Proceedings of the Fifteenth AAAI Conference on Artificial
Intelligence and Interactive Digital Entertainment (AIIDE-19)

A Structured Analysis of Experience Management Techniques

Giulio Mori,[1] **David Thue,**[1,2] **Stephan Schiffel**[1]
[1]Department of Computer Science, Reykjavik University, Menntavegur 1, 101, Reykjavik, Iceland;
[2]School of Information Technology, Carleton University, 1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada
{giulio17, stephans}@ru.is, david.thue@carleton.ca

Abstract

As a field of study, Experience Management spans a set of technology that is becoming increasingly relevant in applications that aim to improve the experiences of their users. Given the youth of the field, however, few attempts have been made to identify and discuss the common elements of systems that manage user experiences as they occur. In this paper, we consider a subset of existing AI experience managers in the context of a shared conceptual framework. We offer directly comparable summaries of the managers that we discuss, and we highlight how some of them use different technologies to perform similar tasks. While many experience managers remain undiscussed, we nonetheless demonstrate that distinct, well-defined components exist across a diverse set of managers, and that a modular method for building new managers might well be within reach.

1 Introduction

Recent years have seen an increasing demand for software systems that can improve the experiences of their users, in the contexts of both applications and video games. Such systems are designed to adapt to each user, providing a tailored experience based on the person's behaviour within the application or game. In an educational game, for example, such an adaptive system might adjust the pacing or order of the lessons that it presents, to increase the student's chances of remembering more later. Many names have been used to describe the task of influencing a system using AI techniques, but we prefer the term *experience management* (Riedl et al. 2008) because it generally describes the task of modifying a user's experience of an interactive system.

To tackle experience management problems, many AI managers have been created using a variety of AI techniques, including planning, curve fitting, filtering, and more. In spite of all this work, however, the field has lacked a critical component for ensuring methodical progress: a way to meaningfully compare the inner workings of different AI managers. In this paper, we claim that such comparisons can be made in the context of an existing conceptual framework, and we support this claim by analyzing and comparing the inner workings of four diverse managers.

2 Representing Experience Management

In the context of game-related research, researchers have often referred to experience management (EM) as *drama management* (DM). One of the first definitions can be traced back to Laurel (1986). She designed an AI system called the PLAYWRIGHT which, utilizing a designer-provided knowledge base and the history of the interactive environment, would generate a temporally ordered sequence that describes the next moments of the interactive simulation. Weyhrauch (1997) developed Laurel's work by designing an AI experience manager named MOE. With this work, he extended the PLAYWRIGHT's capabilities with two key features. First, he represented DM as an optimization problem, where the manager determines the quality of any given narrative using one or more measures or estimates, such as the player's level of engagement. Second, he included the idea that a manager might want to change not only the immediate course of the story, but also its behaviour at some potential future states. Weyhrauch considered DM similarly to game-tree search: a player performs an action within the environment, and the drama manager responds with some moves by performing a game-tree-like search to maximize an evaluation function. The evaluation function was the encoding of the authors' belief of what an excellent interactive experience looks like, and it provided the drama manager with objectives to achieve. Weyhrauch's work led to a representation of EM called Search-Based Drama Management (SBDM) (Nelson et al. 2006; Nelson and Mateas 2008). More generally, drama management usually entails an omniscient AI system that monitors the fictional world and influences what happens next, following authorial constraints (Roberts and Isbell 2008; Yu and Riedl 2013).

Other possible ways to represent EM can be found in prior work on *dynamic difficulty adjustment* (Hunicke and Chapman 2004), *adaptive game mechanics* (Lindley and Sennersten 2006), *player-adaptive games* (Ha et al. 2011), *procedural game adaptation* (Thue and Bulitko 2012), and *generalized experience management* (Thue 2015). "Experience management" can be seen as the hypernym of these terms, since it generally involves an AI system that manages user experiences. For a comparison of different EM representations, see recent work by Thue and Bulitko (2018).

3 Conceptual Framework

Our goal in this paper is to provide an overview and discussion of several computational techniques that have been studied and used in the context of EM research. To structure our discussion, we will use Thue's (2015) "Generalized Experience Management" (GEM) framework, as its generality was demonstrated across several different managers.

GEM defines experience management as the task of optimizing a player's experience in an interactive environment by adjusting that environment while the player is experiencing it. GEM is a framework in the sense that it combines several conventional notions of EM into a well-defined formal structure, providing both a base and a collection of conceptual "building blocks".[1]

The base of GEM can be summarized as follows. Given an interactive environment (e.g., a computer game), a player's experience (called a *trajectory*) is a rotating sequence of the game's *states* (which the player perceives), *actions* (which the player performs), and potentially new variations of the game's *mechanics* (which determine future states). A gameplay *history* is a trajectory that starts from the beginning of the experience and ends at the most recent action the player performed. GEM allows an experience manager to modify a game's mechanics, which makes it simpler to represent some managers using the GEM framework (Thue 2015).

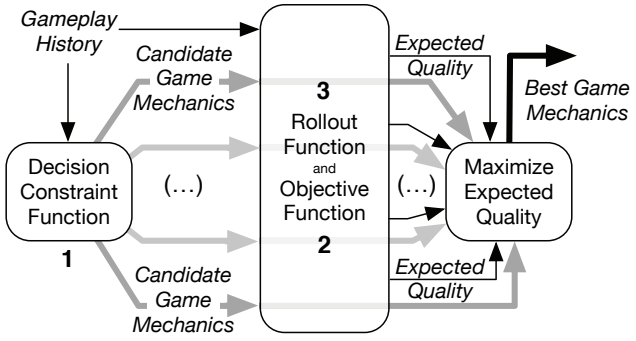

Figure 1: A schematic diagram of a GEM manager's policy. Numbers identify GEM's building blocks. Rounded boxes show functions, italics show data, and arrows show function inputs and outputs. The flow of game mechanics is highlighted with thicker arrows. See Figure 2 for more details.

As sketched in Figure 1, a GEM manager's policy works to maximize the *expected quality* of the player's experience by assessing several candidates for the game's mechanics and choosing the best one. Figure 2 shows the assessment step in more detail, in which the potential futures that could result from each given candidate for the game's mechanics are used to compute an expected quality. Quality is highest when the experience's effect on the player is closest to what the manager's designers intended.

GEM's building blocks form the basis of our forthcoming discussion, and we review their definitions briefly here:

[1]To keep focus on the relevant *concepts* of GEM, we will not use GEM's mathematical definitions in this work.

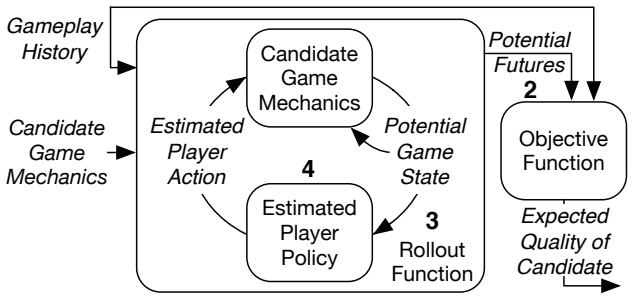

Figure 2: A schematic of how GEM's rollout function, objective function, and estimated player policy are used to assess candidate game mechanics (see Figure 1 for context).

- **Block #1 - Decision Constraint Function**: This allows the manager's designer(s) to prevent it from considering different options for changing a game's mechanics. For example, *Left 4 Dead*'s AI DIRECTOR was constrained to choose between only two versions of the game's mechanics (spawning more and more zombies, and not) (Valve Corporation 2008; Booth 2009). Decision constraints can reduce the computational requirements of EM by decreasing the number of possible experiences that the manager needs to analyze.
- **Block #2 - Objective Function**: When more than one version of the game's mechanics are available for the manager to consider, the manager might consult a designer-provided objective function. This function estimates the quality of a given trajectory, and can thus be used by the manager to estimate the results of its modifications. For example, *Façade*'s drama manager estimated how closely the player's current experience of dramatic tension was following an author-defined curve over time (Mateas 2002; Mateas and Stern 2005).
- **Block #3 - Rollout Function**: To obtain more useful estimates from an available objective function, a manager might use a rollout function to estimate potential futures of the player's experience (for one or many steps). Weyhrauch (1997) illustrated this block with the drama manager MOE, which aimed to build a tree of possible futures to consider when selecting its next dramatic move.
- **Block #4 - Estimated Player Policy**: To increase the reliability of an available rollout function, the manager might use an estimated player policy to estimate which action(s) the player might perform next, given their prior gameplay history. An example of this block can be found in a study by Min et al. (2016), where they attempted to model how players would form and pursue new goals based on their prior experiences in the game.
- **Block #5 - Feature Vector**: Given a trajectory of a player's prior (or potential future) experience in the game, it is common for managers to extract higher-level information that can aid in their reasoning process. A feature vector is a collection of functions, each of which is responsible for extracting one piece of information from a given trajectory. For example, Barber and Ku-

denko (2007) used the player's gameplay history to estimate several features in the form of a player model, including the player's selfishness and faithfulness.

Figures 1 and 2 show schematic diagrams of the GEM framework in terms of how a GEM manager's policy might operate. Given a gameplay history as input (Figure 1), the policy first uses the decision constraint function (Block #1) to obtain a set of possible candidates for game mechanics. Next, each candidate is sent to the rollout (#3) and objective (#2) functions for assessment, along with the gameplay history (Figure 2). Given a gameplay history and a candidate for the game's mechanics, the rollout function generates a set of potential futures; it uses the estimated player policy (#4) to estimate subsequent player actions, and the candidate game mechanics (now as a function, rather than data) to compute subsequent game states. Alongside the player's gameplay history, the set of potential futures is assessed by the objective function (#2) to calculate the expected quality that will result from choosing the given candidate game mechanics. If a manager is created without an estimated player policy, then the rollout function will be limited to producing potential futures that only extend one state past the given gameplay history. If a manager is created without a rollout function, then the objective function will estimate the candidate's expected quality using only the gameplay history. After obtaining an expected quality for each candidate game mechanics (Figure 1), the manager chooses a candidate that maximizes the expected quality and applies it to the game.

Block #5 (a feature vector) is left out of the figures to simplify their presentation. In practice, for any function that accepts a trajectory (such as a gameplay history or a potential future) as one of its inputs, a feature vector can be used when computing the function's result.

All of GEM's building blocks are complementary, and they can be used to categorize the techniques that various researchers have developed in pursuit of better experience managers. We do so in the sections that follow.

4 Managers and Techniques

In our review of the literature, we have found several different techniques that have been used to address specific challenges in one or more of GEM's building blocks. We have also found several experience managers that have combined some of the blocks. We begin by summarizing the techniques used by each manager in turn (Section 4), and then follow by discussing their similarities and differences in the context of each of GEM's blocks (Section 5).

***Façade*'s Drama Manager.** We begin with Mateas and Stern's (2002; 2005) work on *Façade*, an interactive drama that simulates being on stage with two live actors who are driven to cause tense situations to arise. During an experience in *Façade*, an AI manager dynamically determines a sequence of dramatic situations (called "beats") between two virtual actors and the player. Before execution, the designer provides an estimate of how much tension each beat will add to the story once executed, and they also provide a desired curve of tension versus time. Whenever the manager gets to choose a subsequent beat, the set of available candidates are first restricted by preconditions that designers defined for each beat. Since each beat defines an interactive segment of gameplay, choosing between them can be modeled as changing a game's mechanics (Thue 2015). *Façade*'s beat preconditions are thus an example of a decision constraint function (Block #1). *Façade*'s manager chooses between beats to minimize the distance between the given curve and an estimate of the amount of tension that is currently in the story; the inverse of this distance defines *Façade*'s notion of quality, and estimating it is an example of using an objective function (Block #2). The estimate of the story's current tension level is computed from the sequence of beats that have occurred thus far (i.e., the game's history), and it is thus an example of a feature (Block #5). *Façade*'s manager uses no estimate of the player's current policy (Block #4), but its consideration of potential future beats can be well thought of as the computation of a rollout function (Block #3).

Narrative Mediation in Mimesis. Riedl, Saretto, and Young (2003) described a technique to manage the interaction between a player and virtual actors, toward advancing author-given objectives without the player noticing. In particular, they aimed to detect and respond to unplanned user actions (a process called narrative mediation) within Mimesis (Young 2001), an architecture for intelligent interactive narrative worlds. Mimesis used an AI planner to generate narrative plans comprising the actions that the player and the actors should or would perform.

Before each play session, the experience manager analyzed the causal structure of the narrative plan to find opportunities for any exceptions to arise (due to player actions). For each exception, it produced a new plan to recover from it while preserving the goals of the original story. Generating a new plan in this way is an example of a rollout function (Block #3), as it examines potential futures of the game.

To allow plans to be generated, one must create a *planning domain*, which restricts the set of plans that can be found by assigning logical preconditions to each of the actions that could become part of a plan. Creating such a domain is one way to specify a decision constraint function (Block #1), since a plan's execution is analogous to the execution of a particular version of a game's mechanics. Riedl, Saretto, and Young also defined decision constraints in another way: for each exceptional player action, a generated plan was compared to a set of alternative, pre-authored ways to respond to the exceptional player action. This pre-authoring is an example of using decision constraints because each response was implemented as a small variation to the game's mechanics (e.g., causing a gun to misfire instead of shoot).

The comparison between the plan and the pre-authored alternatives was performed using a function that estimated the quality of the plan and each alternative (i.e., an objective function; Block #2). Although the manager considered possible player actions in its rollouts, its search was exhaustive rather than guided by any estimates of what the player would do. Thus it did not use an estimated player policy (Block #4). The authors do not describe any features (Block #5) that they may have used, for example, to compute their manager's objective function.

Manager	Quality	Decision Constraint Func. Block #1	Objective Function Block #2	Rollout Function Block #3	Estimated Player Policy Block #4	Feature Vector Block #5
Façade's Drama Manager	Managed Tension	Preconditions on Events	Minimize distance between estimated value and given curve.	Consider beats just-in-time.	N/A	Computed from annotations on beats.
Narrative Mediation in Mimesis	Good Story Structure	Planning Domain & Hand-authored Alternatives	Assess narrative structure relative to authored goals.	Consider actions in advance.	N/A	N/A
The Personalized Drama Manager	Expected Enjoyment	Hand-authored Alternatives	Estimate expected value using player model.	Consider plot points just-in-time.	Collaborative Filtering (CF) based on player preferences.	Computed using Prefix-based CF.
Player-specific Automated Storytelling	Perceived Agency	Planning Domain	Maximize similarity between content and player model.	Consider actions just-in-time.	Assumes no action other than observing.	Computed from annotations on actions.

Table 1: A summary of four experience managers, divided by GEM's building blocks.

The Personalized Drama Manager. Yu and Riedl (2015) built an EM system that aimed to influence players into choosing actions that would lead to a narrative experience that would maximize their expected enjoyment, according to their tastes and preferences. Their approach can be summarized as follows. They began with a story graph in which multiple player actions could lead to the same plot point[2]. Next, the PERSONALIZED DRAMA MANAGER (PDM) built a model of the player's preferences over potential gameplay trajectories using "prefix-based collaborative filtering" (Yu and Riedl 2012); these preferences represent values of one or more features (Block #5). The PDM used this model in an objective function (Block #2) to estimate how much the player would enjoy different potential experiences. The PDM also built a model of the player's preferences over different potential actions. It used this model to predict which action the player would likely choose at each potential state (Block #4), and to calculate the probability that the player would reach an ending of the story. This consideration of potential futures amounts to the use of a rollout function (Block #3). To facilitate the future with the highest expected enjoyment, the PDM altered the game's mechanics to constrain the set of actions that would be presented to the player. Its choices for these constraints were determined by the given story graph, which constitutes an example of a decision constraint function (Block #1).

Player-specific Automated Storytelling. The final manager that we consider is Ramirez and Bulitko's (2015) PAST (Player-specific Automated Storytelling). PAST aimed to perform narrative mediation similarly to Riedl, Saretto, and Young (2003)'s work in Mimesis, but with addition of a learned player model inspired by Thue et al.'s (2007) PASSAGE. When the player performed an exceptional action, PAST used a modified version of the AUTOMATED STORY DIRECTOR'S planner (Riedl and Stern 2006) to generate a new story plan that was tailored to the player model. Given this plan, PAST then modified the game's mechanics to cause the plan to come to pass. PAST's planning process can be viewed as an example of the manager policy shown in Figure 1. First, based on the gameplay history, a hand-made function was used to identify exceptional player actions and give PAST's planner access to a planning domain (Block #1), similarly to Riedl, Saretto, and Young (2003)'s approach discussed above. This domain defined a set of possible plans including *partial* plans (i.e., those that did not reach a game ending), which PAST explored as part of its search through potential possible futures (Block #3). By analyzing partial potential futures with a heuristic function (i.e., an objective function; Block #2), PAST was able to prioritize its review of the remaining candidates, finding solutions with acceptable quality in a smaller amount of time. Since PAST's player model could not predict the player's actions, its planning process used the simplifying assumption that the player would perform no actions other than observing what occurred (Block #4). The player model itself, however, was made of five features (Block #5) that estimated the player's inclinations toward playing in different styles (e.g., fighting or conversing). PAST's heuristic function was based on that of PASSAGE (Thue et al. 2007), which sought to maximize the similarity between the potential future's supported play styles and values in the player model.

Summary. Table 1 summarizes our analysis of the managers that we considered in this section. For each manager, we state its notion of quality along with the method that it used (if any) to implement each of GEM's building blocks.

5 Comparison via GEM Building Blocks

While the previous section introduced different experience managers in the context of GEM's building blocks, this section considers the similarities and differences that can be found by analyzing them block by block.

5.1 Decision Constraint Functions

The purpose of the decision constraint function is to simplify the work of the manager by reducing the alternatives that are available for it to choose following different player histories. It also serves as a primary way for designers to encode their intentions in the manager's inputs, and it can include both

[2]Using Thue and Carstensdottir's (2018) disambiguation, these are *character-focused* plot points – sections of gameplay that potentially involve both player and non-player actions.

plot and not-plot constraints. From our analysis of the managers in Section 4, we found two main techniques that can be used to implement a decision constraint function:

- the designer hand-authors an explicit mapping between histories and sets of candidates for the manager to consider (as the Mimesis manager and the PDM received);
- the designer specifies a set of preconditions that can be applied to each history to identify which candidates should be considered; the Mimesis manager and PAST received preconditions embedded in a planning domain, while *Façade's* DM had them attached to each story beat.

A general distinction can be drawn between the two techniques. While the first specifies its candidates for new game mechanics *explicitly*, the second does so *implicitly* – some additional computation is required to obtain the required set of candidates, such as evaluating one or more sets of preconditions against the current history. Explicit specification offers designers the benefit of having simple and direct control over what the manager might do, but it comes at a cost; the designer's required specification work increases linearly with the number of opportunities that the manager receives to influence the game. Meanwhile, implicit specification offers at least the potential for a better work to output ratio, because the relationship between required work and manager opportunities is super-linear[3]. For example, a single precondition could be used to map an enormous number of histories to a particular set of candidates for the game's mechanics. So long as the added complexity of producing preconditions (versus explicit candidates) remains low, implicit specification should offer more authorial leverage (Chen, Nelson, and Mateas 2009) than explicit specification. Anecdotally, we find it interesting that Riedl, Saretto, and Young's (2003) manager for Mimesis used both implicit and explicit techniques, despite it being one of the earliest, narrative-focused managers that were implemented.

5.2 Objective Functions

An objective function allows the designer to specify how the manager should assess different candidate game mechanics in relation to the gameplay history and its potential futures. By considering the managers discussed in Section 4, we found several different ways to implement this block.

Mateas and Stern's (2005)'s DM for *Façade* worked to minimize the distance between a curve-over-time provided by the designer and the DM's estimate of the value that was meant to track the curve. This value was related to the story's tension level in *Façade*, but the same method could be used to specify and pursue curves for other aspects of the player's experience (e.g., their level of surprise over time (Bae and Young 2008)). Instead of considering any direct effects on the player (like tension or surprise), Riedl, Saretto, and Young's (2003) manager for Mimesis assessed potential player experiences from a structural perspective. This type of quality assessment follows closely from Weyhrauch's (1997) work on MOE, and can also be found in work by Nelson et al. (2006).

Yu and Riedl's (2015) PDM worked to compute the expected value of each of several potential futures, each of which began with a set of customized actions that the PDM would present to the player. To perform this computation, it combined models of the player's preferences and policy to assess the quality and probability of each trajectory that could be reached.

Ramirez and Bulitko's (2015) PAST computed its objective function using a similarity metric driven by its model of the player's preferences, and this method has been used in at least two other managers (Thue et al. 2007; 2011). Perhaps more interestingly, PAST *used* its objective function unlike any other manager that we considered in this work – as a heuristic function inside a planner. Doing so allowed it to optimize the order in which it considered the available candidates for new mechanics. This method allowed PAST to fully generate only a small number of plans, speeding its search for one that best fit its current player.

When considering the previous methods together, two distinctions can be drawn. The first concerns the nature of the data that the objective function considers: Does it concern the experience's structure (as the functions for *Façade* and Mimesis did), or does it concern direct effects on the player (as the functions for *Facade*, the PDM, and PAST did)? Structural assessments offer a more convenient way to perform evaluations, since running user studies with real players might not be required (Nelson et al. 2006). Unfortunately, the results of such studies suffer from difficulties with generalization – especially when having a direct effect on players is the manager's ultimate goal. Nevertheless, at least some structural characteristics have been shown to be well associated with particular effects on players (Bae and Young 2008). The second distinction concerns the source of the data that the objective function receives: Does it come from authored annotations (like the data for *Façade* and PAST), or does it come from a machine-learned source (like the data for the PDM)? Authored annotations have similar benefits and limitations as explicit decision constraints; they are simple to specify and offer absolute control, but their associated authorial leverage is low. Using a machine-learned source of data offers the potential advantage of transferring learned information between experiences or games, although the costs might include less predictable manager behaviour and more complicated troubleshooting. In general, objective functions are often created with a specific notion of quality in mind. However, even if a manager uses a particular technique to pursue its defined quality, that same method (perhaps with some adjustment) is likely to be useful in pursuing other notions of quality as well.

5.3 Rollout Functions

The rollout function defines the mechanism that a manager uses to examine potential futures of the player experience, to improve the estimate of the objective function. When a forward model of the environment is available, the manager needs only to apply the effects of any given action to advance the state of the world. However, this process can be compu-

[3]We use "super-linear" to describe a function that eventually grows faster than any linear one.

tationally expensive, particularly if the manager's lookahead is long or the rollouts need to be generated in real time. The managers that we analyzed in this paper offered a variety of solutions to these problems, and they can be distinguished along two orthogonal dimensions:

- the time at which the computation occurs (before the experience, online before it is needed, or just-in-time);
- the granularity with which each potential future is represented; player and character actions are highly granular, while a collection of such actions is less granular, when it can be considered as a unit.

In *Façade*, using the low granularity of a "beat" (a segment of gameplay covering various states and actions) allowed the DM's rollout function to be computed just-in-time at a relatively low cost, since the total number of beats in any story was not large.

Riedl, Saretto, and Young's manager for Mimesis computed a variety of possible futures in terms of plans that included player and character actions – its representation was highly granular. However, its computations occurred both before the experience started (for all potential exceptions that could occur in the given original story), as well as online (for any exceptions to a repaired story). These online computations were distributed across times of low processor activity in advance of any exception occurring, so that the results would be ready as soon as they were needed.

The rollout function used by Yu and Riedl's (2015) PDM is more similar to that of *Façade*'s DM than it is to Riedl's prior work on Mimesis, in that the PDM used a granularity that is relatively low: character-focused plot points. However, one difference between the PDM and *Façade*'s DM is that the PDM's rollout function generated complete future experiences, while *Façade*'s DM only considered futures that extended to a single subsequent beat.

Similarly to Riedl, Saretto, and Young's manager for Mimesis, PAST considered potential futures at the granularity of character actions. However, instead of precomputing these potential futures, PAST computed them just-in-time. This difference was necessitated by PAST's use of a learned player model, because the number of possible configurations of the model made it intractable to exhaustively compute all possible futures in advance.

5.4 Estimated Player Policies

An estimated player policy allows the manager to model a player's behaviour and thus examine potential futures in a more player-focused way. The main goal of using this function is to predict the probability with which each possible future can occur. The primary difference between the two estimated player policies that we considered (the PDM's and PAST's) illustrates a common concern that arises when models of player preferences are used: When the data available to learn the model is sparse, it can be unwise to place too much confidence in the model's predictive power. Ramirez and Bulitko (2015)'s sparse player data led them to only use their model for assessing quality (and *not* for predicting player actions); they relied on an assumption of inaction (only observation) instead. Meanwhile, Yu and Riedl (2015) used Collaborative Filtering to extend the usefulness of the player-specific data that they could gather, and were able to obtain useful actions predictions.

Neither of the managers for *Façade* and Mimesis used an estimated player policy, but there seems to be nothing about their designs that would prevent one from being used. Doing so could potentially allow the Mimesis manager to handle larger story plans (by avoiding an exhaustive consideration of actions) and *Façade*'s DM might be able to choose beats more effectively by looking farther into the future.

5.5 Feature Vectors

A feature vector allows the designer to identify and compactly represent patterns in trajectories that can simplify the specification and computation of other functions. The managers that we analyzed differed with respect to how they implemented their feature vectors; two managers used simple algebra and hand-authored annotations (*Façade*'s DM and PAST), and one used a machine-learned approach (the PDM). We discussed the benefits and limitations of hand-authoring versus machine learning in Section 5.2. Although Riedl, Saretto, and Young (2003) did not describe any features that were used by their manager, a variety of features pertaining to narrative structure can be found in related work (Weyhrauch 1997; Nelson et al. 2006).

Conclusions and Future Work

In this paper, we considered the field of Experience Management and some of its related concerns. In doing so, we have made the following contributions. First, we presented a focused, conceptual summary of Thue's (2015) *Generalized Experience Management* framework, which was previously only available in an extended and mathematical format. Second, we used this framework to identify common, comparable elements for four diverse experience managers, which together span much of the history of the field. Third, we used the identified elements to directly compare the different managers' designs, and these comparisons led to a variety of insights concerning how the different building blocks of a manager can be made. To the best of our knowledge, this work represents the first direct comparison between the internal designs of different experience managers. As a result, we hope that it will be considered by others in the field as a proof of existence: comparisons between the designs of managers can be made, and they can offer useful results.

Looking forward, it should be desirable to perform additional analyses like the four that we presented in this paper – both on the variety of other managers that already exist, and on *new* managers as they are created. By including a similar analysis of a new manager as part of its publication, authors will immediately gain a context in which they can meaningfully compare their new work to the managers that came before. Furthermore, given the modularity of the GEM framework, it might serve as a useful guide for building new managers that benefit directly from the techniques that other managers have used. This was demonstrated once by Ramirez and Bulitko (2015)'s construction of PAST using a precursor to GEM (Thue and Bulitko 2012), and we hope that more examples will follow.

References

Bae, B.-C., and Young, R. 2008. A use of flashback and foreshadowing for surprise arousal in narrative using a plan-based approach. In *1st International Conference on Interactive Digital Storytelling*, 156–167. Erfurt, DE: Springer.

Barber, H., and Kudenko, D. 2007. Dynamic generation of dilemma-based interactive narratives. In *3rd AI and Interactive Digital Entertainment Conference (AIIDE 2007)*, 2–7. Palo Alto, California: AAAI Press.

Booth, M. 2009. The AI systems of Left 4 Dead. Presentation at the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2009).

Chen, S.; Nelson, M. J.; and Mateas, M. 2009. Evaluating the authorial leverage of drama management. In *The 5th AI and Interactive Digital Entertainment Conference (AIIDE)*, 136–141. Palo Alto, California: AAAI Press.

Ha, E. Y.; Rowe, J. P.; Mott, B. W.; and Lester, J. C. 2011. Goal recognition with markov logic networks for player-adaptive games. In *7th AAAI Conference on AI and Interactive Digital Entertainment*, AIIDE'11, 32–39. AAAI Press.

Hunicke, R., and Chapman, V. 2004. AI for dynamic difficulty adjustment in games. In *Challenges in Game AI Workshop, Nineteenth National Conference on Artificial Intelligence*, 91–96. San Jose, California: AAAI Press.

Laurel, B. K. 1986. *Toward the Design of a Computer-Based Interactive Fantasy System*. Ph.D. Dissertation, Ohio State University.

Lindley, C. A., and Sennersten, C. C. 2006. Game play schemas: From player analysis to adaptive game mechanics. In *International Conference on Game Research and Development*, CyberGames '06, 47–53. Murdoch University.

Mateas, M., and Stern, A. 2005. Procedural authorship: A case-study of the interactive drama Façade. In *Digital Arts and Culture (DAC)*.

Mateas, M. 2002. *Interactive Drama, Art, and Artificial Intelligence*. Ph.D. Dissertation, Carnegie Mellon University.

Min, W.; Mott, B.; Rowe, J.; Liu, B.; and Lester, J. 2016. Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory. In *25th International Joint Conference on AI*, 2590–2596.

Nelson, M. J., and Mateas, M. 2008. Another Look at Search-Based Drama Management. In *23rd Conference on Artificial Intelligence*, 792–797. Chicago, IL: AAAI Press.

Nelson, M. J.; Roberts, D. L.; Isbell, Jr., C. L.; and Mateas, M. 2006. Reinforcement learning for declarative optimization-based drama management. In *5th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, 775–782. New York, NY, USA: ACM.

Ramirez, A., and Bulitko, V. 2015. Automated Planning and Player Modeling for Interactive Storytelling. *Transactions on Computational Intelligence and AI in Games* 375–386.

Riedl, M. O., and Stern, A. 2006. Believable agents and intelligent story adaptation for interactive storytelling. In *3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, 1–12. Darmstad, DE: Springer.

Riedl, M. O.; Stern, A.; Dini, D.; and Alderman, J. 2008. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning* 4(2):23–42.

Riedl, M.; Saretto, C. J.; and Young, R. M. 2003. Managing interaction between users and agents in a multi-agent storytelling environment. In *2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, 741–748. New York, NY, USA: ACM.

Roberts, D. L., and Isbell, C. L. 2008. A survey and qualitative analysis of recent advances in drama management. *International Transactions on Systems Science and Applications - ITSSA* 4(2):61–75.

Thue, D., and Bulitko, V. 2012. Procedural Game Adaptation: Framing Experience Management as Changing an MDP. In *5th Workshop on Intelligent Narrative Technologies*, 44–50. Palo Alto, California: AAAI Press.

Thue, D., and Bulitko, V. 2018. Toward a Unified Understanding of Experience Management. In *Proceedings of the 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'18)*. AAAI Press.

Thue, D., and Carstensdottir, E. 2018. Getting to the point: Toward resolving ambiguity in intelligent narrative technologies. In *Workshops on Intelligent Narrative Technologies and Intelligent Cinematography and Editing*, 9. CEUR.

Thue, D.; Bulitko, V.; Spetch, M.; and Wasylishen, E. 2007. Interactive storytelling: A player modelling approach. In *3rd AI and Interactive Digital Entertainment Conference (AIIDE 2007)*, 43–48. Palo Alto, California: AAAI Press.

Thue, D.; Bulitko, V.; Spetch, M.; and Romanuik, T. 2011. A computational model of perceived agency in video games. In *AI and Interactive Digital Entertainment Conference (AIIDE)*, 91–96. Palo Alto, California, USA: AAAI Press.

Thue, D. 2015. *Generalized Experience Management*. Ph.D. Dissertation, University of Alberta, Canada.

Valve Corporation. 2008. Left 4 Dead. www.l4d.com.

Weyhrauch, P. 1997. *Guiding interactive drama*. Ph.D. Dissertation, Canergie Mellon University, Pittsburgh, PA.

Young, R. M. 2001. An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment. In *Notes of the AAAI Spring Symposium on AI and Interactive Entertainment*, 77–81.

Yu, H., and Riedl, M. O. 2012. A sequential recommendation approach for interactive personalized story generation. In *11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, 71–78. Richland, SC: International Foundation for AAMAS.

Yu, H., and Riedl, M. O. 2013. Data-driven personalized drama management. In *9th AI and Interactive Digital Entertainment Conference (AIIDE'13)*, 191–197. AAAI Press.

Yu, H., and Riedl, M. O. 2015. Optimizing Players' Expected Enjoyment in Interactive Stories. In *11th AI and Interactive Digital Entertainment Conference (AIIDE'15)*, 100–106. AAAI Press.