Automated Assistants to Aid Humans in Understanding Team Behaviors

Taylor Raines, Milind Tambe, Stacy Marsella

Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292, USA
{Raines, Tambe, Marsella}@isi.edu

Abstract. Multi-agent teamwork is critical in a large number of agent applications, including training, education, virtual enterprises and collective robotics. Tools that can help humans analyze, evaluate, and understand team behaviors are becoming increasingly important as well. We have taken a step towards building such a tool by creating an automated analyst agent called ISAAC for post-hoc, off-line agent-team analysis. ISAAC's novelty stems from a key design constraint that arises in team analysis: multiple types of models of team behavior are necessary to analyze different granularities of team events, including agent actions, interactions, and global performance. These heterogeneous team models are automatically acquired via machine learning over teams' external behavior traces, where the specific learning techniques are tailored to the particular model learned. Additionally, ISAAC employs multiple presentation techniques that can aid human understanding of the analyses. This paper presents ISAAC's general conceptual framework, motivating its design, as well as its concrete application in the domain of RoboCup soccer. In the RoboCup domain, ISAAC was used prior to and during the RoboCup'99 tournament, and was awarded the RoboCup scientific challenge award.

1 Introduction

Teamwork has been a growing area of agent research and development in recent years, seen in a large number of multi-agent applications, including autonomous multi-robotic space missions [5], virtual environments for training [16] and education [8], and software agents on the Internet [15]. With the growing importance of teamwork, there is now a critical need for tools to help humans analyze, evaluate, and understand team behaviors. Indeed, in multi-agent domains with tens or even hundreds of agents in teams, agent interactions are often highly complex and dynamic, making it difficult for human developers to analyze agent-team behaviors.

M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI 1856, pp. 85–102, 2000.

The problem is further exacerbated in environments where agents are developed by different developers, where even the intended interactions are unpredictable.

Unfortunately, the problem of analyzing team behavior to aid human developers in understanding and improving team performance has been largely unaddressed. Previous work in agent teamwork has largely focused on guiding autonomous agents in their teamwork [6, 17], but not on its analysis for humans. Agent explanation systems, such as Debrief [7], allow individual agents to explain their actions based on internal state, but do not have the means for a team analysis. Recent work on multi-agent visualization systems, such as [9], has been motivated by multi-agent understandability concerns (similar to ours), but it still leaves analysis of agent actions and interactions to humans.

This paper focuses on agents that assist humans to analyze, understand and improve multi-agent team behaviors by (i) locating key aspects of team behaviors that are critical in team success or failures; (ii) diagnosing such team behaviors, particularly, problematic behaviors; (iii) suggesting alternative courses of action; and (iv) presenting the relevant information to the user comprehensibly. To accomplish these goals, we have developed an agent called ISAAC. A fundamental design constraint here is that unlike systems that focus on explaining individual agent behaviors [7, 12], team analysts such as ISAAC cannot focus on any single agent or any single perspective or any single granularity (in terms of time-scales). Instead, when analyzing teams, multiple perspectives at multiple levels of granularity are important. Thus, while it is sometimes beneficial to analyze the critical actions of single individuals, at other times it is the collaborative agent interaction that is key in team success or failure and requires analysis, and yet at other times an analysis of the global behavior trends of the entire team is important.

To enable analysis from such multiple perspectives, ISAAC relies on multiple models of team behavior, each covering a different level of granularity of team behavior. More specifically, ISAAC relies on three heterogeneous models that analyze events at three separate levels of granularity: an individual agent action, agent interactions, and overall team behavior. These models are automatically acquired using different methods (inductive learning and pattern matching) -- indeed, with multiple models, the method of acquisition can be tailored to the model being acquired.

Yet, team analysts such as ISAAC must not only be experts in team analysis, they must also be experts in conveying this information to humans. The constraint of multiple models has strong implications for the type of presentation as well. Analysis of an agent action can show the action and highlight features of that action that played a prominent role in its success or failure, but a similar presentation would be incongruous for a global analysis, since no single action would suffice. Global analysis requires a more comprehensive explanation that ties together seemingly unconnected aspects and trends of team behavior. ISAAC uses a natural language summary to explain the team's overall performance, using its multimedia viewer to show examples where appropriate. The content for the summary is chosen based on ISAAC's analysis of key factors determining the outcome of the engagement.

Additionally, ISAAC presents alternative courses of action to improve a team using a technique called 'perturbation analysis'. A key feature of perturbation analysis is that it finds actions within the agents' skill set, such that recommendations are plausible. In particular, this analysis mines data from actions that the team has already performed.

Overall, ISAAC performs post-hoc, off-line analysis of teams using agent-behavior traces in the domain. This analysis is performed using data mining and inductive learning techniques. Analyzing the teams off-line alleviates time constraints for these analysis techniques, allowing a more thorough analysis. Also, using data from the agents' external behavior traces, ISAAC is able to analyze a team without necessarily understanding its internals, allowing analysis of teams developed by different developers in a given domain.

ISAAC is currently applied in the domain of RoboCup soccer simulation [8]. RoboCup is a dynamic, multi-agent environment developed to explore multi-agent research issues, with agent teams participating in annual competitions. Agent-team analysis is posed as a fundamental challenge in RoboCup since team developers wish to understand the strengths and weaknesses of teams and understand how to improve such teams. (There are at least 50 such development groups around the world.) Indeed, ISAAC has been applied to all of the teams from several RoboCup tournaments in a fully automated fashion. This analysis has revealed many interesting results including surprising weaknesses of the leading teams in both the RoboCup '97 and RoboCup '98 tournaments and provided natural language summaries at RoboCup '99. ISAAC was also awarded the 'Scientific Challenge Award' at the RoboCup '99 international tournament. ISAAC is available on the web at http://coach.isi.edu and has been used remotely by teams preparing for these competitions.

While ISAAC is currently applied in RoboCup, ISAAC's techniques are intended to apply in other team domains such as agent-teams in foraging and exploration [2] and battlefield simulations [16]. For example, exploring actions, interactions, and global trends such as target hit rate, friendly fire damage, and formation balance, ISAAC could produce a similar analysis in the battlefield simulation domain, and use similar presentation techniques as well.

2 Overview of ISAAC

We use a two-tiered approach to the team analysis problem. The first step is acquiring models that will compactly describe team behavior, providing a basis for analyzing the behavior of the team. As mentioned earlier, this involves using multiple models at different levels of granularity to capture various aspects of team performance. The second step is to make efficient use of these models in analyzing the team and presenting this analysis to the user. Later sections delve into more specifics of these models. An overview of the entire process is shown in Figure 1.

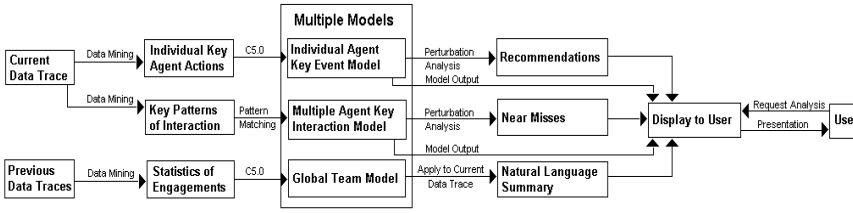

Fig. 1. Flow Chart for ISAAC Model Generation and Analysis

Input to all models comes in the form of data traces of agent behaviors. In the current implementation of ISAAC, these traces have been uploaded from users around the world through the Internet.

As shown in Figure 1, acquiring the models involves a mix of data mining and inductive learning but is specific to the granularity of analysis being modeled. Analysis of an individual agent action (individual agent key event model) uses the C5.0 decision tree inductive learning algorithm, an extension to C4.5, to create rules of success or failure [10]. For analysis of agent interactions (multiple agent key interaction model), pre-defined patterns are matched to find prevalent patterns of success. To develop rules of team successes or failures (global team model), game level statistics are mined from all available previous games and again inductive learning is used to determine reasons for success and failure.

Utilizing the models involves catering the presentation to the granularity of analysis to maximize human understandability. ISAAC uses different presentation techniques in each situation. For the individual agent key event model, the rules and the cases they govern are displayed to the user. By themselves, the features that compose a rule provide implicit advice for improving the team. To further elucidate, a multimedia viewer is used to show cases matching the rule, allowing the user to better understand the situation and to validate the rules (See figure 2). A perturbation analysis is then performed to recommend changes to the team by changing the rule condition by condition and mining cases of success and failure for this perturbed rule. The cases of this analysis are also displayed in the multimedia viewer, enabling the user to verify or refute the analysis.

For the multiple agent key interaction model, patterns of agent actions are analyzed similar to the individual agent actions. A perturbation analysis is also performed here, to find patterns that are similar to successful patterns but were unsuccessful. Both successful patterns and these 'near misses' are displayed to the user as implicit advice. This model makes no recommendations, but does allow the user to scrutinize these cases.

The global team model requires a different method of presentation. For the analysis of overall team performance, the current engagement is matched against previous rules, and if there are any matches, ISAAC concludes that the reasons given by the rule were the determining factors in the result of the engagement. A natural lan-

guage summary of the engagement is generated using this rule for content selection and sentence planning. ISAAC makes use of the multimedia display here as well, linking text in the summary to corresponding selected highlights.

Fig. 2. Multimedia Viewer highlighting key features

ISAAC has been used in the RoboCup simulated soccer environment consisting of two opposing teams of eleven agents each. The agents do not have a centralized control, and act in a complex, dynamic, noisy environment managed by the soccer server, which acts as host and referee for the game. Figure 2 shows ISAAC's multimedia viewer, which displays the soccer field and plays from the games, and can highlight key features specific to ISAAC's analysis. For instance in figure 2, the area around the right soccer goal is highlighted.

3 Individual Agent Key Event Model

This section examines the first of ISAAC's three models, focusing on key actions taken by individual agents, and is specific to each team. In this and the following two sections, we first provide a conceptual overview of the model being analyzed and then discuss its instantiation in RoboCup.

3.1 Conceptual Overview of the Individual Agent Key Event Model

The individual agent model focuses on the analysis of critical events in a team's behavior history relevant to the team's success. There may be many critical events along the path to the team's eventual success or failure that are widely separated in time, only loosely coupled to each other, but nevertheless critical to the team's success. For instance, in a battlefield simulation, there may be many distinct attacks on

enemy units, which are critical to team success, embedded in a larger history of maneuvering.

We consider critical events to be the team's intermediate successes or failures. When something occurs that directly influences the team's eventual success or failure, this is considered to be an intermediate success or failure point. At present, we assume the identification of these intermediate points is part of the domain specific knowledge available to the individual agent analysis model.

Having isolated cases of intermediate success or failure, we can now form rules of successful and unsuccessful behavior, which comprise the individual agent model. These rules are formed using inductive learning techniques over the cases of success and failure based on a set of potentially relevant features in these cases. These features, along with the decision on what are the cases of intermediate success, are the only background information or bias given to the individual agent analysis technique. The features chosen must have the breadth to cover all information necessary for the analysis, but should also be independent of each other if at all possible. In the future, a semi-automated attribute selection may be used [4].

Currently, C5.0 is used to form the rules of success and failure. Each rule describes a class of success or failure cases, based on its feature description. These rules and the cases they represent can be displayed to the user as implicit advice on how individual agents operate in critical situations.

More explicit exploration of this advice is performed using an automated *perturbation analysis*. After ISAAC has produced rules determining which circumstances govern success and failure classifications, ISAAC uses a perturbation analysis to determine which changes would produce the most benefit. Each learned rule consists of a number of conditions. We define a perturbation to be the rule that results from reversing one condition. Thus a rule with N conditions will have N perturbations. The successes and failures governed by the perturbations of a rule are mined from the data and examined to determine which conditions have the most effect in changing the outcome of the original rule, turning a failure into a success. Since these cases are mined from the original data traces, the recommended changes must already be within the agent's skill set. Perturbation analysis is explained in greater detail in Section 3.3.

3.2 Application of Individual Agent Key Event Model to RoboCup

In applying the approach to RoboCup, the domain specific information has to be identified that would be used by ISAAC as bias in its analysis. In particular, in the RoboCup domain, success means outscoring the opponent. Shots on goal are therefore key points of intermediate success or failure as these are situations that can directly affect the outcome of the game. Thus, the focus of ISAAC's individual agent analysis in RoboCup is shots on a team's goal as well as shots by the team on an opponent's goal.

Having defined shots on goal as key events, we need to determine which domain dependant features might be useful in classifying the success or failure of a shot on goal. After an initial set of experiments with a relatively large feature set, ISAAC currently relies on a set of 8 features to characterize successes and failures.

Having determined which features to use in the analysis and the key events (the cases) to examine, the task is transformed to mining the raw data and feeding it to the C5.0 decision tree learning algorithm. From the resulting decision tree, C5.0 forms rules representing distinct paths in the tree from the root of the tree to a leaf as a classification of success (goal-score) or failure (goal not scored). Each rule describes a class of similar successes or failures.

Figure 3 shows an example success rule, describing a rule where shots taken on the Windmill Wanderer team will fail to score (Successful Defense). This rule states that when the closest defender is sufficiently far away (>13.6 m) and sufficiently close to the shooter's path to the center of the goal (<8.98°), and the shooter is towards the edges of the field (>40.77°), Windmill Wanderer will successfully defend against this shot. When viewed using ISAAC, the user can see that the defender is far enough away to have sufficient time to adjust and intercept the ball in most of these cases. Thus the user is able to validate ISAAC's analysis. This rule provides implicit advice to this team to keep a defender sufficiently distant from the ball, or to try to keep the ball out of the center of the field.

```
Distance of Closest Defender > 13.6 m
Angle of Closest Defender wrt Goal <= 8.981711
Angle from Center of Field > 40.77474
→ class Successful Defense
```

Fig. 3. Sample Rule from shots on Windmill Wanderer team of RoboCup'98

The application of a decision tree induction algorithm to this analysis problem must address some special concerns. The goal-shot data has many more failure cases (failed goal shots) than success cases (goals scored). However, analyzing such data using a traditional decision tree induction algorithm such as C4.5 gives equal weight to the cost of misclassifying successes and failures. This usually yields more misclassified success cases than misclassified failure cases. For example, in our analysis of shots by the Andhill team from the RoboCup'97 tournament, our original analysis misclassified 3 of 306 failure cases (less than 1%), but misclassified 18 of 68 success cases (26%). Since a much larger portion of the success cases is incorrectly classified, this produces overly specific rules that govern success cases. To compensate for this lopsided data set, the ability of C5.0 to weight the cost of misclassification is used. Specifically, the cost of misclassifying a success case is set to be greater than the cost of misclassifying a failure case [18]. ISAAC uses a 3 to 1 ratio by default, but this is adjustable.

More generally, differential weighting of misclassification cost provides a mechanism for tailoring the level of aggressiveness or defensiveness of ISAAC's analysis. Consider shots on goal against a team. If a very high cost is assigned to misclassifying a successful shot on goal, the rules produced will likely cover all successful shots, and quite a few misclassified failure cases. In this case, the rule conditions are implicitly advising to make the team very defensive. On the other hand, if a low cost is assigned, the rules may not cover all of the successful cases. Therefore, ISAAC would only give "advice" relevant to stopping the majority of shots on goal. This may not be appropriate if we consider any goal to be a serious failure. Therefore, we allow the user to adjust the weight on success case misclassifications.

3.3 Perturbation Analysis

Perturbations of a failure rule enable users to see what minimal modifications could be made to agent behaviors to convert the failures into success. Mining instances of perturbed failure rules, the developer determines steps that could be taken to move the agent from failure to successful behavior.

For example, one of ISAAC's rules states that when taking shots on goal, the Andhill97 team often fails to score when (i) ball velocity is less than 2.37 meters per time step and (ii) the shot is aimed at greater than 6.7 meters from the center of goal (which is barely inside the goal). ISAAC reveals that shots governed by this rule fail to score 66 times without a successful attempt.

Now consider the perturbations of this rule. In cases where the rule is perturbed such that ball velocity is greater than 2.37 m/t and the shot aim is still greater than 6.7m, Andhill scores twice and fails to score 7 times. In another perturbation, where ball velocity is again less than 2.37 m/t but now shot aim is equal to or less than 6.7m (i.e. shots more towards the center of the goal), Andhill is now scoring 51 times and failing to score 96 times (See figure 4). These perturbations suggest that improving Andhill97's shot aiming capabilities can significantly improve performance, while trying to improve agents' shot velocity may not result in a drastic performance increase.

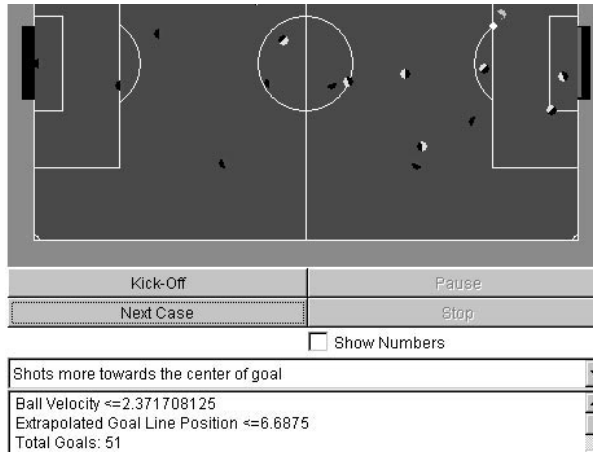

Fig. 4. Perturbation analysis showing Andhill '97 scoring using a perturbed shooting behavior

Perturbations of success rules are also useful. There are two reasons for this. First, it allows ranking of conditions contributing to success. In particular, some changes to the rule will take a team further from success than another. For example, a team may succeed in scoring 95% of the time when all conditions are met. The percentage of success may drop to 50% if the first condition is changed and down to 5% if the second condition is changed. In this case, the developer may decide that even if the first condition is not met, shooting is still the correct course of action while doing so if the second condition is not met is a bad decision. Secondly, allowing the user to see how the team succeeds or fails, more insight can be drawn as to why these conditions are important. Human oversight is important at this juncture to determine if the reasons ISAAC comes up with are truly the reasons the team is succeeding or failing.

4 Multiple Agent Key Interaction Model

4.1 Conceptual Overview of the Agent Interaction Model

To analyze agent interactions, ISAAC relies on matching predefined (possibly user-defined) patterns. These patterns consist of sequences of abstracted actions of different agents that result in intermediate successes (or failures). The patterns are matched against the data traces to find actual instantiated interactions. ISAAC then classifies the patterns into those that lead to intermediate success or failures. This

approach shares similarities with meta-pattern based data-analysis [13], where a user provides 'interesting templates' to discover patterns in the data.

The perturbation analysis in this model finds near misses in the case history, i.e., patterns similar to the successful patterns (within a given threshold) that end in failure. ISAAC identifies these near misses by perturbing the successful patterns, e.g., it may find interactions resulting in failure that result from a slight change in one of the actions specified in the pattern. Near misses help the user to scrutinize the difference between successful and unsuccessful patterns.

For example, in an air combat simulation, suppose that ISAAC finds a common pattern where friendly aircraft respond to the enemy's pincer maneuver with a flanking maneuver and a missile shot, causing an enemy aircraft to be shot down. The perturbation analysis will review traces to find instances where an identical pattern, with a slight variation in the missile shot (or one of the maneuvers), does not result in an enemy aircraft getting shot down. The user is then able to view and compare the successful and unsuccessful (yet similar) patterns to determine possible differences and causes for failure.

4.2 Application of Agent Interaction Model

The first step necessary to apply the agent interaction model to the RoboCup domain is the determination of the patterns to examine and a notion of success or failure of these patterns. We again use a soccer goal as a notion of success, so any pattern leading to a goal is successful. The interaction patterns are made up of the players' actions (kicks) causing the ball to be shot in to the goal.

To further illustrate this type of analysis, we present examples from the Windmill Wanderer and ISIS teams from RoboCup '98. The ISIS team scored 20 of their 35 goals when the player kicking the ball before the shot was on the opponent team (described by a *shooter→opponent→shooter* pattern). This suggests a very opportunistic scoring behavior for ISIS, and viewing the cases shows that ISIS tends to push the ball deep into the opponent territory, and sometimes they are able to get the ball back for a quick shot. In contrast, the Windmill Wanderer team scored 17 goals from the shooter dribbling in before the shot (*shooter→shooter→shooter*) and another 9 goals from a teammate controlling the ball before passing to the shooter (*teammate→teammate→shooter*), out of a total 37 goals. Thus, this team scores more often when they control the ball all the way in to the goal, a stark contrast from the ISIS team.

For the Windmill Wanderer team from above, 27 near misses were found similar to the 17 goals from the dribbling pattern, suggesting this pattern was well defended or the team was making some mistakes. Windmill Wanderer placed third in the tournament, and the 27 near misses may have been the culprit in its third place finish. The developer can review and compare these cases in the multimedia viewer, and make the determination as to what changes would be most beneficial.

5. Automated Game Summary: Team Model

5.1 Conceptual Overview of Team Model

The purpose of the global team model is to analyze why teams succeed or fail over the course of the entire engagement (as opposed to teams' intermediate success/failure at a single time point). The assumption of this model is that there can be many different factors that impact a team's overall success or failure. In a complex environment, a team may have to perform many actions well in order to be successful. These actions may involve entirely different sub-teams, and very different kinds of events, perhaps widely separated in time, may be implicated in success or failure. Nevertheless, there may be patterns of these factors that, although not strongly related in the behavioral trace, do in fact correlate with whether a team succeeds in a domain. The global team analysis attempts to find these patterns to explain success/failure.

In designing this model, we had two options possible. One was to tailor the analysis to specific teams. In particular, by analyzing data traces of past behaviors of a specific team, it would be possible to explain why this specific team tends to succeed or fail. This approach would be similar to the one followed in Section 3, which explained why agents' critical actions tend to succeed or fail in a team-specific manner. A second option was to analyze teams in terms of why teams succeed or fail in the domain in general, in a non-team-specific manner (which does not require data traces from the particular team being analyzed, but from other teams in this domain). Despite the advantages of option 1 (team-specific explanations), this option was rejected due to the lack of large amounts of team-specific engagement data, and option 2 was used. In particular, unlike the individual agent model as in Section 3, which can obtain lots of key event data points even from a single engagement, a single engagement is just one data-point for the global team model. For instance, even a single RoboCup game provides large numbers of shots on goal to begin learning the individual agent model; yet, this single game is not enough to begin learning a global team model.

Exercising option 2 above implies acquiring the team model by examining the behavior traces of many different teams in a domain. Here again we rely on the domain expert to provide the set of overall features that lead to success or failure over an entire engagement. Again the C5.0 induction algorithm is used on these features, classifying the engagement as a success or failure for each team, and learning rules that capture the dominant features that lead to success (or failure).

A different approach is taken for using the rules learned via C5.0. When analyzing a specific engagement, we mine the features from the engagement and determine which learned rule the current game most closely matches. This rule then becomes the reasoning for why each team succeeded or failed. ISAAC uses the rule as

the basis for its natural language summary generation to ease human understanding of the engagement as a whole.

The learned rules are critical in ISAAC's natural language summaries. Indeed, we initially attempted to provide a natural language summary without using such learned rules. In this case, we were only able to present all of the game statistics glossed with natural language phrases, with no ordering or reasoning behind them. This initial approach failed because all of the summaries were long, uniform, and they failed to emphasize the relevant aspects of the game.

Instead, with the current method, ISAAC generates a natural language summary of each encounter employing specific rules as the basis for content selection and sentence planning in accordance with Reiter's architecture of natural language generation [11]. Reiter's proposal of an emerging consensus architecture is widely accepted in the NL community. Reiter proposed that natural language generation systems use modules for content determination, sentence planning, and surface generation. ISAAC's NL generation can be easily explained in terms of these modules.

Starting with the raw data of the game, ISAAC mines the features it needs, and matches it to a pre-existing rule. This rule is thus used in content determination for the natural language generation, since the rule contains that which ISAAC believes pertinent to the result of the game. Furthermore, the conditions of each rule also have some ordering constraints, since the rules come from a decision tree learning algorithm, and we use this to form our sentence planning. We consider branches closer to the root of the tree to have more weight than lower branches, and as such should be stated first. Each fact is associated with a single sentence, and ordered accordingly. From these, ISAAC creates a text template of the summary for performing the surface generation. This template is augmented with specific data from the game, and links to examples of the features found earlier to be shown in the multimedia viewer.

5.2 Application of Team Model to RoboCup

To learn rules of why teams succeeded or failed in previous engagements, ISAAC reviews statistics of previous games. The domain expert must provide the domain knowledge of what statistics to collect, such as possession time and number of times called offside. ISAAC uses this information (10 features in all) to create a base of rules for use in analysis of future games.

ISAAC learns and uses seven classes of rules covering the concepts of big win (a victory by 5 goals or more), moderate win (a victory of 2-4 goals difference), close win (1 goal victory), tie, close loss (by 1 goal), moderate loss (2-4 goals), and big loss (5 or more goal loss). The motivation for such subdivision is that factors leading to a big win (e.g., causing a team to outscore the opponent by 10 goals) would appear to be different from ones leading to a close win (e.g., causing a one goal victory) and should be learned about separately. While this fine subdivision thus has some advantages, it also has a disadvantage, particularly when the outcome of the game is at the border of two of the concepts above. For instance a 2-0 game (moder-

ate win) could very well have been a 1-0 game (close win). Thus, we anticipate that the learned rules may not be very precise, and indeed as discussed below, we allow for a "close match" in rule usage.

To use these rules, ISAAC first matches the statistics of a new (yet to be analyzed) game with the learned rules. If there is a successful match, ISAAC checks the score of the game against that predicted by the matching rule before writing the summary. If the match is exact or close (e.g. the actual game statistic matched a close win rule, although the game had an outcome of 2-0), the template is used as is. If there are multiple matches, the closest matching rule is used. However, if no match is close to the actual score, ISAAC still uses the rule, but changes the template to reflect surprise that the score did not more closely match the rule.

The matched rule discussed above provides the content selection, so ISAAC now has a template to shape the game summary. The template orders components of the rule according to their depth in the original decision tree, in accordance with our sentence planning technique. ISAAC then fills in the template, mining the features of this particular game to create a summary based on the rule. An example rule is shown in Figure 5.

```
Ball in Opponent Half > 69%
Average Distance of Opponent Defender <= 15 m
Bypass Opponent Last Defender > 0
Possession time > 52%
Distance from Sideline of Opponent Kicks > 19 m
→ class Win Big
```

Fig. 5. Example team rule for big wins.

To see how this rule is used in creating a natural language summary, we examine one summary generated using this rule as a template. In this case, ISAAC is arguing the reasons for which 11Monkeys was able to defeat the HAARLEM team:

HAARLEM Offense Collapses in Stunning Defeat at the hands of 11Monkeys! [1]

11monkeys displayed their offensive and defensive prowess, shutting out their opponents 7-0. 11monkeys pressed the attack very hard against the HAARLEM defense, keeping the ball in their half of the field for 84% of the game and allowing ample scoring opportunities. HAARLEM pulled their defenders back to stop the onslaught, but to no avail. To that effect, 11monkeys was able to get past HAARLEM's last defender, creating 2 situations where only the goalie was left to defend the net. 11monkeys also handled the ball better, keeping control of the ball for 86% of the game. HAARLEM had a tendency to keep the ball towards the center of the field as well, which may have helped lead them to ruin given the ferocity of the 11monkeys attack.

The underlined sentences above correspond directly to the rule, with some augmentation by actual statistics from the game. By using the rule for content selection

[11] The title and first sentence of our summary does not come from the model, but is based solely on the score of the game. We used examples from (human soccer) World Cup headlines and let ISAAC randomly choose among these in categories of tie, close win, moderate win, and big win.

and sentence planning, ISAAC is able to present the user the reasons for the outcome of the engagement, and avoid presenting irrelevant data consisting of irrelevant features.

6. Evaluation and Results

To evaluate ISAAC, we evaluate each of its models in isolation and then the effectiveness of the integrated ISAAC system. We begin by evaluating the individual agent model.

A key measure of ISAAC's individual agent model is the effectiveness of the analysis, specifically the capability to discover novel patterns. Section 3.3 highlighted a rule learned about the Andhill97 team concerning their aiming behavior. This rule was one instance of ISAAC's surprising revelation to the human observers; in this case, the surprise was that Andhill97, *the 2nd place winner of '97*, had so many goal-shot failures, and that poor aim was at least a factor. Not only was this surprising to other observers, this was also surprising to the developer of the team, Tomohito Andou. After hearing of this result, and witnessing it through ISAAC's multimedia interface, he told us that he "was surprised that Andhill's goal shooting behavior was so poor…" and "… this result would help improve Andhill team in the future." [Andou, personal communication]

Another interesting result from the individual agent analysis model comes from the number of rules governing shooting behavior and defensive prowess. Figure 6 shows that in each year, the number of rules for defense decreased for the top 4 teams, perhaps indicating more refined defensive structures as the teams progress. Also, the number of rules necessary to capture the behavior of a team's offense is consistently more than that necessary for defense, possibly due to the fact that no single offensive rule could be effective against all opponent defenses. The key here is that such global analysis of team behaviors is now within reach with team analyst tools such as ISAAC.

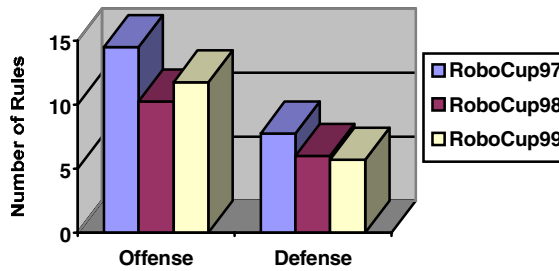

Fig. 6. Number of rules by year.

Another point of evaluation is understanding how well the model captures the shooting behaviors. To this end, ISAAC models were applied to predict game scores

at RoboCup '99, a rather difficult problem even for humans. ISAAC used rules describing a team's defense and matched them with the raw averaged data of the shots taken by the other team to produce an estimate of how many goals would be scored against that team in the upcoming game. Performing this analysis for both teams produced a predictive score for the outcome of the game[2]. This prediction obviously ignores many critical factors, including the fact that some early games were unrepresentative and that teams were changed by hand during the competition. Yet in practice, ISAAC's predictive accuracy was 70% with respect to wins and losses, indicating it had managed to capture the teams' defenses quite well in its model.

To evaluate the game summary model, a small survey was distributed to twenty of the participants at the RoboCup '99 tournament, who were witnessing game summaries just after watching the games. Figure 7 shows the breakdown of the survey, showing that 75% of the participants thought the summaries were very good.

Another measure of game summaries is a comparison of number of features used in the current summaries versus those generated earlier that did not use ISAAC's approach. On average, ISAAC uses only about 4 features from its set of 10 statistics in the summaries, resulting in a 60% reduction from a natural language generator not based on ISAAC's machine learning based analysis. Thus, ISAAC's approach was highly selective in terms of content. Indeed as mentioned earlier, summaries generated without ISAAC were much longer, lacked variety, and failed to emphasize the key aspects of the game.

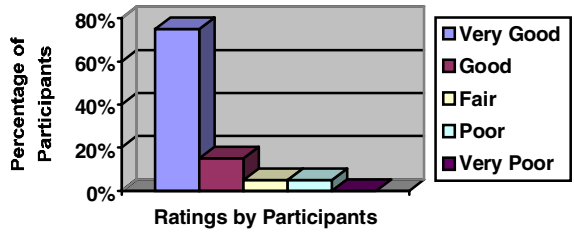

Fig. 7. Automated game summary survey.

Yet another measure of ISAAC's use of the team model for natural language generation is available by viewing the error rates from the machine learning algorithm used. These error rates tell us how accurately ISAAC's learned rules reflected the game. On the original set of games for which ISAAC's rules were learned, 87% of

[2] No prediction was done for the preliminary rounds as ISAAC gathered data on the teams. Prediction was performed only on the double-elimination tournament.

the games were classified correctly (70% exact match, 17% close match), resulting in an error rate of 13%. Our test set of (unseen) RoboCup '99 games produced 72% classified correctly (39% exact match, 33% close match), for an error rate of 28%. If an error does occur, ISAAC still produces a summary, but it reflects its surprise at the outcome, thus explaining the error. The high error rate on our training data could indicate that a better feature set is possible or that the data may be noisy.

Evaluating ISAAC as an integrated system is more difficult. However, some observations can still be made. ISAAC was awarded the 'Scientific Challenge Award' by the RoboCup committee. ISAAC was used extensively at the RoboCup '99 tournament in Stockholm, and received a great deal of praise and other feedback. Developers used ISAAC to analyze opponent teams after the early round matches to get a feel for the skill of upcoming opponents. Spectators and developers alike were able to view ISAAC's game summaries just minutes after a game, and there was also a great deal of speculation concerning ISAAC's predictions on future games.

7. Related Work

The research presented in this paper concerns areas of multi-agent team analysis and comprehensible presentation techniques. We compare research in each of these areas.

André et al have developed an automatic commentator system for RoboCup games, called ROCCO, to generate TV-style live reports for matches of the simulator league [1]. ROCCO attempts to recognize events occurring in the domain in real time, and generates corresponding speech output. While both ROCCO and ISAAC use multimedia presentations, ROCCO attempts to analyze events quickly to produce live reports. However, the ROCCO analysis does not use multiple models of behavior for multi-perspective analysis as in ISAAC, and its analysis is not designed to help users and developers understand teams' abilities. ROCCO also has no capability to perform perturbation analysis.

Bhandari et al's Advanced Scout uses data mining techniques on NBA basketball games to help coaches find interesting patterns in their players and opponents' behaviors [3]. Advanced scout also enables coaches to review the relevant footage of the games. Advanced Scout is able to capture statistical anomalies of which coaches can take advantage. However, Advanced Scout does not have some of ISAAC's extensions including the use of multiple models to analyze different aspects of teams, perturbations to make recommendations, and game summaries for an analysis of overall team performance.

Ndumu et al's system for visualization and debugging multi-agent systems comprises a suite of tools, with each tool providing a different perspective of the application being visualized [9]. However, the tools do not perform any in-depth analysis on the multi-agent system, and the system has no capability for perturbing this analysis. ISAAC also uses a visualization component, but only as an aid to understanding its analysis.

Johnson's Debrief system enables agents to explain and justify their actions [7]. This work focuses on agents' understanding the rationales for the decisions they make and being able to recall the situation. Debrief also has a capability for agent experimentation to determine what alternatives might have been chosen had the situation been slightly different. ISAAC performs something similar in its perturbation analysis; however, ISAAC focuses on an entire team, not just an individual, necessarily.

Stone and Veloso have also used a decision tree to control some aspects of agents throughout an entire game, also using RoboCup as their domain [14]. However, this work pertains to execution of agents rather than analysis of agent teams, and since it is internal to the agent, their work has no means of presentation.

8. Conclusion

Multi-agent teamwork is a critical capability in a large number of applications including training, education, entertainment, design, and robotics. The complex interactions of agents in a team with their teammates as well as with other agents make it extremely difficult for human developers to understand and analyze agent-team behavior. It is thus increasingly critical to build automated assistants to aid human developers in analyzing agent team behaviors. However, the problem of automated team analysts is largely unaddressed in previous work.

We have taken a step towards these automated analysts by building an agent called ISAAC for post-hoc, off-line agent-team analysis. ISAAC uses two key novel ideas in its analysis. First, ISAAC uses multiple models of team behavior to analyze different granularities of agent actions, using inductive learning techniques, enabling the analysis of differing aspects of team behaviors. Second, ISAAC supports perturbations of models, enabling users to engage in "what-if" reasoning about the agents and providing suggestions to agents that are already be within the agent skill set. Additionally, ISAAC focuses on presentation to the user, combining multiple presentation techniques to aid humans in understanding the analysis, where presentation techniques are tailored to the model at hand.

While ISAAC is intended for application in a variety of agent team domains, ISAAC has currently been applied in the context of the RoboCup soccer simulation. It is available on the web for remote use. ISAAC has found surprising results from top teams of previous tournaments and was used extensively at the RoboCup '99 tournament. ISAAC was awarded the 'Scientific Challenge Award' at RoboCup '99 where its analysis and natural language game summaries drew a crowd throughout the tournament.

Acknowledgement

We thank Intel Corp for their generous support of the work reported in this article.

References

1. André, E., Herzog, G., Rist, T. Generating Multimedia Presentations for RoboCup SoccerGames. In *RoboCup-97: Robot Soccer World Cup I,* 1997.
2. Balch, T. The Impact of Diversity on Performance in Multi-robot Foraging. In *Proceedings of the Third Annual Conference on Autonomous Agents*, 1999.
3. Bhandari, I., Colet, E., Parker, J., Pines, Z., Pratap, R., Ramanujam, K. Advanced Scout: Data Mining and Knowledge Discovery in NBA Data. *In Data Mining and Knowledge Discovery*, 1997.
4. Caruana, R., Freitag, D. Greedy Attribute Selection. In *11th Proceedings of the 11th International conference on Machine Learning (ICML)*, 1994.
5. Dorais, G., Bonasso, R., Kortenkamp, D., Pell, B., Schreckenghost, D. Adjustable Autonomy for Human-Centered Autonomous Systems. *Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems*, 1999
6. Jennings, N. Controlling Cooperative Problem Solving in Industrial Multi-agent System Using Joint Intentions. *In Artificial Intelligence, Vol. 75,* 1995.
7. Johnson, W. L. Agents that Learn to Explain Themselves. In *Proceedings of AAAI-94*, 1994.
8. Kitano, H., Tambe, M., Stone, P., Veloso, M., Noda, I., Osawa, E. & Asada, M. The RoboCup synthetic agent's challenge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI),* 1997.
9. Ndumu, D., Nwana, H., Lee, L., Haynes, H. Visualization and debugging of distributed multi-agent systems. In *Applied Artificial Intelligence Journal, Vol 13 (1),* 1999.
10. Quinlan, J. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1994.
11. Reiter, E. Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation*, 1994.
12. Sengers, P. Designing Comprehensible Agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI),* 1999.
13. Shen, W.M., and Leng, B. A Metapattern-Based Automated Discovery Loop for Integrated Data Mining. In *IEEE Transactions on Data and Knowledge Engineering*, 1996.
14. Stone, P., Veloso, M. Using Decision Tree Confidence Factors for Multiagent Control. In *Proceedings of the International Conference on Autonomous Agents,* 1998.
15. Sycara, K., Decker, K., Pannu, A., Williamson, M., Zeng, D., Distributed Intelligent Agents. In *IEEE Expert*, 1996.
16. Tambe, M. Johnson, W. L., Jones, R., Koss, F., Laird, J. E., Rosenbloom, P.S., Schwamb, K. Intelligent Agents for Interactive Simulation Environments. In *AI Magazine, 16(1) (Spring)*, 1995.
17. Tambe, M. Towards Flexible Teamwork. In *Journal of Artificial Intelligence Research, Vol. 7,* 1997.
18. Ting, K. Inducing Cost-Sensitive Trees via Instance Weighting. In *Principles of Data Mining and Knowledge Discovery (PKDD 98),* 1998.