Using Active Probing by a Game Management AI to Faster Classify Players

Arkady Eidelberg
Computer Science Department
University of Calgary
Calgary, Canada
aeidelbe@ucalgary.ca

Christian Jacob
Computer Science Department
University of Calgary
Calgary, Canada
cjacob@ucalgary.ca

Jörg Denzinger
Computer Science Department
University of Calgary
Calgary, Canada
denzinge@cpsc.ucalgary.ca

***Abstract*—In this paper, we present the use of a so-called Game Management AI to classify players not just by passively observing them, but by actively manipulating the game to get the players to provide data currently missing to achieve the classification. We call this "Active Probing". The Game Management AI uses two sets of rules, one set that contains rules that are intended to represent the knowledge allowing a classification and one set that contains rules that indicate which game events can contribute to triggering conditions used in the first rule set. When a rule of the first set comes near to being triggered, the event suggested by an appropriate rule in the second set is then offered to the player in the game.**

We instantiated this use of a Game Management AI to identify players with a very high interest level for the role playing game "Realm of Dreams", a game that we created for this purpose. Our experimental evaluation showed that using the active probing by the Game Management AI allows us to identify players in our targeted class in a quarter of the time that was needed to classify such players without active probing.

***Index Terms*—Game analytics, Player modelling, Active probing, Game Management AI**

I. Introduction

It is essential for game producers to get an idea how players play and enjoy a game, which includes, for example, how they use (or not use) particular features of the game. In addition to traditional methods like test players and focus groups, the use of game analytics (see [9]) has become very important for this. Being able to identify different types of players and types of play (see [13], [15]) allows to also use these classifications to identify the appropriate types for a new player (or at least many of them), which then offers various possibilities. For example, a game can monitor how much a player is enjoying the game and interfere before the player loses interest. Also, marketing teams can use player analytics systems to optimize their monetization techniques. For example, in games where the main monetization vector are in-app purchases, optimizations could lead to players purchasing more in-game items. Alternatively, if the monetization follows a model where players pay a subscription fee, these optimizations can judge how likely a player is to burn out and interfere to get the player to continue the subscription. On top of that, the popular massive online games usually tend to introduce updates every once in a while in order to introduce new content and to balance existing content in order to keep the players happy. This process is accompanied by increasingly sophisticated tools that simulate game playing and collect logs of activity from existing players [1] to make sure that such updates do not flop. There are already proposals for systems that employ AI methods like Machine Learning within games to identify behavioral patterns in players [4]. This includes using datasets of long-term game-play to customize the player experience for each person or a group of players [8], [10], [13].

So far, all these approaches are assuming that analytics is a passive process, just observing the player playing the game, even in approaches that aim at adjusting the game after a classification has happened. While this follows generally accepted data collection processes in many areas, it is an inherently slow process, as usually games have a lot of choices in regards to what the players can do, and players choose to do these actions in a very indirect manner. If an observation method requires to observe the player in a given situation in order to classify him or her, so far, nothing can be done but wait until the player is put in such a situation by chance.

In contrast to this current practice, we propose to speed-up the classification process by having a so-called Game Management AI (GMAI) perform *active probing*. This means that the GMAI tries to introduce players to key situations as soon as possible, reducing the element of chance. It also gets the player into situations which have a high chance to reveal information that makes a particular classification more likely.

More precisely, our GMAI collects data about the player and uses two rule sets to help identify when a player's behavior puts him or her into a known player class. The first rule set, the *class recognition rules*, inspects the collected data about a player and triggered rules raise the score for a particular class of players. When such a score is above a given threshold, the player is considered to belong to this class. In order to realize the active probing, a second set of rules, the *threshold probing rules*, is used to inspect the class recognition rules and look for rules that are nearly triggered. Once such a rule is identified, the set of threshold probing rules is checked for rules that indicate to the GMAI that a specific game scenario might bring this class recognition rule nearer to triggering and the GMAI will then present this scenario to the player.

We evaluated this concept within an on-line Role Playing

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

Game (RPG) called "Realm of Dreams" to identify players that are highly interested in the game. This game was created specifically for this research, using the Unity3D Game Engine [12] and a centralized web server component that collected player analytics and instantiated our GMAI system. Using this system, we identified 6 players that exhibited higher interest behavior patterns out of our 37 participants. Out of these 6 players, 4 were in the group that had the GMAI system which used the active probing technique. These 4 players were identified to have a high level of interest towards the game in only a quarter of the time needed to identify the other 2 players who were only passively observed.

This paper is organized as follows: after this introduction, in Section II we present how a GMAI can be used for player classification. In Section III, we present the instantiation to Realm of Dreams and in Section IV we describe our evaluation. This is followed by some more related work in Section V and concluding remarks in Section VI.

II. Using a GMAI for player classification

The idea of a GMAI is to customize the whole gaming experience for a particular player using general information about game playing, about the particular game, the general playing behavior of players of the game and the playing behavior of the particular player. The possibilities for customization range from game world building over just building scenarios to the creation (and triggering) of in-game events, but also including events outside of the game world, like sales promotions. A GMAI can be used to customize the game to provide more fun to the player (using different definitions of fun), to get players that do not "fit" the player community to give up on the game, other purposes and, naturally, combinations of purposes.

The particular customization task we are concerned with in this paper is the active detection of players in a class that exhibit high levels of interest in the game. For this task, we use two sets of rules (created by the game designer). The first set of rules, the class recognition rule set $ClassRec$, consists of rules of the form $cond_1(P) \wedge ... \wedge cond_k(P) \rightarrow AddToScore(y, P)$, where P is the player information about a player (who we will also call P). Each $cond_i$ has the form $feature_i(P) > threshold_i$, evaluating the current value of $feature_i$ for player P. $AddToScore(y, P)$ is the action of adding y points to player P's class score $ClassScore(P)$, which is part of P's information and represents our only class of interest for our proof of concept system. The second set of rules, the threshold probing set $ThreshProb$, has rules of the form $feature_i(P) \rightarrow CreateEvent(P)$, where $CreateEvent$ is a particular action for the GMAI to have player P encounter a particular event (or events) in the game as soon as possible.

For each player P, the GMAI collects at least the information about all features occurring in rules in $ClassRec$ during game play whenever the player does something that leads to a change of the feature value. Additionally, the GMAI is integrated into the standard control loop of the game in the following manner: After each meaningful interaction with the game by player P, the player information, including the interaction counter, is updated. If the GMAI wants then to evaluate the rules in $ClassRec$ (which can be determined by either using a timer or a number of interactions, but should not be done after every interaction), it does so, updating the $ClassScore$ value according to every newly triggered rule in $ClassRec$. If the $ClassScore$ has reached the threshold that is considered as indicating that the player should be a member of the class, the GMAI performs whatever actions the game designers want to be done in that case. Otherwise, if the GMAI thinks that it should do active probing, it determines the feature $feat$ it should do probing on and selects among all the rules in $ThreshProb$ that have $feat$ as condition the rule $feat \rightarrow CreateEvent(P)$ that it wants to use. It then initiates $CreateEvent(P)$ and finishes the control loop. If there is no followup to a previously initiated event from $ThreshProb$ and the GMAI does not want to evaluate the rules in $ClassRec$ or does not want to do active probing, the game uses the "normal" control loop given by the game designers.

There are several possibilities how the GMAI can determine if it wants to do active probing, depending on how much game designers allow their normal game control loop to be influenced. The possibilities range from using evaluations of how near rules in $ClassRec$ are from being triggered to just simple counters. Also for determining what feature $feat$ to do active probing on there are different general possibilities. One possibility, that we used in our system, is to have for each feature $feature_i$ of a rule additionally a weight w_i, which is intended to be the importance of the feature for the rule. All the weights for a rule add up to 1. We then divide the current $feature_i(P)$-value by $threshold_i$ and multiply it by w_i, which gives us this feature's weighted completion percentage. Adding up these percentages for all features gives us the completion percentage for a particular rule. Then we can determine the rule in $ClassRec$ (that has not been triggered, yet) with the highest completion percentage (which means that it should be nearest to being triggered) and select the feature within that rule that has the lowest weighted completion percentage as $feat$. Finally, the selection of the rule from $ThreshProb$ can be done in several ways. One way would be to allow for each possible feature only one rule with that feature as condition. Another way is to choose one possible rules randomly or assigning weights to the rules and select the rule with highest weight that has not been selected before.

The combination of possibilities for the various decisions the GMAI has to make is chosen based on a particular game and the available knowledge for creating rules. We suggest using rules and the presented modification of the control loop to allow for easily changing the rules in both sets, to accommodate new knowledge or new possibilities for $CreateEvent$ (in new game versions, for example).

III. Instantiation to Realm of Dreams

In this section, we will first describe the game "Realm of Dreams" we created to allow us to evaluate our active probing concept for classifying players and the various data we collect in the game about a particular player. Then we will present

the two rule sets $ClassRec$ and $ThreshProb$ we created for this game and why we think that these rule sets allow for fast classification of the level of interest a player will exhibit through that player's life-cycle in the game. Finally, we present our choices from the possibilities for the various decisions in the game control loop for a player.

A. Realm of Dreams

In Realm of Dreams (available at: www.realmofdreams.ca), the player controls an avatar in a virtual world. It is a multi-player game that mimics the popular style of Multi-User Dungeon (MUD) games from the 80's and 90's and takes inspiration from more modern Role Playing Games (RPG) such as World of Warcraft, Tibia and Ultima Online. This avatar travels a virtual world and has the ability to interact with non-player characters and monsters. There are different items that the player can use, to both increase the damage it deals to monsters and reduce damage taken from them. There are healing poitions and other supporting materials as well. For our purposes, a key component of the game is our *FakeAd* system, which tries to emulate video games where the player gets an extra benefit from watching a short commercial advertisement. This system rewards the player with a special in-game currency, *souls*, that can be used to give the player an advantage in the game (or not). This includes better healing potions, the removal of some death penalties and the ability to instantly travel to a specific location on the map. However, in order to collect these souls, the player has to watch a timer of different lengths (30 to 90 seconds of count down), which is a very boring and routine task. We assume that time spent watching the timer, along with some other key statistics, are a good indicator of a player's level of interest in the game. This assumption is often validated in industry and the key features of the player's interest can be adjusted easily in our system.

The following are the meaningful interactions between the player and the game, which is stored in the player information:

- player movement on a single map,
- player killing a monster,
- player interacting with Non-Player Characters (NPC),
- player sending a message in chat,
- player finishing a quest,
- player watching a FakeAd,
- the completion of a few special quests by the player,
- usage of items by the player and if it was successful or not (certain items have a failure chance - if the item usage fails, the player depletes the item but does not gain the beneficial effect),
- movement of players between maps (large areas of the game world),
- the amount of in-game gold and experience a player has gathered and
- player deaths.

The number of occurrences of these interactions or the collected amounts (both for the whole game and for a session) are part of the features. Additionally, we keep track of the playing time for each session of play by the player in P.

There are two special quests that a player can complete during game play. Both can be given to the player by the normal control loop of the game or by our instantiation of the GMAI. The first special quest asks the player to kill a slightly more difficult "boss" monster. Doing so awards the player with a custom look for the player's avatar - more specifically, the avatar wears a magic hat on top of the regular appearance, setting this player apart from the rest of the participants. This effect lasts only for the current session (i.e. until the player logs out of the game). The second special quest involves the player killing a specific type of creatures. The type of the creature is determined based on the player's advancement through the game. Once the player has slain the requested amount of the correct creatures, the player is rewarded with the ability to change the game's welcoming message - the first message that is automatically sent by the server when a player logs into the game.

B. Our rules for $ClassRec$

There is a lot of literature on why people play games. Our review of previous works helped us identify several main indicators for when a player is very interested in a game. Some of these are not only based on the behavior of a player in the game [5], [6], [16]. Moreover these indicators can be used for player behavior prediction [13], [14].

For our interest class recognition rules we were inspired by 6 such indicators, namely: immersion, social involvement, "busy work", repeat failure, the amount of time the player spent in the game and the player's in-game achievements (by gold, experience and exploration). These are the guiding principles on which we will instantiate our $ClassRec$ rules. For every class recognition rule that deals with global averages, the running average of all players were updated and calculated with each player's interaction that affected the relevant average. For example, when a player sent a message, that player's message count would increase and the game server would update the average amount of messages sent by all players. This would be in effect for any future calculations of the relevant rules.

More specifically, we used 500 points as threshold for $ClassScore$. Since we are not covering all known indicators and our rules are only inspired by some indicators, it is important to note that we do not claim that the current configuration of rules and parameters is ideal. Our system only identifies players exhibiting behavior patterns that according to existing literature correspond to those of players with high interest in the game. This information could be used by the gaming industry to both help prevent abusive behavior in players who might be showing signs of self-destruction or could be used for marketing purposes and future game design. In the following, we present the rules in $ClassRec$ grouped by the 6 indicators mentioned above.

1) Immersion: Players who seek to escape the real world and get immersed in a virtual adventure are a natural fit for highly interested players. Such Escapism was shown to be a very good indicator of the player's involvement

in the game [16]. The idea is that when players begin to associate their avatar in a virtual world with themselves, they are highly engaged with the game and are in the group of players that we seek to identify. Our rule that corresponds to immersion uses the fact that the player has done the special hat quest, indicating that there is a strong link between player and avatar, and the amount of time the player was logged in for the current session compared to the player's average session length. Formally, the rule is: $CurrentSessionTime(P) > AvgSessionTime(P) \wedge HasHat(P) > 0 \rightarrow AddToScore(300, P)$. Note that 300 is a very large addition to the interest score as this is a major indicator according to [16].

2) Social Involvement: Social relationships are a major part of gaming. In fact, it has been shown that persons with poor social relationships in real life tend to gravitate to social interactions in a virtual setting and that these players have a higher attachment to their virtual avatars [3], [17]. Therefore, for this indicator, we used the second special quest and checked if the player has changed the game's welcome message or not. This was measured together with the total number of messages the player has sent and compared to the player's average number per session. Formally, the rule has the following form: $PlayerSessionMsgs(P) > AvgMsgsPerSession(P) \wedge HasChangedWelcome(P) > 0 \rightarrow AddToScore(300, P)$. The first part checks if the player has sent more messages to the chat than he does on an average session and the second checks if the player has completed the second special quest and changed the server's welcome message. As this rule checks for the player's main forms of social interaction, this is another strong indicator and upon successful triggering of this rule, 300 points were added to the player's interest score.

3) "Busy Work": A large part of what sets highly involved players apart from more casual players is their dependency on positive reinforcement from the game, even when there is not much of a challenge to the player [6]. In our game, players receive positive reinforcement when they kill a monster and improve their character by gaining experience. We track the number of creatures killed in a session as well as the overall difficulty of all creatures killed. To this end, each time a creature is killed a difficulty score is added to the player's information using the formula $difficultyToAdd = CreatureDifficulty - (playerLevel/3)$. The creature difficulty was hard coded into the game for each creature according to the perceived difficulty. As the gap between the player level and the creature difficulty increased, the difficulty component of this rule becomes fulfilled faster, based on the concept of easy challenges and low reward being common in highly involved players [17]. As such, the full form of the rule is $CreaturesKilledThisSession(P) > 40 \wedge CreatureDifficulty(P) > 40 \rightarrow AddToScore(200, P)$. The first part of the rule checks that the player has killed more than the hard-coded threshold for the number of creatures needed to be killed (40) and the second part of the rule checks that the total difficulty exceeded the threshold (40) as well. These thresholds were the same number by coincidence, but they don't have to be. As a medium indicator, when this rule is triggered, we increase that player's interest score by 200 points.

4) Repeat High Risk Failure: In a similar way to "busy work", a player that loses the game multiple times in a row can indicate a high level of interest. Such players are sucked into the game and seek the excitement of a very hard challenge, one that they know they are unlikely to overcome. While showing that the player is not very skillful at the game, such behavior in the game could indicate that the player is very engaged with the game and enjoys taking high-stake risks within the game's environment [16]. By tracking the player deaths, we were looking for sessions that are close to each other in time (at least an hour from each other) in which the player died 3 times in a row. As such, if the player managed to fail 3 times but was not deterred from continuing the game, we considered the player highly engaged. The rule has only one feature and is of the form $PlayersDeathInARow(P) > 2 \rightarrow AddToScore(100, P)$. This is considered a weak indicator and increases the player interest score by 100 points.

5) Longer Sessions: One of the features that highly engaged players exhibit is an excessive amount of game time. This is one of the ways by which such players can be distinguished from other more casual players, according to [18]. For this rule, we compare the player session's length to the global session average of all other players to determine if the player has spent more time playing the game than most other players. As such, the rule has the form $PlayerSessionTime(P) > GlobalAvgSessionTime() \rightarrow AddToScore(100, P)$. When this rule is triggered, it is considered a weak indicator and the player interest score is incremented by 100 points.

6) In-Game Achievements: Finally, the attempts to become very successful at the game is another potential indicator of interest [2], [16]. There are 3 rules that measure the player's in-game success. As such, we track three per session pieces of information - how much in-game gold the player has accumulated, how much experience the player has accumulated by killing monsters and through how many maps the player has traveled [2].

The first rule compared in-game gold achievements and is of the form $PlayerSessionGold(P) > GlobalAvgSessionGold() \rightarrow AddToScore(100, P)$. In a similar fashion, the second rule is of the form $PlayerSessionExp(P) > GlobalAvgSessionExp() \rightarrow AddToScore(100, P)$, for the player's experience gain. Finally, the last rule measures how much exploration a player has achieved in a single session and has the form $PlayerMapVisitedSession(P) > 4 \rightarrow AddToScore(100, P)$, where the threshold of 4 was selected as most dungeon sets (a reasonable amount of exploration) were 2 or 3 maps. Each of these rules we consider a weak indicator and add 100 points to the interest score.

A summary of all the rule motivation, rule components and their relative score can be seen in Table I.

TABLE I
SUMMARY TABLE OF THE $ClassRec$ RULES, INCLUDING THE RULE MOTIVATION AND THE SCORE TO ADD TO THE INTEREST SCORE OF A PLAYER.

Rule Motivation	Rule Components	Score
Immersion	CurrentSessionTime(P), AvgSessionTime(P), HasHat(P)	300
Social Involvement	PlayerSessionMsgs(P), AvgMessagesPerSession(P), HasChangedWelcome(P)	300
Busy Work	CreaturesKilledThisSession(P), AverageDifficulty(P)	200
Repeated High Risk Failure	PlayersDeathInARow(P)	100
Longer Sessions	PlayerSessionTime(P), Global-AvgSessionTime()	100
In-Game Achievements (Gold)	PlayerSessionGold(P), Global-AvgSessionGold()	100
In-Game Achievements (Experience)	PlayerSessionExp(P), Global-AvgSessionExp()	100
In-Game Achievements (Exploration)	PlayerMapVisitedSession(P)	100

C. Our rules for $ThreshProb$

For each feature in the interest recognition rules discussed in the previous section, a game event had to be configured. As mentioned before, the class recognition rule which is closest to activation is selected for probing and the feature that is the most distant from completion is then the one that dictates which game event is created.

As the scope of our game was limited, some of the game events were re-used for different threshold probing rules where they were appropriate. Other game events can be written for every rule and more work needs to be done to identify the parameters for the best game events - we have used the reasoning behind each interest probing indicator to define some that are sufficiently close. Note that while we created only one rule for each feature there could be more rules for each feature. In addition to having the two special quests, we will use quests that are part of the normal game control loop as events as often as possible, since this limits the additional effort needed for active probing (and we expect that this can be done for many games). This also feels natural for many players, who should be somewhat familiar with quests as dynamic missions, based on their experiences with other modern games. This is important, as the fact that a player realizes that he or she are being probed could influence the effectiveness of the system.

The rule $CurrentSessionTime(P) \rightarrow GiveTimeQ(P)$ indicates that the player has not spent enough time in the game to make this feature condition true in the recognition rules and the GMAI should trigger a game event that offers a time consuming quest to the player. The dynamically created event $GiveTimeQ(P)$ encourages the player to stay in the game by killing a set amount of creatures on the map that the player is currently located on.

$HasHat(P) \rightarrow GiveHatQ(P)$ is the obvious rule around the hat, triggering the hat quest if the player has not gotten the hat, yet.

Next, $PlayerSessionMsgs(P) \rightarrow TalkNPCQ(P)$ gets the game to offer the player to conduct a dialog with NPCs in the game. This dialog is story driven, an event that is already part of the game and increases the number of messages sent to chat if the player decides to act on it.

Another rule is $HasChangedWelcome(P) \rightarrow GiveWelcomeQ(P)$. This indicates that the player has not completed the special quest that allows him or her to change the welcome message. Therefore the $GiveWelcomeQ(P)$ event triggers the game to offer this special quest.

The rule $CreaturesKilledThisSession(P) \rightarrow CreaturesKillQ(P)$, triggers a game event that offers a quest for the player to kill more creatures, optimized on quantity. $CreatureDifficulty(P) \rightarrow CreaturesDiffQ(P)$ triggers a similar event, except this time it is targeting easier creatures compared to the player level to minimize difficulty (and therefore maximize increase in $CreatureDifficulty(P)$).

Another rule is $PlayersDeathInARow(P) \rightarrow HardChallenge(P)$. This in-game event encourages the player to explore areas that are considered much too difficult for the player by offering such a quest (and providing the means to enable the player to get to this fast), increasing the likelihood of the player dying, again.

The $PlayerSessionTime(P) \rightarrow GiveTimeQ(P)$ reuses the $GiveTimeQ$ event to increase the time the player spends in the current session (which is unsurprising, given that the features this rule and the first rule above are targeting are rather similar). Similarly, the $PlayerSessionGold(P) \rightarrow GiveGoldQ(P)$ rule uses a game event that sends the player to kill more creatures or a boss monster - this presents the player with a quest that results in more gold collected. The $PlayerSessionExp(P) \rightarrow GiveExpQ(P)$ rule uses an event meant to increase the player's gained experience (so the monster selection is optimized around maximizing experience gained). Finally, with the $PlayerMapVisitedSession(P) \rightarrow GiveExploreQ(P)$ rule, the game tasks the player with a quest that will take the player to a new area in the game to which the player has not traveled yet.

A summary of all the rule components and the events the GMAI invokes can be seen in Table II.

D. The instantiated GMAI control loop

We have presented a number of options in regard to how the GMAI can be involved in the control loop of the game in a previous section. For our instantiation, we introduce a minimal activation period - a player has to be in the game for at least that long before the GMAI checks the $ClassRec$ rule set. This is introduced to give the system some minor amount of time to decide which rule is actually likely to fire and prevent rules from being fired based on the first observable action (which could be just a coincidence, at the very beginning). As such, not all players are probed by the active learner at the same time. This random amount of time was between 2 and 5 minutes in our experiments. While the exact overhead of our GMAI system was not measured, it was negligible and

TABLE II
SUMMARY TABLE OF THE $TreshProb$ RULES AND WHAT GAME EVENTS ARE INVOKED BY THE GMAI FOR EACH RULE COMPONENT.

Lowest Rule Components	Game Event
CurrentSessionTime(P)	GiveTimeQ(P), offer the player a time consuming quest.
HasHat(P)	GiveTimeQ(P), GivaHatQ(P), offer the hat quest to the player
PlayerSessionMsgs(P)	TalkNPCQ(P) offer the player to talk to an NPC for more messages
HasChangedWelcome(P)	GiveWelcomeQ(P) offer the welcome message quest to the player
CreaturesKilledThisSession(P)	CreaturesKillQ(P) offer the player a quest to kill more easy monsters
CreatureDifficulty(P)	CreaturesDiffQ(P) offer the player a quest to kill more difficult monsters
PlayersDeathInARow(P)	HardChallenge(P) offer the player a very difficult quest
PlayerSessionTime(P)	GiveTimeQ(P) offer the player a very time consuming quest
PlayerSessionGold(P)	GiveGoldQ(P) offer the player a lucrative quest in terms of gold reward
PlayerSessionExp(P)	GiveExpQ(P) offer the player a lucrative quest in terms of experience reward
PlayerMapVisitedSession(P)	GiveExploreQ(P) offer the player a quest that will take the player to a new area of the game world

allowed the game to continue without noticeable effect for the player or major resource consumption from the game server.

Other than this modification, the GMAI activates every 15 seconds on the general game loop and evaluates the $ClassRec$ rule set for every player that is still on-line. This amount of time was a general balance point between observing these rules often enough and not taxing the game server with too frequent updates, but this will vary for other games. A player specific evaluation also occurs as part of the log-in sequence and the log-out sequence of the game control loop.

The evaluation of the $ThreshProb$ rule set (the active probing) happens right after the evaluation of $ClassRec$. Based on the latest update of $ClassRec$, a feature is selected and a potential game event is executed. There are a few exceptions to this, however. First, the system checks if there is an already active game event. An active game event can expire - in which case it is canceled (with the consequences communicated to the player) and it is added to a special list which prevents this event from being activated again during the current game session. If there is a game event that is currently active and it has not expired, the active probing stops here and awaits the completion or the expiration of the event. If there were no active events, a feature is selected as is described in the general method, unless they have unsuccessfully been probed prior to this, in which case the next best feature is selected. For our rules, all features had the same w_i, so all rules with a single feature had the weight as 1 and the rules with 2 features had the weight as 0.5 for both features. Finally, if the evaluation was the same for 2 rules or more, the tie was broken randomly.

IV. EVALUATION

To evaluate our GMAI for player classification based on player interest we recruited players via an online recruiting site (www.realmofdreams.ca), which we published on online gaming forums and in Facebook ads that targeted both genders and an age group of 18 to 65+ located in the major English speaking countries who showed interest in computer games. Each player creating an account for our game was randomly put into one of two groups: the "active" group that played the game using our GMAI as described in the last section and the "passive" group that played a game variant where the active probing via the rule set $ThreshProb$ was disabled. The data we discuss in the following was collected over 31 hours of collective game play from all the players. There were 37 active players in this experiment. A player was considered active if he or she created an account and spent at least 10 minutes playing the game. There were 20 players in the passive group and 17 players in the active group.

The first data of interest is naturally if there were players identified as having a high level of interest and when that identification point happened. As we hoped, the active probing by the GMAI was able to expedite the identification of highly engaged players, and therefore we had more highly engaged players in the active group than in the passive group for the same amount of testing time. We have identified a total of 6 such players, 4 from the active group and 2 from the passive group, as shown in Table III. It is interesting to note that these 6 players are responsible for 15 hours of game play – almost half of the total amount played.

In this section, players are identified just by an ID number (for privacy reasons) and the group they were in is identified by P for the passive group or by A for the active group. In Table III we compare all 6 highly engaged players and how long (in-game seconds) it took for them to reach the identification threshold we defined (column "Identification Time").

TABLE III
TIME STATISTICS (IN SECONDS) ON HIGHLY ENGAGED PLAYERS.

Player	First Probe	Identification Time	Time Played
A1	190	3268	15427
A2	242	2342	5065
A3	212	1074	3671
A4	162	2258	6182
P1	315	10695	13896
P2	205	8212	8579

As Table III clearly show, players from the active group exceeded the identification threshold much faster. In fact, the average time it took to identify players from this group was 2,235.5 seconds (37.25 minutes), compared to 9,452.5 seconds (157.5 minutes) for the passive group. This means that the GMAI with active probing needed only a quarter of the time than the game version without active probing!

The other columns of Table III provide a closer look at the 6 players identified. Since active probing begins only some

random playing time after a player first logs into the game, the table lists under the column First Probe when active probing became enabled for a player, respectively when the GMAI first performed a probe in the game. For passive players (who were not actively probed), we kept a log of when such a probe would have happened, had they been in the active group.

As can be seen, this had no influence on the results. The last column shows that for all the players in the active group the highly interested player classification took place before they have played half of their total time, whereas without the active probing it took the vast majority of a player's total game-play time to classify the player.

TABLE IV
COMPARISON BETWEEN PLAYERS THAT WERE CLASSIFIED AS HIGHLY ENGAGED AND PLAYERS WHO WERE CLASSIFIED AS MORE CASUAL BASED ON MAJOR GAME METRICS AVERAGES.

Highly Engaged?	Score	Gold	Moves	Kills	Souls
Yes	1483	1230	4251	535	2682
No	53	48	566	35	145

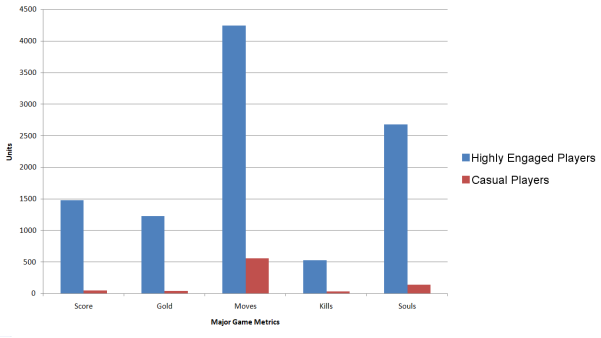

Fig. 1. Graphical representation of Table IV

Table IV compares the player groups that were classified as highly engaged and those that were not on some major game metrics (averaging over all players in each group). Figure 1 makes the trend even clearer: among the players that were classified as highly interested in the game, there was a higher average interest score, higher average total gold earned, they have moved almost 8 times as much as their counterparts and killed 500 monsters more on average. This is not surprising, as players that show interest in the game would likely play the game more. Finally, players that were classified as highly interested have used the (very boring) FakeAd system 18 times more on average!

Table V expends on the comparison of the player groups that were classified as highly engaged and those that were not, by comparing the key game analytics events for those two groups. We compare the following between the two groups to show that indeed the groups are very distinct in their behaviors:

- *Average total time played* for each group, measured in seconds for each player.
- *Average total experience gained* in-game by each group.

TABLE V
COMPARISON BETWEEN PLAYERS THAT WERE CLASSIFIED AS HIGHLY ENGAGED AND PLAYERS WHO WERE CLASSIFIED AS MORE CASUAL BASED ON MAJOR GAME METRICS AVERAGES.

Metric (Average)	Highly Interested Players	Other Players
Time Played (s)	8214.3	1597.2
Experience Gained	356491.3	387.1
Number of Sessions	7.6	2
Player Movement	3793	516.5
Creatures Killed	463.8	30.5
Shop Sessions	42	1.6
Private Messages	8	1.2
Public Messages	38.8	2.6
Quests Completed	13	6
Fake Ads Watched	19	0.6
Souls Collected	2350	75
Soul Items Purchased	27.6	0.3
Bosses Killed	2.1	0.03
Maps Area Traveled	21.3	1.7

- The *average number of sessions* players in each group played.
- The *average number of movement commands* sent to the server from players in each of the groups.
- The *average number of creatures killed* by players from each group.
- The average number of times a *player interacted with an in-game shopkeeper*. This event is required for a player to buy and sell equipment in the game, both using gold and souls.
- The *average number of public and private messages* sent by players from each of the groups. Public messages are messages that were sent in the general chat, which every other online player can see. Private messages are ones directed to NPC or other players and only the given player sees them.
- The *average number of quests completed* by players in both groups.
- The *average number of Fake Ads watched* by players in both groups.
- The *average reward of Souls for Fake Ads* by players in both groups.
- The average number of *redeemed benefits* in the form of purchases of items by players in both groups. These ranged from stronger healing potions to teleport scrolls and blessings that temporally removed death penalties.
- The *average number of boss monsters killed* by players in both groups. These monsters are farther in the dungeons and require longer playing sessions to reach, as well as have a higher difficulty level.
- The *average number of maps traveled* by players in both groups. Each map has connection points to other maps, which players must reach in order to travel deeper into the dungeon or back to town. This category of events represents how far and how often players traveled in our game.

As can be clearly seen from Table V, Highly Interested Players highly exceed the other players on these core game

metrics. This is not surprising, as these metrics were chosen to measure core game activities - and highly active players are likely to be highly interested in the game. This validates that the players our system classified as highly interested in the game are indeed so, regardless of active probing.

Together, Table V and Table IV show that what our system was classifying was indeed two separate groups of behaviors, where one group is clearly more engaged in the game than the other. This classification is clearly distinct and players are usually quite obviously placed in one group or the other. Finally, as Table III shows, the amount of time to find the classified players as highly engaged was much shorter in the active probing group. This fulfills the initial goal of the active probing - to reduce the time of data collection to achieve this player classification, so that the benefits of knowing a player is highly interested in the game could be acted upon earlier by game producers, maximizing player conversion rates, monetizing and deriving game design decisions as quickly as possible.

V. Related Work

There are three related areas of work to our research. The first area of work that has seen some attention from researchers dealt with customizing the game experience for different players. This topic has been widely explored and covered areas such as adjusting the difficulty of the game, generating content based on player actions and using an AI to direct the flow of the game on the fly [8], [10], [13]. A special note should be made when comparing our GMAI system to "Experience Managers" or "AI Directors" as in the work of [7], [11]. While both systems use AI techniques to interact with the player and modify the game environment, a core difference is the motivation of doing so. In classic drama and experience management, the idea is to serve a different game that would be considered "fun" for the player. In the GMAI approach, the core functionality is to identify as fast as possible if a player falls into a stereotype class that the developers and publishers of the game have defined as lucrative or at high risk. While similar techniques can be used, the two problems are very different.

The second area covers the psychology of gamers. There is a lot of work on modelling the gaming motivations of players and why players even play games to begin with, for example [2], [16]. Finally, researchers have already tried to use this knowledge to build systems that datamine information about the user [13], [14]. While our research also uses game analytics, our inclusion of active probing has not been applied in any of these areas. It would be interesting to see how more established analytics frameworks could be enhanced using this active probing approach by a Game Management AI.

VI. Conclusions and Future Work

We presented the idea to add to the use of game analytics to identify the class of a player (in our case the class of players that show signs of being highly engaged in playing a specific game) the idea of not relying on just observing game play but also actively probing the player for signs of interest. The later is achieved by using a Game Management AI, an AI system that modifies the game according to general knowledge about games, the particular game, behavior indicating high involvement of the player and other player related observations.

In our experimental evaluation, we identified signs of very high levels of interest with active probing in a quarter of the time without active probing. These players were clearly exhibiting a different pattern of behavior, and our shorter discovery time enhances both game developers' and marketing teams' ability to identify these players sooner. This either influences their behavior by smart game design that is tailored for them or promotes player interest with limited offers targeted at these highly involved individuals. Future work should look into creating more ideas for rules for the GMAI (both for the indicators we used and for additional indicators in the literature) for highly interested players but also other player classes of interest and, naturally, into applying our ideas to other games and game genres. It would be interesting to see how a more mature framework for game analytics could be improved when using active probing and managed by GMAI.

References

[1] C. Bauckhage, A. Drachen, and R. Sifa. 2015. Clustering game behavior data. IEEE Transactions on Computational Intelligence and AI in Games, 7(3), 266-278.

[2] R. Bartle. 1996. Hearts, clubs, diamonds, spades: players who suit MUDs, Journal of MUD Research, 1, 1.

[3] C. Herodotou, N. Winters, and M. Kambouri. 2015. An Iterative, Multidisciplinary Approach to Studying Digital Play Motivation, Games and Culture, 10(3), 249-268.

[4] K. Hullett, N. Nagappan, E. Schuh, and J. Hopson. 2011. Data Analytics for Game Development (NIER Track). Proc. 33rd ICSE, pp. 940-943.

[5] D.J. Kuss, and M.D. Griffiths. 2012. Internet gaming addiction: A systematic review of empirical research. International Journal of Mental Health and Addiction, 10(2), 278-296.

[6] D.J. Kuss. 2013. Internet gaming addiction: current perspectives. Psychology Research And Behavior Management, 6, 125-137.

[7] M.J. Nelson, and M. Mateas. 2005. Search-based drama management in the interactive fiction Anchorhead. AIIDE 2005, pp. 99-104.

[8] M.O. Riedl, and A. Zook. 2013. AI for game production. Proc. CIG 2012, pp. 1-8.

[9] M. Seif El-Nasr, A. Drachen, and A. Canossa. 2013. Game Analytics. Springer.

[10] J. Shen, O. Brdiczka, N. Ducheneaut, N. Yee, and B. Begole. 2012. Inferring personality of online gamers by fusing multiple-view predictions. Proc. 20th UMAP, pp. 261-273.

[11] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen. 2007. Interactive Storytelling: A Player Modelling Approach. Interactive Storytelling: A Player Modelling Approach. In AIIDE 2007 (pp. 43-48).

[12] Unity Technologies. https://unity.com/.

[13] B. G. Weber, M. John, M. Mateas, and A. Jhala. 2011. Modeling Player Retention in Madden NFL 11. Proc. 23rd IAAI, pp. 1701-1706.

[14] H. Xie, S. Devlin, D. Kudenka, and P. Cowling. 2015. Predicting player disengagement and first purchase with event-frequency based data representation. Proc. CIG 2015, pp. 230-237.

[15] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. Andr. 2013. Player modeling. In Dagstuhl Follow-Ups (Vol. 6). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[16] N. Yee. 2006. Motivations for Play in Online Games. Motivations for Play in Online Games. CyberPsychology & Behavior, 9(6), 772-775.

[17] K. S. Young. 2010. When gaming becomes an obsession: Help for parents and their children to treat online gaming addiction. http://www.netaddiction.com/articles/Online/Gaming/Treatment.pdf (as seen June 18, 2018).

[18] J. D. Zhan, and H. C. Chan. 2012. Government Regulation of Online Game Addiction. Communications of the Association for Information Systems, 30(1), 13.