Table 2). The data did not reveal exactly what cue(s) led to this move, but the move itself had a common theme: to assess scouting operations. The casters used the Vision Toggle to allow themselves to see the game through the eyes of only *one* of the players, but their default behavior was to view the game with ALL vision. This provided the casters with the ability to observe *both* players' units and structures simultaneously. Toggling the Vision Sensor in this way enabled them to assess what information was or had been gathered by each player via their scouting actions (29 of the 36 total Vision Toggles), since an enemy unit would only appear to the player's sensors if they had a friendly unit (e.g., a scout) nearby. Toggling the vision Sensor was the second most common patch move.

Besides the act of following cues, IFT has another foraging operation: *enriching* their information environment to make it more valuable or cost-efficient [29]. The aforementioned Vision Toggle was one example of this, and another was when casters added on information visualizations derived from the raw data, like Performance measure popups or other basic visualizations. Two examples of the data obtained through this enrichment are shown in Figure 3.

These Performance measures gave the shoutcasters at-a-glance information about the ways one player was winning. The most commonly used tab, for example, the Units Lost tab (Figure 3), showed the number of units lost and their total value, in terms of resources spent. This measure achieves "at a glance" by aggregating *all* the data samples together by taking a *sum*; derived values like this allow the visualization to scale to large data sets [32]. However, Table 2 indicates that the lower data aggregation patches were more heavily used. The casters used the Production tab to see units grouped by type, as Figure 4 shows, so *type* information was maintained with only *positional* data lost. This contrasts with the Minimap (medium aggregation), in which type information is discarded but positional information maintained at a lower *granularity*. The casters used Performance measure patches primarily to understand present state data (HUD), but these patches were also the only way to access *past* state information (Table 2).

Implications for an interactive explainer

These results have several implications for automated explanation systems in this domain. First, the A-E-P+S loop and how the casters traversed it reveals priority and timing implications for automated explanation systems. For example, the cues that led them to switch to different information patches could also be cues in an automated system about the need to avail different information at appropriate times. For example, our casters showed a strong preference for actuator information as "steady state" visualization, but preferred performance information upon conclusion of a subtask.

Viewing the casters' behaviors through the dual lens of PEAS + IFT has implications for not only the kinds of patches that an explanation system would need to provide, but also the cost to users of not providing these patches in a readily accessible format. For example, PEAS + IFT revealed a costly foraging problem for the casters due to the relative inaccessibility of some Actuator patches.

There is no easily accessible mechanism in StarCraft by which they could navigate to an Actuator patch with fighting or scouting actions in progress. Instead, the only way the casters could get access to these actions was via *painstaking* camera placement. The casters made countless navigations to move the

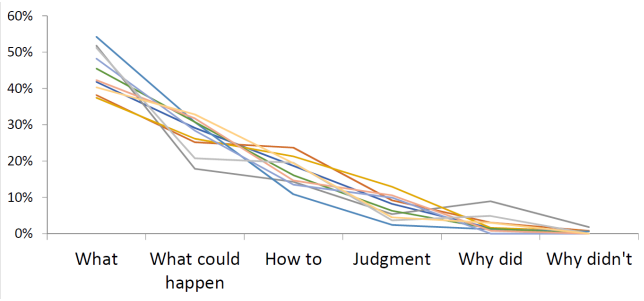

Figure 5. Frequency of Lim & Dey questions answered by casters, with one line per caster pair. Y-Axis represents percentages of the utterances which answered that category of question (X-Axis). Note how casters structured answers consistently.

camera using the Minimap, traditional scrolling, or via tabs with links to the right unit or building.

However, despite all these navigation affordances, sometimes the casters were unable to place the camera on all the actions they needed to see.

For example, at one point when Pair 4 had the camera on a fight at Xy's base, a second fight broke out at DeMuslim's base, which they completely missed:

Pair 4a: <*surprised, noticing something amiss*>
"Xy actually killed the 3rd base of DeMuslim."
...<the pair tries to figure what must have happened>...
Pair 4b: *"Oh my god, you're right Alex."*
Pair 4a: *"Yeah, it was killed during all that action."*

RQ3 Results: What implicit questions do shoutcasters answer and how do they form their answers?

We considered how the shoutcasters gathered and assessed information for the first two research questions. We now shift focus to the explanations themselves.

Much of the prior research into explaining agent behavior starts at some kind of observable effect and then explains something about that effect or its causes [12, 16, 19, 37]. Most such observable effects are the result of player actions in RTS games, and recall from RQ1 that the casters spent most of their information-gathering effort examining the players' Actuators to discover and understand actions.

The casters used the information they gained to craft explanations to answer implicit questions (i.e., questions their audience "should be" wondering) about player actions. Thus, drawing from prior work about the nature of questions people ask about AI, we coded the 1024 casters' explanations using the Lim & Dey "intelligibility types" [20].

The shoutcasters were remarkably consistent (Figure 5) in the types of implicit questions they answered. As Table 3 sums up, casters overwhelmingly chose to answer `What`, with `What-could-happen` and `How-to` high on their list. (The total is greater than 1024 because explanations answered multiple questions and/or fit into multiple categories.)

These results surprised us. Whereas Lim & Dey [19] found that `Why` was the most demanded explanation type from users, the casters rarely provided `Why` answers. More specifically, in the Lim & Dey study, approximately 48 of 250 participants, (19%) demanded a `Why` explanation. To contrast with our study, only 27 of the casters' 1024 utterances (approximately 3%) were `Why` answers.

Discussion and implications for an interactive explainer

Why so few `Why`s? Should an automated explainer, like our shoutcasters, eschew `Why` explanations, in favor of `What`?

One possibility is that the casters delivered exactly what their audience wanted, and thus the casters' distribution of explanation types was well chosen. After all, the casters were experts paid to provide commentary for prestigious tournaments, so they would know their audience well. The expertise level of the audience may have been fairly high, because the tournament videos were available only *on demand* (as opposed to *broadcast* like some professional sports) at websites that casual audience members may not even know about. If a well-informed audience expected the players to do exactly what they did, their expectations would not be violated, which, according to Lim & Dey, suggests less demand for `Why` [19]. This suggests that the extent to which an automated explainer needs to emphasize `Why` explanations may depend on both the *expertise* of the intended audience, which drives their expectations, and the agent's *competence*, which drives failure to meet reasonable expectations.

However, another possibility is that the audience really did want `Why` explanations, but the casters rarely provided them because of the time they required — both theirs and the audience's. The shoutcasters explained in *real time* as the players performed their actions. It takes time to understand the present, predict the future, and link present to future; and spending time in these ways reduces the time allowable for explaining interesting activities happening in present. The corpus showed casters interrupting themselves and each other as new events transpired, as they tried to keep up with the time constraints. This also has implications to the audience's workflow, because it takes time for the audience to mentally process shoutcasters' departures from the present, particularly when interesting actions continuously occur.

Even more critical to an explanation system, `Why` questions also tend to require extra effort (cognitive or computing resources), because they require connecting two time slices:

Pair 10: *"After seeing the first phoenix [flying unit] and, of course, the second one confirmed, Snute is going to invest in a couple spore crawlers [air defense structure]."*

The casters had to connect past information (scouting the phoenix) with a prediction of the future (investing in spore crawlers) in this example.

Answering `Why-didn't` questions was even rarer than answering `Why` questions (Table 3). Like `Why` questions, `Why-didn't` questions required casters to make a connection between previous game state and a potential current or future game state. For example, Pair 2: *"The probe [worker unit] already left a while ago, so we knew it wasn't going to be a pylon [support structure] rush.."* `Why-didn't` answers' rarity is consistent with the finding that understanding a `Why-didn't` explanation requires even more mental effort than a `Why` explanation [20].

Code	*Freq*	*Description*	*Example*
What	595	What the player did or anything about game state	"The liberators are moving forward as well"
What-could-happen	376	What the player could have done or what will happen	"Going to be chasing those medivacs away"
How-to	233	Explaining rules, directives, audience tips, high level strategies	"He should definitely try for the counter attack right away"
*How-good/bad-was-that-action	112	Evaluation of player actions	"Very good snipe there for Neeb"
Why-did	27	Why the player performed an action	"...that allowed Dark to hold onto that 4th base, it allowed him to get those ultralisks out"
Why-didn't	6	Why the player did not perform an action	"The probe already left a while ago, so we knew it wasn't going to be a pylon rush"

Table 3. Utterance type code set, slightly modified from the schema proposed by Lim & Dey. The asterisk denotes the code that we added, How-good/bad-was-that-action because the casters judged actions based on their quality.

As for an interactive explanation system, supporting Why questions requires solving both a *temporal credit assignment problem* (determining the effect of an action taken at a particular time on the outcome) and a *structural* one (determining the effect of a particular system element on the outcome). See [1] for an accessible explanation of these problems.

The casters found a potentially "satisficing" approximation of Why, a combination of What and What-could-happen, the two most frequent explanation types. Their What answers explained what the player did, what happened in the game, and description of the game state. These were all things happening in the present, and did not require the additional cognitive steps required to answer Why or Why-didn't, which may have contributed to its high frequency. Further, the audience needed this kind of "play-by-play" information to stay informed about the game's progression; for example, Pair 4: *"This one hero, marine [inexpensive unit], is starting to kill the vikings [flying unit]."* When adding on What-could-happen, casters were pairing What with what the player will or could do, i.e., a hypothetical outcome. For example,

> Pair 1: *"...if he gets warning of this he'll be able to get back up behind his wall in."*

Although answering the question What-could-happen required predicting the future, it did not also require the casters to tie together information from *past* and future.

The other two frequent answers, How-good/bad-was-that-action and How-to, also sometimes contained "why" information. Casters *judged* an action for How-good/bad-was-that-action, e.g.: Pair 1: *"Nice maneuver from Jjakji, he knows he can't fight Neeb front on right now, he needs to go around the edges."* Casters gave the audience tips and explained high level strategies for How-to. For example, consider this rule-like explanation, which implies "why" the player used a particular army composition: Pair 10: *"Roach [and] ravager [long range unit] in general is really good..."*

The next rule-like How-to example is an even closer approximation to "why" information. Pair 8: *"Obviously when there are 4 protoss units on the other side of the map, you need to produce more zerglings [inexpensive unit], which means even fewer drones [worker unit] for Iasonu."*

The casters are giving a rule in this case: given a general game state (protoss units on their side of the map) the player should perform an action (produce zerglings). However, the example does more; it also implies a Why answer to the question "Why isn't Iasonu making more drones?" Since this implied answer simply relates the present to a rule or best practice, it was produced at much lower expense than a true Why answer that required tying past events to the present.

Mechanisms casters used to circumvent the need for disruptive and resource-intensive Why explanations, such as using How-to, may also be ways to alleviate the same problems in explanation systems.

RQ4 Results: What relationships and objects do shoutcasters use when building their explanations?

To inform future explanation systems' content by expert explanations — the patterns of nouns, verbs, and adjectives/adverbs in these professionally crafted explanations — we drew upon a code set from prior work [22] (see the Methodology section). Table 4 shows how much caster pairs used each of these types of content, grouping the objects (nouns) in the first group of columns, then actions (verbs), and then properties (adjectives and adverbs). Table 5 shows how the casters' explanations used these concepts *together*, i.e., which properties they paired with which objects and actions.

The casters' explanation sentences tended to be noun-verb constructions, so we began with the nouns. The most frequently described objects were *fighting object*, *production object*, and *enemy*, with frequencies of 49%, 34%, and 16%, respectively (Figure 4). (This is similar to results from [22], where production, fighting, and enemy objects were the three most popular object subcodes.) As to the actions ("verbs"), the casters mainly discussed *fighting* (40%) and *building* (23%). It is not surprising that the casters frequently discussed *fighting*, since combat skills are important in StarCraft [13], and *producing* is often a prerequisite to *fighting*. This may suggest that, in RTS environments, an explanation system may be able to focus on only the most important subset of actions and objects, without needing to track and reason about most of the others.

The casters were quite strategic in how they put together these nouns and verbs with properties. The casters' used particular properties with these nouns and verbs to paint the bigger picture of how the game was going for each player, and how that tied to the players' strategies. We illustrate in the next subsections a few of the ways casters communicated about player decisions — succinctly enough for real time.

	enemy	fighting object	vision object	production object	environmental object	unspecified object	Upgrade object	building/producing	fighting	Scouting	Moving	distance	point/region	size	arrangement	ordering	timing	speed	repetition	indefinite quantity	numeric quantity	comparative	absolute
Pair 1	20%	53%	2%	39%	0%	11%	5%	34%	47%	3%	2%	9%	30%	0%	13%	31%	26%	3%	20%	21%	27%	10%	7%
Pair 2	22%	43%	1%	37%	0%	9%	2%	17%	41%	7%	2%	10%	30%	0%	11%	28%	12%	2%	12%	20%	34%	10%	3%
Pair 3	16%	52%	0%	27%	0%	5%	16%	39%	36%	0%	2%	14%	23%	0%	9%	36%	41%	11%	11%	7%	27%	20%	5%
Pair 4	14%	43%	2%	23%	3%	7%	10%	23%	49%	1%	0%	11%	18%	0%	12%	9%	18%	7%	8%	13%	20%	10%	4%
Pair 5	20%	44%	8%	28%	1%	2%	9%	23%	34%	5%	4%	6%	17%	1%	4%	13%	18%	11%	13%	15%	31%	8%	4%
Pair 6	16%	38%	8%	35%	0%	3%	2%	17%	41%	4%	4%	9%	19%	0%	9%	23%	24%	5%	16%	10%	22%	14%	6%
Pair 7	17%	56%	4%	40%	1%	4%	4%	16%	40%	6%	6%	16%	22%	1%	10%	18%	18%	6%	19%	8%	44%	12%	6%
Pair 8	18%	59%	0%	53%	0%	0%	2%	30%	38%	4%	8%	16%	29%	1%	10%	55%	40%	15%	25%	16%	27%	6%	10%
Pair 9	8%	48%	0%	23%	0%	8%	5%	15%	38%	2%	3%	11%	18%	0%	9%	34%	31%	3%	20%	17%	34%	11%	5%
Pair 10	9%	53%	5%	40%	7%	2%	3%	19%	33%	7%	2%	9%	17%	5%	3%	14%	12%	17%	14%	9%	24%	7%	7%
Mean	16%	49%	3%	34%	1%	5%	6%	23%	40%	4%	3%	11%	22%	1%	9%	26%	24%	8%	16%	14%	29%	11%	6%
Stdev	5%	7%	3%	9%	2%	4%	5%	8%	5%	2%	2%	3%	5%	2%	3%	14%	10%	5%	5%	5%	7%	4%	2%

Table 4. Occurrence frequencies for each code, as a percent of the total number of utterances in the corpus. From left to right: *Object* (pink), *Action* (orange), *Spatial* (green), *Temporal* (yellow), and *Quantitative* (blue) codes. The casters were consistent about kinds of the content they rarely included, but inconsistent about the kinds of content they most favored.

"This part of the map is mine!": Spatial properties

RTS players claim territory in battles with the *arrangement* of their military units, e.g.:

> Pair 3: *"He's actually arcing these roaches [ranged unit] out in such a great way so that he's going to block anything that's going to try to come back."*

As the *arrangement* column of Table 5 shows, the objects that were used most with *arrangment* were *fighting objects* (12%, 72 instances) and *enemy*, (10%, 26 instances). Note that *arrangement* is very similar to *point/region*, but at a smaller scale; *Arrangement* of *production object*, such as exactly where buildings are placed in one's base, appeared to be less significant, co-occurring only 5% of the time.

The degree to which an RTS player wishes to be aggressive or passive is often evident in their choice of what *distance* to keep from their opponent, and the casters often took this into account in their explanations. One example of this was evaluation of potential new base locations.

> Pair 5: *"...if he takes the one [base] thats closer that's near his natural [base], then it's close to Innovation so he can harass."*

Here, the casters communicated the control of parts of the map by describing *bases* as a *region*, and then relating two regions with a *distance*. The magnitude of that distance then informed whether the player was able to more easily attack. Of the casters' utterances that described *distance* along with *production object*, 27 out of 44 referred to the distance between bases or moving to/from a base.

"When should I...": Temporal properties

Casters' explanations often reflected players' priorities for allocating limited resources. One way they did so was using

		Properties											
		distance	point/region	size	arrangement	ordering	timing	speed	repetition	indefinite	numeric	comparative	absolute
Nouns	enemy	11%	12%	0%	10%	12%	8%	3%	10%	10%	11%	8%	6%
	fighting object	12%	16%	0%	12%	18%	15%	6%	13%	15%	15%	9%	7%
	vision object	3%	4%	2%	4%	3%	3%	4%	3%	3%	2%	2%	4%
	production object	10%	16%	1%	5%	17%	18%	8%	17%	8%	28%	8%	4%
	environmental object	2%	2%	10%	2%	1%	1%	2%	0%	0%	1%	1%	0%
	unspecified object	2%	5%	0%	3%	4%	4%	2%	4%	8%	4%	11%	6%
	Upgrade object	1%	1%	0%	0%	6%	10%	11%	3%	1%	10%	6%	4%
Verbs	building/producing	3%	7%	0%	2%	16%	21%	12%	18%	7%	20%	6%	3%
	fighting	14%	19%	1%	12%	19%	13%	5%	12%	15%	16%	8%	7%
	Scouting	2%	5%	2%	4%	5%	3%	2%	3%	3%	2%	1%	4%
	Moving	8%	8%	3%	5%	4%	3%	4%	2%	2%	3%	1%	2%

Table 5. Co-Occurrence Matrix. Across rows: *Object* (pink, top rows) and *Action* (orange, bottom rows) codes. Across columns: *Spatial* (green, left), *Temporal* (yellow, center), and *Quantitative* (blue, right). Co-occurrence rates were calculated by dividing the intersection of the subcodes by the union.

speed properties: Pair 4: *"We see a really quick third [base] here from XY, like five minutes third."* Since extra bases provide additional resource gathering capacity, the audience could infer that the player intended to follow an "economic" strategy, as those resources could have otherwise been spent on military units or upgrades. This contrasts with the following example, Pair 8: *"He's going for very fast lurker den.. [unit production building]."* The second example indicated the player's intent to follow a different strategy: unlocking stronger units (lurkers). *Speed* co-occurred with *building/producing* most often (12%, 36 instances).

"Do I care how many?": Quantitative properties

We found it surprising how often the casters described quantities without numbers. In fact, the casters often did not even include *type* information when they described the players' holdings, instead focusing on *comparative* properties (Table 5). For example, Pair 1: *"There is too much supply for him to handle. Neeb finalizes the score here after a fantastic game."* Here, "supply"[5] is so generic, we do not even know what kind of things Neeb had — only that he had "too much" of it.

In contrast, when the casters discussed cheap military units, like "marines" and "zerglings," they tended to provide *type* information, but about half of their mentions still included no precise numbers. Perhaps it was a matter of the high cost to get that information: cheap units are often built in large quantities, so deriving a precise quantity is often very tedious. Further, adding one weak unit that is cheap to build has little impact on army strength, so getting a precise number may not have been worthwhile — i.e. the value of knowing precise quantities is low. Consider the following example, which quantified the army size of both players vaguely, using *indefinite quantity* properties: Pair 6: *"That's a lot of marines and marauders [heavy infantry unit] and not enough stalkers [mobile unit]."*

Workers are a very important unit in the RTS domain. Consistent with this importance, workers are the only unit where the casters were automatically alerted to their death (Figure 1, region 4), and are also available at a glance on the HUD (Figure 1, region 1). Correspondingly, the casters often gave precise quantities of workers (a *production object*). Workers (workers, drones, scvs, and probes) had 46 co-occurrences with *numeric quantities*, but only 12 with *indefinite quantities* (e.g., lot, some, few). Pair 2: *"...it really feels like Harstem is doing everything right, and [yet] somehow ended up losing 5 workers."*

Implications for an interactive explainer

These results have particularly important implications for interactive explanation systems with real-time constraints. Namely, the results suggest that an effective way to communicate about strategies and tactics is to modify the critical objects and actions with particular properties that suggest strategies. This not only affords a *succinct* way to communicate about strategies and tactics (fewer words) but also a *lighter load* for the audience than attempting to build and process a rigorous explanation of strategy.

Specifically, spatial properties can communicate beyond the actual properties of objects to strategies themselves; e.g., casters used distance to point out plans to attack or defend. Temporal properties can be used in explanations of strategies when choices in resource allocation determines available strategies.

Finally, an interactive explanation system could use the quantitative property results to help ensure alignment in the level of abstraction used by the human and the system. For example, a player can abstract a quantity of units into a *single group* or think of them as *individual units*. Knowing the level of abstraction the human players use in different situations can help an interactive explanation system choose the level of abstraction that will meet human expectations. Using properties in these strategic ways may enable an interactive explanation system to meet its real-time constraints while at the same time improving its communicativeness to the audience.

[5]"Supply" is used in an overloaded fashion here, while looking at *military units*, as opposed to the traditional definition — maximum army size the player can have at one time.

CONCLUSION

The results of our study suggest that explaining intelligent agents to humans has much to gain from looking to the human experts. The expert explainers in our case — RTS shoutcasters — revealed implications into what, when, and how human audiences of such systems need explanations, and how real-time constraints can come together with explanation-building strategies. Among the results we learned were:

RQ1 Investigating the what's and where's of casters' real-time information foraging to *assess and understand* the players showed that the most commonly used *patches* of the information environment were the Actuators ("A" in the PEAS model). This suggests that explanation systems that currently support only Performance measures should consider also presenting information from the Actuators and Sensors.

RQ2 The how's of casters' foraging revealed a common pattern, which we termed the A-E-P+S loop, and the most common cues and triggers that led shoutcasters to move through this loop. Future explanation systems may be well-served to prioritize and recommend explanations according to this loop and its triggers.

RQ3 As *model explainer*, the casters revealed strategies for "satisficing" with explanations that may not have precisely answered all the questions the audience had in mind, but were feasible given the time and resource constraints in effect when comprehending, assessing, and explaining, all in *real time* as play progresses. These strategies may be likewise applicable to interactive explanation systems.

RQ4 The detailed contents of the casters' explanations revealed patterns of how they paired properties ("adjectives and adverbs") with different objects ("nouns") and actions ("verbs"). Interactive explanation systems may be able to leverage these patterns to communicate succinctly about an agent's tactics and strategies.

Ultimately, both shoutcasters' and explanation systems' jobs are to improve the audience members' mental model of the agents' behavior. As Cheung, et al. [4] put it, "...commentators are valued for their ability to expose the depth of the game." Hopefully, future explanation systems will be valued for the same reason.

ACKNOWLEDGMENTS

This work was supported by DARPA #N66001-17-2-4030 and NSF #1314384. Any opinions, findings and conclusions or recommendations expressed are the authors' and do not necessarily reflect the views of NSF, DARPA, the Army Research Office, or the US government.

REFERENCES

1. Adrian K Agogino and Kagan Tumer. 2004. Unifying temporal and structural credit assignment problems. In

Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2. IEEE Computer Society, 980–987.

2. Svetlin Bostandjiev, John O'Donovan, and Tobias Höllerer. 2012. TasteWeights: A visual interactive hybrid recommender system. In *Proceedings of the Sixth ACM Conference on Recommender Systems*. ACM, 35–42.
3. Nico Castelli, Corinna Ogonowski, Timo Jakobi, Martin Stein, Gunnar Stevens, and Volker Wulf. 2017. What happened in my home?: An end-user development approach for smart home data visualization. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 853–866.
4. Gifford Cheung and Jeff Huang. 2011. Starcraft from the stands: Understanding the game spectator. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 763–772. DOI: http://dx.doi.org/10.1145/1978942.1979053
5. Ed H Chi, Peter Pirolli, Kim Chen, and James Pitkow. 2001. Using information scent to model user information needs and actions and the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 490–497.
6. Kelley Cotter, Janghee Cho, and Emilee Rader. 2017. Explaining the news feed algorithm: An analysis of the "News Feed FYI" blog. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1553–1560.
7. Jonathan Dodge et al. 2018. Supplemental materials: How the experts do it: Assessing and explaining agent behaviors in real-time strategy games. web site. (2018). Retrieved December 28, 2017 from http://web.engr.oregonstate.edu/~burnett/XAI-CHI2018-rebuilt_supplementary_materials/.
8. Scott D. Fleming, Chris Scaffidi, David Piorkowski, Margaret Burnett, Rachel Bellamy, Joseph Lawrance, and Irwin Kwan. 2013. An information foraging theory perspective on tools for debugging, refactoring, and reuse tasks. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 22, 2 (2013), 14.
9. Wai-Tat Fu and Peter Pirolli. 2007. SNIF-ACT: A cognitive model of user navigation on the world wide web. *Human-Computer Interaction* 22, 4 (2007), 355–412.
10. Alex Groce, Todd Kulesza, Chaoqiang Zhang, Shalini Shamasunder, Margaret Burnett, Weng-Keen Wong, Simone Stumpf, Shubhomoy Das, Amber Shinsel, Forrest Bice, and others. 2014. You are the only possible oracle: Effective test selection for end users of interactive machine learning systems. *IEEE Transactions on Software Engineering* 40, 3 (2014), 307–323.
11. Robert R Hoffman and Gary Klein. 2017. Explaining explanation, part 1: theoretical foundations. *IEEE Intelligent Systems* 32, 3 (2017), 68–73.
12. Ashish Kapoor, Bongshin Lee, Desney Tan, and Eric Horvitz. 2010. Interactive optimization for steering machine classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1343–1352.
13. Man-Je Kim, Kyung-Joong Kim, SeungJun Kim, and Anind K Dey. 2016. Evaluation of starcraft artificial intelligence competition bots by experienced human players. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 1915–1921.
14. Josua Krause, Adam Perer, and Kenney Ng. 2016. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 5686–5697. DOI: http://dx.doi.org/10.1145/2858036.2858529
15. Cliff Kuang. 2017. Can AI be taught to explain itself? New York Times, (2017). Retrieved December 26, 2017 from https://www.nytimes.com/2017/11/21/magazine/can-ai-be-taught-to-explain-itself.html.
16. Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. ACM, 126–137.
17. Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. 2012. Tell me more? The effects of mental model soundness on personalizing an intelligent agent. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1–10.
18. Sandeep Kaur Kuttal, Anita Sarma, and Gregg Rothermel. 2013. Predator behavior in the wild web world of bugs: An information foraging theory perspective. In *Visual Languages and Human-Centric Computing (VL/HCC), 2013 IEEE Symposium on*. IEEE, 59–66.
19. Brian Y Lim and Anind K Dey. 2009. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing*. ACM, 195–204.
20. Brian Y. Lim, Anind K. Dey, and Daniel Avrahami. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2119–2128.
21. Diane Litman, Steve Young, M.J.F. Gales, Kate Knill, Karen Ottewell, Rogier van Dalen, and David Vandyke. 2016. Towards using conversations with spoken dialogue systems in the automated assessment of non-native speakers of English. In *SIGDIAL Conference*. 270–275.
22. Ronald Metoyer, Simone Stumpf, Christoph Neumann, Jonathan Dodge, Jill Cao, and Aaron Schnabel. 2010. Explaining how to play real-time strategy games. *Knowledge-Based Systems* 23, 4 (2010), 295–301.

23. Nan Niu, Anas Mahmoud, Zhangji Chen, and Gary Bradshaw. 2013. Departures from optimality: Understanding human analyst's information foraging in assisted requirements tracing. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 572–581.
24. Donald A Norman. 1983. Some observations on mental models. *Mental models* 7, 112 (1983), 7–14.
25. S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. 2013. A survey of real-time strategy game AI research and competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games* 5, 4 (Dec 2013), 293–311. DOI: http://dx.doi.org/10.1109/TCIAIG.2013.2286295
26. Alexandre Perez and Rui Abreu. 2014. A diagnosis-based approach to software comprehension. In *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 37–47.
27. David Piorkowski, Scott D. Fleming, Christopher Scaffidi, Margaret Burnett, Irwin Kwan, Austin Z Henley, Jamie Macbeth, Charles Hill, and Amber Horvath. 2015. To fix or to learn? How production bias affects developers' information foraging during debugging. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE, 11–20.
28. David Piorkowski, Austin Z Henley, Tahmid Nabi, Scott D Fleming, Christopher Scaffidi, and Margaret Burnett. 2016. Foraging and navigations, fundamentally: Developers' predictions of value and cost. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 97–108.
29. Peter Pirolli. 2007. *Information foraging theory: Adaptive interaction with information*. Oxford University Press.
30. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144.
31. Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A modern approach* (2 ed.). Pearson Education.
32. Robert Spence. 2007. *Information Visualization: Design for interaction (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
33. Sruti Srinivasa Ragavan, Sandeep Kaur Kuttal, Charles Hill, Anita Sarma, David Piorkowski, and Margaret Burnett. 2016. Foraging among an overabundance of similar variants. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 3509–3521.
34. David J Stracuzzi, Alan Fern, Kamal Ali, Robin Hess, Jervis Pinto, Nan Li, Tolga Konik, and Daniel G Shapiro. 2011. An application of transfer to american football: From observation of raw video to control in a simulated environment. *AI Magazine* 32, 2 (2011), 107–125.
35. Adam Summerville, Michael Cook, and Ben Steenhuisen. 2016. Draft-Analysis of the Ancients: Predicting Draft Picks in DotA 2 using Machine Learning. (2016). https://aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/view/14075
36. Katia Sycara, Christian Lebiere, Yulong Pei, Donald Morrison, and Michael Lewis. 2015. Abstraction of analytical models from cognitive models of human control of robotic swarms. In *International Conference on Cognitive Modeling*. University of Pittsburgh.
37. Joe Tullio, Anind K Dey, Jason Chalecki, and James Fogarty. 2007. How it works: A field study of non-technical users interacting with an intelligent system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 31–40.
38. Oriol Vinyals et al. 2017. StarCraft II: A New Challenge for Reinforcement Learning. Tech Report. (2017). Retrieved December 22, 2017 from https://deepmind.com/documents/110/sc2le.pdf.
39. Robert H Wortham, Andreas Theodorou, and Joanna J Bryson. 2017. Improving robot transparency:real-time visualisation of robot AI substantially improves understanding in naive observers, In IEEE RO-MAN 2017. *IEEE RO-MAN 2017* (August 2017). http://opus.bath.ac.uk/55793/