An Inclusive Taxonomy of Player Modeling

Adam M. Smith, Chris Lewis, Kenneth Hullett, Gillian Smith, Anne Sullivan
Center for Games and Playable Media
University of California, Santa Cruz
{amsmith,cflewis,khullett,gsmith,anne}@soe.ucsc.edu

May 2011

Technical Report UCSC-SOE-11-13

Abstract

"Player modeling" is a loose concept. It can equally apply to everything from a predictive model of player actions resulting from machine learning to a designer's description of a player's expected reactions in response to some piece of game content. This lack of a precise terminology prevents practitioners from quickly finding introductions to applicable modeling methods or determining viable alternatives to their own techniques. We introduce a vocabulary that distinguishes between the major existing player modeling applications and techniques. Four independent facets define the kind for a model: the *scope* of application, the *purpose* of use, the *domain* of modeled details, and the *source* of a model's derivation or motivation. This vocabulary allows the identification of relevant player modeling methods for particular problems and clarifies the roles that a player model can take. It is intended to be a general vocabulary, applicable to all game genres and research approaches.

1 Introduction

Consider this vignette: You've had some exposure to the concept of player modeling via a mixture of mentions in literature and an intuitive sense of what a model of a player must describe. Seeking your wisdom, a colleague asks you "Can you recommend a good introduction to player modeling?". "Depends," you say, "what kind?" But what are the kinds of player modeling? How do you tell if something even *is* a player model?

This paper was borne out of a conversation between the authors that started as the above. Each of us had a different perspective on the kinds and desiderata of player modeling emerging from our exposure to different literature and undocumented game design practices. The term "player model" is nebulous, and a broad spectrum of possible definitions have been used in various publications. While many people have described player modeling and discussed its importance, the definition and usage of the idea has grown organically over time, with no single clear definition or scope.

Our goal is to categorize the different kinds of player modeling that exist in practice in a way that makes clear which kinds of player models are applicable to which problems, what related kinds of models make for viable alternatives and what areas have yet to be explored – to help researchers provide colleagues such as the one in the vignette with some helpful direction. To do this, we introduce a vocabulary based on four independent facets which distinguish player models on the basis of who they apply to (*scope*), what they are used for (*purpose*), the kind of details they model (*domain*), and how they are derived or motivated (*source*). This vocabulary abstracts over the type of game and is readily applicable to games from any genre.

Our strategy in introducing this language is not to provide a final definition of player modeling. Instead, we adopt an *inclusive* approach, casting a wide net to consider any work that has been published using the term "player modeling" to be a valid instance. We will also include several techniques implicit in game design practices that are not traditionally called player modeling, such as the process of interpreting actions observed in playtesting, because they are highly relevant to the purposes of other player modeling work.

It is not our intention to provide a comprehensive survey of player modeling. Instead, we often elect one or two examples of a particular kind of player modeling to stand for the whole, introducing published applications and techniques as they arise in the exposition.

We begin with a brief sampling of existing player models to establish the breadth of examples we wish to classify and to suggest how easily distinctions arise when talking about many applications at a time. Then, we introduce and apply the concepts and terminology of our taxonomy, demonstrating how to drill down on the kind of a player model by making a series of small judgments. Later, we visualize the space of modeling techniques suggested by the taxonomy and share discussions our own internal use of the taxonomy has prompted. Finally, we suggest new avenues of research that hinge on the presence of an effective taxonomy of player modeling.

2 Breadth of Player Modeling

To the best of our knowledge, the earliest published work that explicitly defines player modeling is Houlette's "Player Modeling for Adaptive Games" from 2003 [9]. Houlette describes models of individual players that are created by monitoring gameplay, recording numerical values such as the number of smoke grenades a player throws. The game's artificial intelligence (AI) can query these values to adapt itself: to "change and evolve with time to suit the player." Around the same time, Charles and Black called for games to become more adaptive to different types of players with their proposed framework for "player-centered" games [5]. This was achieved by using input from both Houlette-inspired gameplay metrics and explicit interrogation of the player. As in Houlette's description, the player model created from these inputs is used by the game's system to realize an experience uniquely adapted to a single player. Charles and Black go on to describe player models as having two main purposes: classifying player behavior, and to "instill human-like qualities into a non-player character."

The more general concept of a player model as a computational model of player behavior predates Houlette's usage of player models for adapting to particular individuals. Laird's QuakeBot [12] used a model of its opponent – the human player – as an integral part of its functioning. This model was neither specific to any individual nor adaptive to observed opponent play. Further, QuakeBot's model of the player was implemented as a recursive query to itself ("what would I do in the opponent's situation?"). That is, QuakeBot is both a system that uses a player model and indeed a meaningful player model itself. QuakeBot's design is an instance of the much older game playing AI tradition of using generic models of an opponent to reason over their potential next moves when selecting its own, an idea dating back to 1950 [17].

These two threads alone demonstrate that the broader idea of player modeling has never been tied uniquely to a particular kind of application (e.g. adaptation or forward simulation). Attempting to synthesize these initial examples into a definition of a player model – perhaps as a software process which represents a certain observed individual or general class of players that can be queried by a game system to predict actions in play – is inadvisable. This would miss the even more pervasive kind of player modeling naturally engaged in by game designers when they form internal models of players from a mixture of observation and accumulated design experience. Schell advises new game designers: "you must adopt their mental perspective as well, actively projecting yourself into the mind of your player" [16]. These models, though rarely externalized, arguably influence the gameplay experience at a far deeper level than any realized adaptive game system. They inspire specific choices in the design and production of the game.

Houlette's original terminology applied to a kind of prototypical player modeling, models that describe in-game actions of an individual player as induced (learned) from recorded data. This prototype provides an interesting contrast with more recent work. In EMPath [20], a *Zelda*-style adventure game utilizing an AI system for drama management, two additional distinct types of player modeling are used. In the drama manager's look-ahead search in story-space, one kind of player model generates game actions (triggering plot points), and

another kind of player model generates reactions in place of a human player (reporting how well the story seemed to flow or how manipulative the drama manager felt). These models are motivated by the designers' understanding of the game's world and story components, as opposed to an empirical view. A different pairing of player model types occurs in a track generator for racing games [26] where action generators (driving agents) are fit to individual human players' performance, and a separate model (one which is equally applicable to all players) generates subjective reactions to proposed tracks (the fitness function mapping the performance of the individually-trained controllers to an evaluation of fun).

There are other models of players in use today that do not appear in the literature under the index of player modeling. The concept of perfect play, where an imagined player makes choices with full knowledge of all of a move's possible futures, is a player model despite the non-existence of perfect players for most games. Labeled strategies such as the specification of a build order for a real-time strategy game must be a kind of player model as well, in that they allow interesting conclusions to be drawn about the potential behavior of players who adopt these strategies. However informal, the player models indicated by lines sketched over two different routes through a platformer level allow distinguishing between different styles of play (one, perhaps, showing the route of a "speed-runner" and the other of a "completionist").

Today, we are faced with a variety of systems that all use what they call a player model, but have no means of precisely communicating what type of model each one is. Nor do we have a means of explaining how this relates to similar concepts that model players without reference to the the key words "player modeling".

3 Taxonomy

It is safe to introduce local distinctions when talking about only a few different examples of player modeling at a time. However, to help guide newcomers to the field and resolve the relationships between existing techniques, we need a set of terms that can be defined on their own. In this section, we introduce the organization of the key terms of our taxonomy and then provide a definition and contextual examples for each. It is important to note that the four primary facets of our taxonomy are non-hierarchical; they are orthogonal to each other.

3.1 Overview

Each player model in our taxonomy is described with a *kind*. These kinds are defined by a selection from each of the four independent *facets*. In Figure 1, a kind of player model can be formed by picking a single selection from each of the columns. Stringing the names of the selections together (in any convenient ordering) provides a very dense label for one corner of the space of player models we imagine, e.g. "Individual Induced Descriptive Reaction" and "Universal Synthetic Generative Action." In such labels, we capitalize the selection names to make it clear that we are referring to the sense of these words defined in our taxonomy as

Scope	Purpose	Domain	Source
Individual *applicable only to one player* Class *applicable to a sub-population* Universal *applicable to all players* Hypothetical *unlikely to be applicable to any players, but interesting nonetheless*	Generative *literally produces details in place of a human player* Descriptive *conveys a high-level description, usually visually or linguistically*	Game Actions *details recorded inside of the game's rule system* Human Reactions *details observable in the player as a result of play*	Induced *learned/fit/recorded by algorithmic means* Interpreted *concluded via fuzzy/subjective reasoning from records* Analytic *derived purely from the game's rules and related models* Synthetic *justified by reference to an internal belief or external theory*

Figure 1: An overview of the terms in our taxonomy. For each *facet* (column), choose one selection, and the resulting labels will compactly describe a *kind* of player modeling such as "Class Synthetic Descriptive Action" player models

opposed to a more colloquial sense. Facets can be omitted to abstract over a broader range of player models as in "Action Generators" and "Hypothetical Analytic models."

It is important to think of the kinds of player models definable with this vocabulary as archetypes. Practical applications of player modeling may not strictly conform to one kind or another, but they will often have a single archetype which they most closely resemble. Additionally, a single application may meaningfully embody more than one kind of player model, as in EMPath's distinct Action and Reaction Generator components.

In the definitions that follow, we work from left to right in Figure 1. Occasionally, we make use of selection names that have not yet been defined in an effort to demonstrate the full vocabulary in use. In these cases, the table can be used to look up which facet defines the unfamiliar term in later subsections.

3.2 The Scope Facet

The Scope of a model describes to whom the model is intended to be relevant or who is being distinguished in the model.

3.2.1 Individual

Individual models pertain only to one player, allowing comparisons between "this player" and "that player." These models are the basis of personalization in games, such as the kind proposed by Charles and Black [5], and capture the scope intended by Houlette's player models [9].

Drivatars in the game *Forza Motorsport* [8] are AI systems that are trained on a particular player's driving style by recording the player's performance over reference track segments. Once trained, the model can stand in place of the original human player, allowing

asynchronous competition with friends by simply having them race against your personal Drivatar. Applying a complete label, Drivatars are Individual Induced Generative Action models.

Individual models need not by tied to human players; our taxonomy allows one to speak of the player model embodied in any identifiable game playing agent. In this way, we can refer to the Individual model of Deep Blue, the chess-playing computer, and draw contrast between the machine's style of play and its most famous human opponent, Gary Kasparov – likewise for the machine Watson[1] and the humans Ken Jennings and Brad Rutter on the televised game show *Jeopardy!*.

3.2.2 Class

Class models distinguish between populations of players, often taking the form of a partitioning of the audience for a particular game into recognizable categories. Class models are pervasive as they correspond to reasoning about stereotypes; for example, it is easy to imagine that a designer might think "I want my game to be completable by speed-runners in no less than six hours, but I want explorers to be able to find at least twenty hours of content."

Bartle's player types are Class-scoped models that partition multiplayer online game players into the categories of "achievers", "explorers", "socializers", and "killers" [3]. For each category, Bartle describes both in-game actions (that achievers will seek a 100% completion rating) and personal reactions (that explorers will experience enjoyment in having found a game glitch). Thus, Bartle's player types are Interpreted Descriptive Class models.

In contrast to Bartle's well-reasoned vocabulary, informal player types such as "newbie" and "griefer" may not provide a clean partitioning of the space of players, nor will "casual" and "hardcore" be possible to formally define in a universally satisfying way. Nonetheless, the use of such labels in the definition of a model simultaneously signifies the intent to scope the conclusions of that model to a specific sub-population of all players while not drawing distinctions at the level of individual players.

3.2.3 Universal

Universal models apply equally to all players. Where Individual and Class are often used to draw distinctions between players, Universal models are intended to capture assumptions or conclusions that should hold in general. Almost any system component which Generates or Describes Actions or Reactions (in the senses we describe later) without being explicitly scoped to a class or individual can be seen as an instance of a Universal model. Concretely, the fitness functions used in genetic algorithms, such as the fun estimate in the previously mentioned track generator [26], are Universal player models (the same code runs for all players).

As attempts at expressing universal truths, these player models will certainly demonstrate a range of accuracy. The creators of EMPath created a sequence of increasingly refined

[1] `http://artsbeat.blogs.nytimes.com/2011/02/14/on-jeopardy-watson-rallies-then-slips/`

player models for the plot-point selection component of their system, ranging from a simple uniform probability distribution model, which completely ignored the traversability of the game world's map, to a richer model that used a spatially anchored distribution to capture expectations about a player's tendency to wander [20]. While each of these models would occasionally generate actions that would be impossible for a human player to realize, the intent of these models to stand in place of human players (though no player in particular) is what makes them Universal Generative Action models.

3.2.4 Hypothetical

Where individual models are defined with a particular, usually human, player in mind, Hypothetical models speak to unlikely, edge-case, or counter-factual situations in gameplay. The idea of a player with foreknowledge of his opponent's next move, knowledge of the location of hidden pieces, or the ability to manipulate dice rolls in his favor is a player model that, despite being impossible to realize, is valuable for what it can expose about the design of a game (e.g. the presence of an exploit or rare catastrophic failure case). The notion of perfect play in chess is a familiar Hypothetical Analytic Descriptive Action model.

In Ludocore [18], a tool for producing logical models of video games, a designer can ask the system for traces of actions that demonstrate how to satisfy arbitrary logical constraints in play. Using the mechanism of "speculative assumptions", the designer can shape a Ludocore model into a Hypothetical Action Generator that is specific to the sharply defined spaces of the designer's current interest, often as part of debugging the game's core mechanics (perhaps defining a Hypothetical player model that generates only those actions which achieve victory in twelves steps while holding no more than one item in a character's inventory at a time in an effort to expose an exploit in the rules).

Selecting amongst design alternatives at the conceptualization stage of game design requires applying a kind of "what if" reasoning over the play for games that have yet to be realized. To the degree that this reasoning allows a designer to describe potential actions or reactions, the imagined play encodes a player model, particularly a Hypothetical Synthetic model. These are the models that permit the creation of visual storyboards which "simulate a player moving through the game" despite the game only existing at the level of an idea [7].

3.3 The Purpose Facet

The Purpose of a model describes the function of a model in its intended application. While a given player modeling technique may afford multiple functions, the Purpose of a model refers to its actual use.

3.3.1 Generative

Generative player models generate data where a human player could otherwise be consulted. These models are generally algorithms, taking the form of either a specialized procedure or a set of inputs to a previously defined procedure (parameters to a function, index into a table,

etc.). Often, Generative models will only be able to stand in for a human player in one area of competency. Drivatars intend to capture and reproduce a particular player's racing performance [8]; however, the Drivatar cannot be used to predict which cars and upgrades the player would buy outside of the racing game mode. Polymorph, a platformer level generator with dynamic difficulty adjustment [10], is an example of a system that uses two Generative Reaction models: one that learns the difficulty of level segments according to a Universal model of difficulty, and one that learns the skill of Individual players. Automating the difficulty evaluation process for level segments allows the system to cull generated levels that are a poor fit for the current player.

3.3.2 Descriptive

While Generative models are most useful in the context of live software systems, Descriptive models intend to convey information to human interpreters, such as game designers wishing to better understand their games and the people who play them. Descriptive models contain high-level, informative messages, often encoded in natural language or visual imagery. The Playtracer system [2], a tool for analyzing recorded play traces, produces Individual Induced Descriptive Action player models. These take the form of a directed graph indicating (for several players at a time) trajectories through the abstract puzzle space of *Refraction*. These highlight common stumbling blocks on a player's progression towards a goal state. The generated heat maps for *Halo 3* describe a distribution of actions, such as being killed at particular locations on a map, across the game's entire player base [22], making them Universal Induced Descriptive Action models.

The depth of the message conveyed by a Descriptive model varies dramatically, ranging from lengthy articles in the case for Bartle's player types [3] to a single word paired with only its conflicting, informal definitions, as in the case of "hardcore." When a Descriptive model describes a concrete strategy, it is tempting to create a Generative model that serves as an executable reference of this strategy (as done in the NonyBot system described later).

In the context of the late-stage design of new games, descriptive models can be used to document assumptions made about the game or its intended audience. The walkthrough solutions (or input scripts) sometimes provided with interactive fictions are Individual Synthetic models that afford both Generative and Descriptive purposes: they are both executable as Action Generators and readable as natural-language Action Descriptions [14]. That is, to judge the Purpose of such models, one needs to reference a particular usage of the script.

3.4 The Domain Facet

The Domain of a model answers the question of what it is that the model generates or describes.

3.4.1 Game Actions

Actions are in-game choices made by the player – moves in combat, navigation, upgrade selection, inventory arrangement, team management, etc. – including choices that do not directly affect gameplay such as avatar personalization. At this point, we have already introduced several of these models: the player model that triggers plot points in EMPath; QuakeBot that is able to run, jump, collect and equip weapons, and fire at opponents (an Individual Synthetic Generative Action model); and two kinds of player models that are able to drive race cars (Drivatars and the evolved controllers of Togelius' system).

3.4.2 Human Reactions

Human reactions cover the properties of a gameplay experience that exist outside of the game's simulated world. These include objective physiological measures such as heart rate and eye movement as well as subjective measures such as the level of fun, challenge, and frustration that player might report in a survey or interview. Reactions also include unrecorded sentiments about the game itself or other human players who were met in game-mediated interactions. Finally, our definition of human reactions include game-related activities such as purchasing a game, cancelling a subscription, or conversing with other players in simultaneous voice chat.

Yannakakis has devoted a long line of work to building models of player reactions [28]. Where this work generally produces Induced Reaction models, game designers will often build up strong expectations for how their audience will react to their work, exercising Synthetic Reaction models. These models that are similar in character to Csikszentmihalyi's theory of flow which predicts reactions such as boredom and arousal or anxiety and relaxation in response to the balance to the game's challenge and the player's skills [6].

3.5 The Source Facet

Finally, the Source facet describes how a player model is motivated or derived. Broadly, player model sources can be categorized as Empirical or Theoretical (which we will occasionally use as pseudo-selections), but we provide four mutually distinct selections below to help further distinguish player models. Empirical (Induced and Interpreted) models intend to capture the external truth present in data recorded from the play of real human players. Meanwhile, Theoretical (Analytic and Synthetic) models intend to represent the truth of a gameplay situation, absent of any direct references to data. Splitting these four labels another way, Subjective (Interpreted and Synthetic) models hinge on the credibility of their inventor (an analyst or designer), where as Objective (Induced and Analytic) attempt to stand on their own.

3.5.1 Induced

Induced models use recorded data with an objective, often automated, inductive analysis. The best Induced models will use the established form of machine learning other statistical

analysis for the modeling problem so that uncertainty about the model applies mostly to the data itself as opposed to the choice of inductive method. Player modeling work descended from Houlette's original work focuses specifically on Induced models, even if the learning method is only maintaining weights or counts of events. The training process for Drivatars is another such induction process. In an application to *World of Warcraft*, a very large scale data analysis effort (and even the invention of a new matrix factorization technique) resulted in a Class Induced Descriptive Action model for player guilds which illustrated archetypal guild experience distributions [24].

In many systems, an Induced model is paired with a different kind of model that makes more detailed statements about the player. In Thue's PaSSAGE system [23], an Individual Induced Descriptive Action model maps a player to a label (associating players who often engage in combat actions with the "fighter" label). Another model in the same system, a Class Synthetic Generative Reaction model, takes the form of a table encoding how players of a given type rate the suitability of a given event happening in their play experience (predicting that a fighter would rate "headlong assault" strongly positive). Layering these two player models together results in an Individual Synthetic Generative Reaction model which is able to, on a person-by-person basis, generate suitability scores which can be used to select the best event to trigger in the game world next (realizing an adaptive, personalized gameplay experience). This kind of layering (pairing a Descriptive model with a Generative model) can be used to incorporate Descriptive models into the live execution of a game. In layering, two different kinds of models are used to implement a larger player model, that when treated as a black box has a distinct kind of its own.

3.5.2 Interpreted

Interpreted models are a more subjective alternative to Induced models. To create some of the richer Descriptive models (such as Bartle's player types), human interpretation is needed to map the empirical observations to informative descriptions. Given access to a set of recorded play traces or unstructured player interviews, an interpreter can use their past experience and intuition to make more sense of the data than a hard statistical method. The interpretive process may involve generalization from incomplete data, producing natural-language summaries, or hand-picking interesting examples to be used as references in defining the player model.

Not all Interpreted models are Descriptive, however. The process of capturing a player's moves in a hand-coded algorithm produces an Individual Interpreted Generative Action model. NonyBot[2] is a bot for *StarCraft* that intentionally attempts to parody the repetitive play style and strategy of a particular well-known expert human player.

3.5.3 Analytic

On the Theoretical side, Analytic methods use other automated methods (such as theorem proving, search, or optimization) to extract the truths inherent in a game's design in a useful

[2] `http://www.youtube.com/watch?v=AxBHwpItv84`

form. Machine-generated endgame tablebases for chess are a simple example of Universal Analytic Generative Action player models – the rules of the game have been exhaustively searched to yield the objectively best outcome (and moves to achieve this) for a variety of late-game situations [4].

Analytic models exist outside of board games; metrics already computed by a game's system can be used with existing optimization and learning methods (e.g. applying policy optimization or reinforcement learning with the game's score as a reward function [21]) to produce Analytic Action Generators. In a genetic algorithm setup for evolving game bots, the genetic individuals which the algorithm manipulates are Individual Analytic models because they are mutually distinguished, while having been constructed purely syntactically by some series of mutation and crossover operations (objectively). If the (Universal) fitness function used in the genetic algorithm simply evaluates genetic individuals by a predefined in-game metric, we say that the fitness function itself is an Analytic Generative Reaction model (that is, we imagine the fitness function as a kind of stand-in in for a post-play survey asking the individual how it thought that it had performed). Togelius et al. use this setup to evolve agents for *Infinite Mario Bros* [25].

Dropping the reference to a particular agent framework or metric to be optimized, the naked rules of a game directly define an implicit player which describes players as potentially capable of doing anything the rules allow; Ludocore falls back to this default Universal Analytic Generative Action model when no special player modeling assertions are provided [18].

3.5.4 Synthetic

The hallmark of a Synthetic player model is a reference to some concept that comes from outside of the game itself. Two published systems make this reliance on external ideas clear. Smith's platformer level generator, Launchpad, operates under the assumption that players experience levels rhythmically and uses this to derive a model of level traversal times [19], a Universal Synthetic Generative Action model. In another content generation project, Togelius and Schmidhuber's "An Experiment in Automatic Game Design" [27] derives its model of fun (the fitness function, a Universal Synthetic Generative Reaction player model) from Koster's theory of fun [11] and Csikszentmihalyi's notion of flow [6].

The use of hunches, intuition, and other beliefs which are not traceable to any particular piece of evidence is a kind of Synthetic player modeling that is pervasive in game design. This is not to say these models are unfounded; they may derive from knowledge transferred from an Interpreted player model for another design project or from game design wisdom that is difficult to communicate. The uses of informal player models for documenting assumptions about the audience that we mention when discussing Descriptive models are all Synthetic as well.

4 Visualization

Looking at all combinations of selections for facets yields sixty-four distinct kinds of player models (notwithstanding the ability to layer different modela together or using models for separate purposes within a single application). To concisely summarize all of the player models we have explicitly mentioned, we created a visual representation of our taxonomy in Figure 2 which maps different paths through the four facets down to non-overlapping regions. Although this paper is not intended as a comprehensive survey of player modeling, some regions of the diagram are clearly more active research areas than others. Possible reasons for this are discussed in the following section.

5 Discussion

In performing a deep analysis of player models in order to create the taxonomy, a number of questions regarding the character of player modeling were raised. This section discusses the broader implications of the taxonomy.

5.1 Character and Opponent AI

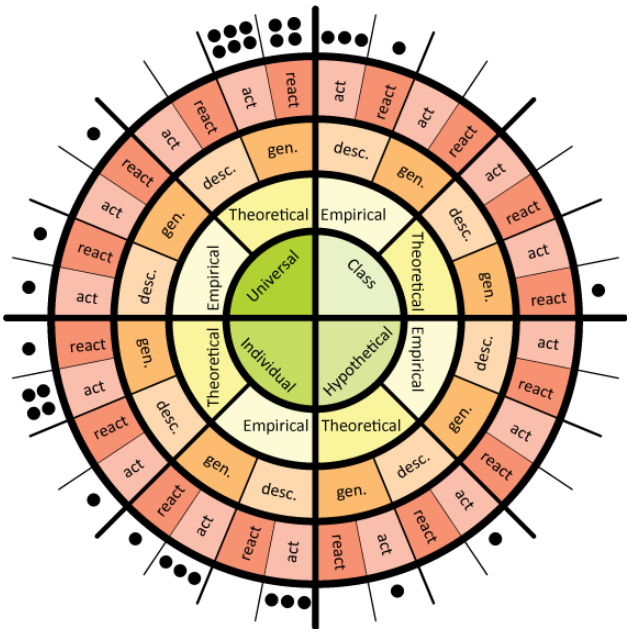
Although our taxonomy includes Action Generators, we caution that not everything that generates actions taken by a character in a game should be considered a player model. There are two major questions we can ask to determine if a particular AI system uses a player model: does it take the same actions that a player can take, and does it reason about player behavior?

Consider the prototypical shopkeeper in a fantasy computer role-playing game. The character takes a set of actions in the same game world as the human player, and these actions are generated. However, the shopkeeper fails both of our requirements for an AI to be considered a player model. No human player can take on the role of the shopkeeper, and in reacting to the player's requests for purchasing weapons and potions, it does not typically reason about the player's behavior.

Market trading bots for games like *World of Warcraft* and *EVE Online* are an example of AIs that do use player models [15]. Even though the bots have a limited set of actions that they can take, they are actions that are normally able to be taken by the player. The bots use a Universal (or Individual, if customized) model of player behavior that describes the appropriate time to buy and sell items in the market.

However, we should also consider the case of opponent AI. Opponents in asymmetrical games, such as *Tetris*, take different actions from those of the player (in this case, deciding which piece comes next) but may make use of a player model to better adapt its choices. *Bastet*[3], a particularly evil variant of *Tetris*, uses a player model to determine the worst possible piece to give the player next. While it uses a player model, the piece-giver in *Bastet* is not a player itself. Meanwhile, recall QuakeBot, the agent for *Quake* which imagined its

[3] `http://fph.altervista.org/prog/bastet.html`



Instance	Scope	Source	Purpose	Domain
"Speed-runner" and "completionist"	Class	Interp.	Descr.	Act.
Bartle's player types	Class	Interp.	Descr.	Both
WoW guild archetypes (Thurau)	Class	Induced	Descr.	Act.
PaSSAGE (Thue)	Class	Synth.	Gen.	React.
Storyboards (Fullerton)	Hypo.	Synth.	Descr.	Act.
Ludocore (Smith)	Hypo.	Analytic	Gen.	Act.
Houlette	Indiv.	Induced	Descr.	Act.
Playtracer (Andersen)	Indiv.	Induced	Descr.	Act.
PaSSAGE (Thue)	Indiv.	Induced	Descr.	Act.
Race track generation (Togelius)	Indiv.	Induced	Gen.	Act.
Drivatars	Indiv.	Induced	Gen.	Act.
NonyBot	Indiv.	Interp.	Gen.	Act.
Polymorph (Jennings-Teats)	Indiv.	Induced	Gen.	React.
Interactive fiction walkthroughs (Reed)	Indiv.	Synth.	Both	Act.
QuakeBot (Laird)	Indiv.	Synth.	Gen.	Act.
IBM's Deep Blue and Watson	Indiv.	Synth.	Gen.	Act.
Mario bots (Togelius)	Indiv.	Analytic	Gen.	Act.
PaSSAGE (Thue)	Indiv.	Synth.	Gen.	React.
Heatmaps for Halo 3	Uni.	Induced	Descr.	Act.
Preference modeling (Yannakakis)	Uni.	Induced	Descr.	React.
Polymorph (Jennings-Teats)	Uni.	Induced	Gen.	React.
Engames tablebases (Bellman)	Uni.	Analytic	Gen.	Act.
EMPath (Sullivan)	Uni.	Analytic	Gen.	Act.
IMPLANT (Tan)	Uni.	Analytic	Gen.	Act.
Ludocore (Smith)	Uni.	Analytic	Gen.	Act.
Market bots	Uni.	Synth.	Gen.	Act.
Launchpad (Smith)	Uni.	Synth.	Gen.	Act.
EMPath (Sullivan)	Uni.	Synth.	Gen.	React.
Race track generation (Togelius)	Uni.	Synth.	Gen.	React.
Flow inspired (Csikszentmihalyi)	Uni.	Synth.	Gen.	React.
Mario bots (Togelius)	Uni.	Analytic	Gen.	React.

Figure 2: A visual summary of all of the player models explicitly mentioned in the body text. The Induced and Interpreted selections have been collapsed into the Empirical label to reduce visual complexity, likewise for Analytic and Synthetic with the Theoretical label. In the accompanying table, rows are ordered as to label the dots in a clockwise fashion starting at the top.

opponents actions to decide which was best to choose for its own. QuakeBot both is a player model and uses a player model (likewise for tree-searching Deep Blue).

An interesting edge case to examine is that of the cheating AI. When an opponent AI usually makes use of only player actions, but occasionally "cheats" by using special AI-only actions, the categorization is less clear. If opponent bots in a racing game drive fairly while on screen, but when not visible rubber-banding allows them to drive faster than their cars allow, the bots are Generative Action player models (albeit with times when they become very inaccurate models). The same is true for a real-time strategy opponent which can spend unearned resources but nonetheless builds buildings and orders units around as the player does. Creating an executable model that closely matches human players (especially particular individual players) is very difficult, and falling back to occasional cheating or the use of generic procedures is common. For example, the pit stop behavior of individual Drivatars is not specialized to individual players (a Universal model takes over for pitting) [8].

5.2 Demographics

We considered including "demographics" as a selection of its own for the Domain of a player model. This selection would have covered work that primarily focused on who the player was in terms of gender, age, nationality, educational background, etc. However, upon closer examination, we found that some of this work actually links demographically-defined sub-populations to what they will do in a game or how they react to play, thus communicating Empirical Descriptive player models.

The book "Gender Inclusive Game Design" [13] is one such example: it delves deeply into both the Actions and Reactions of male and female players in various kinds of games. Other work that covers demographic details without talking about implications for play, such as the ESA's report [1] on the state of the game industry, is not considered player modeling. Such analyses seem to not be modeling populations of players as players, but more as populations of consumers.

5.3 Usage and Publication Bias

In our informal estimation, published work seems to cluster around Universal Theoretical models and Individual Empirical models. We think this is because theories usually attempt to be universally applicable and individuals are the primary source of empirical data, making these models the most direct. Class models are more difficult to motivate in an academic context, requiring either justification of a theory of stereotypes or aggregation of sufficient individual data to build up class descriptors. Thus, we expect class models to be used more in practice than they are reported.

Game designers regularly invent Individual and Class Synthetic models as a product of their amassed design experience. These ephemeral models are difficult to convey without the sum of indirect experience and other pet theories that inspired then, and thus they do not appear in the record of literature.

Finally, Hypothetical player models are ubiquitous in work published as game theory or optimal control where they apply to game-theoretic games or state-transition formalisms (such as Markov decision processes). This work does not seem to be true to the sense of games foregrounded by work that explicitly mentions player modeling. However, the introduction of formal logic tools for video game design, such as Ludocore [18], are starting to allow rigorous statements about edge-case and counter-factual play for constructs recognizable as video games.

5.4 Unpopulated Taxonomy Areas

Figure 2 visually suggests that there are holes in our taxonomy: categories that apply to no realized player models. While some areas would remain unpopulated in even the most comprehensive survey of player modeling work, we assert that there are no unrealizable combinations. Consider the seemingly contradictory kind of Hypothetical Induced models: at first blush, it seems unlikely that one could build a model of hypothetical player behavior when looking at data from real players.

However, suppose that an analysis of logs from a first person shooter shows that all but one weapon in the game is used relatively frequently. Immediately one tries to think of what it is like to play using that weapon to guess at why it would be ignored – this is a Hypothetical Induced Descriptive Action player model. It is Hypothetical because the designer is exploring behavior demonstrably unlikely to arise in normal game play. It is Induced because the motivation to examine this particular weapon comes from an empirically-determined gap in the data. Finally, it is a Descriptive Action model because it is constituted in the simple description "the player uses this weapon," regarding a player's in-game choice.

6 Future Work

One function of our taxonomy is to quickly decide when two or more player models are related (i.e. sharing some or all selections for their facets) and when they are not. With the ability to clearly define a region in the space of player models, we imagine further research tackling the problem of providing a more comprehensive survey of work done on a single kind, such as one covering Individual Induced Generative Action models (player-specific bots akin to Drivatars), or of a collection of kinds, as in simply Individual Induced models (the larger set of systems that learn from individual players).

Having exposed several implicit yet meaningful kinds of player modeling that are systematically underrepresented in literature (particularly those that relate to the mental processes in the not-well-understood practices of game design), we can also imagine an effort to document these otherwise invisible instances of player modeling. Currently, the explicit use of such mental models cannot be found in any introductory text and it must be rediscovered by each game designer in the course of their early experiences. Statements such as "designers imagine how different kinds of players will play through their games and react to what they experience" might seem too obvious to publish, but they convey a player-centric philosophy

that can help new designers separate their own play style or reactions to a game from a more objective view of how players other than themselves would react or play.

Finally, it seems the ability to actually define player modeling in an inclusive way is now within our grasp. Player modeling has indeed been defined many times before (recall Houlette), but no definition spans the space of relevant techniques we have collected in this document. While this taxonomy can be used as a kind of default definition of player modeling ("if you can classify it, it's a player model"), we have not tested the boundaries of this model so carefully as we have considered the distinctions it draws between realized systems.

7 Conclusion

Our goal has been to clean up the terminology of player modeling in a way that ties modeling methods (best distinguished by their Source) to the modeling problem they solve (described by their Scope, Purpose, and Domain). We have taken an inclusive strategy that intends to address all work published as "player modeling" and beyond into the internalized models used by game designers.

Using the rich vocabulary we have defined, it is now possible to make very concise descriptions of specific instances of player modeling, for example identifying Drivatars as Individual Induced Generative Action player models. Further, we can now point out the general strategy behind the nuanced layering of player models in PaSSAGE in which an Individual Induced Descriptive Action models was plugged into a Class Synthetic Generative Reaction model to produce an Individual Synthetic Generative Reaction models which was exactly the technology in PaSSAGE that enabled the novel gameplay experience in that system.

By considering different player modeling approaches within the same kind, drop-in alternatives can be found. In fact, a study of EMPath integrated a series of increasingly refined player models of the same kind and observed their effects on play [20]. Looking outside of a specific kind, player models of a nearby kind can inspire new development. For example, replacing EMPath's single Universal Synthetic Generative Reaction model (the story quality evaluator) with a set of hand-coded Class-scope model would allow for coarse-grained player adaptation with only limited changes to surrounding systems in the game. Doing this would imply a subtle redefinition of the problem of player modeling in that system without changing the Source (the author's hunches about player reaction).

We hope this taxonomy inspires the reader to adopt a more inclusive view of player modeling and inspires them use the distinctions it draws to formulate new and interesting conclusions about player modeling that were previous difficult to express in a general way.

8 Acknowledgements

The authors would like to thank Michael Mateas, Mark J. Nelson, Noah Wardrip-Fruin, and Jim Whitehead for their support, insightful comments, and interesting counterexamples.

References

[1] 2010 Essential Facts About the Computer and Video Game Industry. Technical report, Entertainment Software Association, 2010.

[2] E. Andersen, Y. E. Liu, E. Apter, F. B. Genesse, and Z. Popović. Gameplay analysis through state projection. In *Proc. of the 5th Int'l Conf. on the Foundations of Digital Games*, FDG '10, pages 1–8, 2010.

[3] R. A. Bartle. Players Who Suit MUDs. *Journal of Online Environments*, 1(1), 1996.

[4] R. E. Bellman. On the Application of Dynamic Programming to the Determination of Optimal Play in Chess and Checkers. In *Proc. of the National Academy of Sciences of the United States of America*, volume 53, pages 244–246, February 1965.

[5] D. Charles and M. Black. Dynamic Player Modelling: A Framework for Player-Centered Digital Games. In *Proc. of Int'l Conf. on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, November 2004.

[6] M. Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper Perennial Modern Classics, 1st edition, July 2008.

[7] T. Fullerton. *Game Design Workshop, Second Edition: A Playcentric Approach to Creating Innovative Games (Gama Network Series)*, page 168. Morgan Kaufmann, 2 edition, February 2008.

[8] R. Herbrich. Drivatar - Microsoft Research `http://research.microsoft.com/en-us/projects/drivatar/default.aspx`.

[9] R. Houlette. Player Modeling for Adaptive Games. In S. Rabin, editor, *AI Game Programming Wisdom 2*. Charles River Media, December 2003.

[10] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin. Polymorph: A Model for Dynamic Level Generation. In *Proc. of the 6th Artificial Intelligence for Interactive Digital Entertainment Conf. (AIIDE 2010)*, October 2010.

[11] R. Koster. *A Theory of Fun for Game Design*. Paraglyph Press, 1 edition, November 2004.

[12] J. E. Laird. It knows what you're going to do: adding anticipation to a Quakebot. In *Proc. of the fifth int'l conference on Autonomous agents*, AGENTS '01, pages 385–392, 2001.

[13] S. G. Ray. *Gender Inclusive Game Design: Expanding The Market (Advances in Computer Graphics and Game Development Series)*. Charles River Media, 1 edition.

[14] A. Reed. *Creating Interactive Fiction with Inform 7*, page 387. Course Technology PTR, 1 edition, August 2010.

[15] J. Reeder, G. Sukthankar, M. Georgiopoulos, and G. Anagnostopoulos. Intelligent trading agents for massively multi-player game economies. In *Proc. of the 4th Artificial Intelligence for Interactive Digital Entertainment Conf. (AIIDE '08)*, Oct. 2008.

[16] J. Schell. *The Art of Game Design: A Book of Lenses*, page 99. Morgan Kaufmann, August 2008.

[17] C. E. Shannon. XXII. Programming a computer for playing chess. *Philosophical Magazine (Series 7)*, 41(314):256–275, 1950.

[18] A. M. Smith, M. J. Nelson, and M. Mateas. Ludocore: A Logical Game Engine for Modeling Videogames. In *Proc. of IEEE Conf. on Computational Intelligence and Games (CIG 2010)*, August 2010.

[19] G. Smith, J. Whitehead, M. Mateas, M. Treanor, J. March, and M. Cha. Launchpad: A Rhythm-Based Level Generator for 2D Platformers. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(1), March 2011.

[20] A. Sullivan, S. Chen, and M. Mateas. From Abstraction to Reality: Integrating Drama Management into a Playable Game Experience. *Proc. of the AAAI 2009 Spring Symposium*, 2009.

[21] C. T. Tan and H.-l. Cheng. IMPLANT: An Integrated MDP and POMDP Learning AgeNT for Adaptive Games. In *Proc. of the 5th Artificial Intelligence for Interactive Digital Entertainment Conf. (AIIDE '09)*, October 2009.

[22] C. Thompson. Halo 3: How Microsoft Labs Invented a New Science of Play. August 2007.

[23] D. Thue, V. Bulitko, and M. Spetch. Player Modeling for Interactive Storytelling: A Practical Approach. In *AI Game Programming Wisdom 4*, pages 633–646. Charles River Media, February 2008.

[24] C. Thurau and C. Bauckhage. Analyzing the Evolution of Social Groups in World of Warcraft. In *Proc. of IEEE Conf. on Computational Intelligence and Games (CIG 2010)*, pages 170–177, August 2010.

[25] J. Togelius, S. Karakovskiy, J. Koutnik, and J. Schmidhuber. Super Mario Evolution. In *Proc. of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 156–161, September 2009.

[26] J. Togelius, R. D. Nardi, and S. M. Lucas. Making racing fun through player modeling and track evolution. In *Proc. of the SAB'06 Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games*, 2006.

[27] J. Togelius and J. Schmidhuber. An Experiment in Automatic Game Design. In *Proc. of the IEEE Symposium on Computational Intelligence and Games (CIG)*, 2008.

[28] G. N. Yannakakis, M. Maragoudakis, and J. Hallam. Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(6):1165–1175, November 2009.