Understanding Player Patterns by Combining Knowledge-Based Data Abstraction with Interactive Visualization

Nithesh Javvaji
javvaji.n@northeastern.edu
Northeastern University
Boston, USA

Casper Harteveld
c.harteveld@northeastern.edu
Northeastern University
Boston, USA

Magy Seif El-Nasr
mseifeln@ucsc.edu
University of California Santa Cruz
Santa Cruz, USA

ABSTRACT

Digital games are often created with large virtual worlds and a high number of degrees of freedom for players. For analyzing patterns of play, it is imperative to consider all features affecting the gameplay, which leads to an explosion in data space. To address this problem, we propose an approach for knowledge-based data abstraction – inspired by applications in medicine – that uses interactive visualizations based on game telemetry data to study player patterns. The approach involves iterative knowledge-based abstraction of elements of player action sequences to condense the data space to a level interpretable by game designers and user researchers. We developed this approach as part of developing a puzzle game using an existing interactive visualization tool, and demonstrate its value for understanding this game's player patterns. Based on the lessons learned from this study, we present a general set of guidelines for knowledge-based data abstraction for other game genres and interactive visualization systems.

CCS CONCEPTS

• **Human-centered computing** → **Visualization systems and tools**; • **Applied computing** → **Computer games**.

KEYWORDS

Games User Research; Visual Game Analytics; Interactive Visualization; Game Design; Data Abstraction; Knowledge Representation

ACM Reference Format:
Nithesh Javvaji, Casper Harteveld, and Magy Seif El-Nasr. 2020. Understanding Player Patterns by Combining Knowledge-Based Data Abstraction with Interactive Visualization. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '20), November 2–4, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3410404.3414257

1 INTRODUCTION

Digital games are dynamic systems containing many obstacles and challenges for players to overcome, individually or in teams. Understanding how players solve problems in these systems can enhance the game design process [10]. By understanding how players learn the game mechanics, how they use these mechanics to overcome obstacles, and where they struggle to apply the mechanics correctly, designers can learn if their design is working as intended and whether there are significant flaws in the player experience [51]. The utility in investigating player patterns (i.e., player actions, behaviors, and progression) also extends further, to e.g. the development of game AI, personalized non-player characters, and game guides. As such, understanding player patterns can lead to developing better games that address wider target audiences [11, 29, 47].

However, games are often created with large virtual worlds and a high number of degrees of freedom for players. The volume, variety, dimensionality, and dynamic nature of this data makes understanding player behavior and making sense of player strategies increasingly complex [52]. These issues will likely only tend to aggravate in the near future, given advances in data collection: Datasets become increasingly sparse with more expansive, granular information collection and as all players do not cover the entire game space. Recent research has attempted to address this problem using techniques from statistics, data mining, and machine learning to model player behavior, strategies, and skills from large data sets [5, 8, 39]. Although these studies are able to predict player strategies, profile player types, and model player behaviors, they come with two caveats. The first caveat is interpretability (in terms of interpreting the data): Results from these studies are often non-interpretable to non-experts, and even in some cases to experts in the field. Approaches such as human-centered machine learning [44, 45] might help here, as they provide transparency and interpretability, and encourage non-experts to pursue these techniques. The second caveat is transparency (in terms of player patterns): The aforementioned methods and techniques result in algorithms, clusters, classifications, and predictions; however, these do not immediately illuminate how players actually play the game, which will be useful for game designers to understand player patterns.

The issues of interpretability and transparency have always been central to the game analytics and visualization community [16]. As such, a large number of visualization tools exist to view, interact with, and interpret player action sequences [53]. Different visualization systems have been developed over the years based on e.g. heatmaps [22], sankey diagrams [55], action sequences [2, 33, 54], and spatio-temporal data [9, 52]. Some of these are game and genre specific and are thus not generalizable to other game genres or non-game contexts. But the main disadvantage with most of these systems is their inability to handle high-dimensional data (i.e., large number of attributes in a dataset), which makes these systems incomprehensible for humans and, as a result, interactive visualizations have not been widely utilized in the game industry [3].

CHI PLAY '20, November 2–4, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8074-4/20/11…$15.00
https://doi.org/10.1145/3410404.3414257

Historically, researchers [46] used techniques such as classical multidimensional scaling (CMDS), an exploratory technique to visualize similarities of high-dimensional data in a low-dimensional space (2D, 3D), to address the issue of high dimensionality. Node-link representations used CMDS for abstracting high-dimensional data which cannot be visualized in spatial relationship to the virtual environment [2, 26, 52]. Although there are variety of visualization tools available, a gap exists for methods to abstract high-dimensional data to understand player behavior (e.g., by visualizing player traces) and discover patterns of what players are doing over time (e.g., by analyzing occurrence, context and frequency), particularly for non-spatial data such as puzzle games.

This paper's goal is to augment the current use of visualization and game analytics in design by providing a general set of guidelines to abstract player traces from raw data and demonstrate this in the context of puzzle games. Abstraction is crucial in leveraging play traces to gain knowledge because of the high dimensionality problem of game data. While our work is focused on puzzle games, it is critical to look into generalizable methods for abstracting play traces for the field to progress and unlike other disciplines like medicine who have already established a set practice for doing this [31, 41], game analytics has yet to develop any guidelines.

Inspired by the medical field [31, 41], we use *knowledge-based data abstraction*, which involves abstracting the low-level game data based on expert knowledge, feeding the resulting abstraction rules into a script to automatically abstract the data, and then visualizing the abstractions with *interactive visualization* (i.e., we used *Glyph* [33]) for analysis and iteration of the abstraction rules. We define knowledge-based abstraction in the context of games as: the process of consolidation of game state-space with the help of an expert (with knowledge about the design of the game and analysis objectives) to make the visualization more interpretable.

Our proposed approach further involves identifying the *ideal trace*, inspired from Horn et al. [23], which is what we expect an individual or team to follow if they are aiming to achieve the optimal outcome. Having such ideal trace allows to objectively measure the difference between intended and actual player pattern. Using a distance measure (i.e., Time Warp Edit Distance), we then compare traces of players with this ideal trace to capture the variability among players and find the different player patterns.

We developed this method as part of developing *Daedalus* [21], an interactive puzzle game where players work in teams to solve puzzles. Our aim was to find the patterns of play in solving puzzles, and to identify any unexpected player paths. Another objective is to identify puzzles that are engaging (in terms of multiple failed attempts leading to eventual success), frustrating (failed attempts leading to eventual withdrawal) and relatively easy to solve (none to few failed attempts leading to success). In this paper we present our method and demonstrate how we applied it in the context of developing this game to discover any flow, balancing, and design problems. Based on our lessons learned, we present a general set of guidelines for knowledge-based data abstraction for other game genres and interactive visualization systems. Our contributions are thus: (1) A method that combines knowledge-based abstraction with interactive visualization to analyze player patterns; and (2) a general set of guidelines to abstract low-level game log data using expert knowledge for human interpretable game analysis.

2 RELATED WORK

In this section, we first discuss the work on analyzing player behavior using data mining and machine learning approaches. Then we focus on different visualization techniques, particularly spatio-temporal approaches and, finally, we move on to data abstraction, which makes the visualization systems more interpretable.

2.1 Player Behavior Modeling

Historically, designing games and AI in games are guided through behavior profiling [12, 15, 17, 43, 59]. Such profiling based on play traces has been of enormous interest in recent times [16, 30, 35] using techniques from the fields of statistics, data mining, and machine learning. Sometimes player telemetry data is combined with data from chat services, social networks, demographic information, which reveal interesting relationships, such as a link between skill level and influence on other players [6, 42]. Sifa et al. [42] recently published a detailed review of behavior profiling of players based on telemetry data. This review shows several efforts based on clustering [5, 14] and classification [15, 36], which provide insight into player profiles and individual differences; however, they do not provide insight into the problem-solving behaviors by players.

In addition, increasing amount of work is being conducted on obtaining human-readable, interpretable rules for player performance prediction [37, 38, 56] and winning strategies [58], but these do not provide the granular details for players' interaction with game elements for gaining insights into game design issues. Moreover, much of the work in this area is limited to Real Time Strategy (RTS) and Multiplayer Online Battle Arena (MOBA) genres. Our current effort is aimed at contributing to an interpretable, transparent, and generalizable method for puzzle games that provide insights about game elements and mechanics.

2.2 Visualization Systems for Games

The number of available methods, tools, and techniques for analyzing telemetry data from games has grown rapidly in the past years, fueled by both academic and industry interests for utilizing player traces. There are many different applications, including (but not limited to) dynamic difficulty adjustment [24], game AI [59], monetization [56], measuring the stability of the software [60], and optimization [49]. Wallner et al. [53] provided a thorough review of visualization-based analysis of gameplay data. Here, we focus on the work that studied visualization of play traces to inform the design of games – also referred to as *design evaluation*.

In that light, Moura et al. [32] showed that while heat maps are effective for analysis, such as depicting positions of player deaths (cf. [11, 48]) and time spent in certain areas of the maps (cf. [4]), they are limited when analyzing the progression of players over time in the game. This issue with heat map-based visualizations has been addressed in subsequent work, as time is a crucial factor in almost all games. Andersen and Liu et al. [2, 26] first used this node-link representation of high-dimensional gameplay data that do not naturally map to two-dimensional spaces.

Identically, Gagne et al. [18] presented a visualization tool capable of including both space and time of player movement in an RTS game. Wallner and Kriglstein [54] proposed *PLATO*, a visual analytics system for multidimensional and temporal gameplay data

analysis, which utilizes graph algorithms based on their previous work [50, 52]. This interactive visualization system integrated techniques such as subgraph matching, path finding, and clustering, and demonstrated its utility in four different game domains. Similar to that, Nguyen et al. [33] proposed *Glyph*, a system we used because of its applicability to puzzle games. A detailed explanation of this system is provided in Section 3.2.

2.3 Data Abstraction

Abstraction of raw data based on metrics is the most common data abstraction practice, particularly in industry areas [16]. This is achieved through aggregation over time, or averages or summations on performance related metrics to develop variables of interest. For example, an aggregation such as mission success rate removes the time dimension during analysis. Similarly, kill/death ratio is a common feature/metric in shooter games. Other examples include number of conversations with PC/NPC characters, which could be used when analyzing the player's dependence on these characters.

However, fields like medicine have the practice of standard aggregations of variables based on expert knowledge or biological principles. Shahar et al. [31, 41] proposed a widely-known technique of knowledge-based temporal abstraction to aggregate raw data in specific time windows. Methods such as association rules, casual models, and classification tables are applied to temporal data over particular time windows to achieve abstraction. Standard practices like these are yet to be established for games. Recent models developed for churn prediction [13, 20, 27, 57] started making steps in this direction with metrics such as Monthly Active Users (MAU), Daily Active Users (DAU), Lift Time Value (LTU), and Conversion Rate (CR). But the metrics and churn prediction models do not provide insights into the game design elements leading to churn.

Almost all visualization systems [2, 26, 52, 54] for spatio-temporal data in RTS games discussed in Section 2.2 handled the issue of high-dimensionality data by mapping them onto 2D/3D spaces using classical multidimensional scaling (CMDS), a statistical technique to explore similarities in data. Wallner et al. [51] applied CMDS to analyze player behavior in an educational puzzle game. However, CMDS requires a dissimilarity matrix and unavailability or inappropriate dissimilarity function hinders the use of the method.

Canossa et al. [7] introduced a visualization tool called *G-Player* for exploratory analysis of multimodal data using heatmaps. The main advantage of this tool is that it lets users use Boolean operators to identify and label behaviors that can be used for abstraction of raw low-level data. Furthermore, its ability to connect to databases, define features, and filter using time and space makes the system unique. Building on this work, Ahmad et al. [1] proposed the Interactive Behavior Analytics Method (IBAM), which uses the two visualization systems *StratMapper* (built on *G-Player)* and *Glyph*. *StratMapper* allows human users to label high-level behaviors from low-level data, and feed this labeled data into *Glyph* to further condense and adjust the abstractions. They demonstrated the use of this method for modeling team and individual behaviors while maintaining the interpretablility of models, using data from two multiplayer team-based games, *BoomTown* and *DOTA 2*.

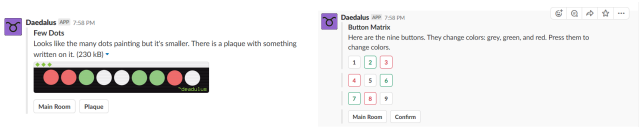
The work presented in this paper intends to eliminate the requirement for specific visualization systems for abstraction, by providing a general set of guidelines for expert knowledge-based abstraction of telemetry data. Such abstraction-based analysis enables users to identify feedback loops and problem-solving patterns, which can be relatively easily identified with a domain expert in the loop.

3 PROBLEM CONTEXT

We developed *Daedalus* to study adaptability in teams by investigating the problem-solving behaviors in players [21]. To ensure the game is achieving its purpose, we turned to game user research methods. While player interviews and observations are possible, they come with issues of subjective biases and poor scalability. So we turned to objective analysis through visualization of player traces, which comes with issues of interpretability and transparency. To get around these issues, we developed the proposed method of knowledge-based abstraction of player traces along with visualization. This work is resulting through the development of *Daedalus*. We also note that one cannot easily choose a game: with *Daedalus*, we had complete access to the game as well as the designers and the data, requirements that are needed for knowledge-based abstraction. Below we describe the game we developed, the visualization system used, and the dataset we obtained for our analysis.

3.1 Game: *Daedalus*

We collaborated with a game company to create *Daedalus*. It is a *Slack*-based digital escape room where players, as part of a team (3-5 players), attempt to complete all puzzles within a week. It revolves around a central narrative of a sentient AI testing players through a series of puzzles, in order to learn human problem-solving strategies so it can solve humanity's most pressing problems. There are five stages. Each stage contains a different set of interconnected puzzles, some of which involve traveling to a real-world location to find the answers. Players are allowed and encouraged to message one another via *Slack* to help each other figure out the solutions, and communicate about these once they have been found. Some of the puzzles are solvable by one individual, while others are designed to require coordination between the members of the team. All teams who fully completed *Daedalus* received a cash prize of $100 per team; the team that completed the game the fastest received a cash prize of $300. The latter prize was added to incentivize players to play *Daedalus* seriously and quickly, thus putting them under pressure to solve puzzles as fast as they could.



(a) Few dots screen **(b) Button matrix screen**

Figure 1: Two screenshots from *Daedalus*. Figure 1a shows the cue screen for the few dots puzzle and Figure 1b shows the keypad used to enter in the solution

Each puzzle provides a new challenge for players to overcome, and through that the need for situational tactics arises. Players are unsure what will come next, and perhaps, will be ill-equipped to deal with the situation. Those who are able to problem-solve in these

new contexts will be able to power through and complete the game. In each stage, *Daedalus* provides a large number of information sources, which we refer to as *cues*. Some cues provide the information necessary to solve one or more puzzles; other cues provide no meaningful information. Consequently, each puzzle requires the player to rethink what each cue means and re-contextualize it in response to new information. Thus, discontinuity in thinking and reconstructing the puzzle's representation are both required to progress. It is technically possible to "brute force" a solution in some of the puzzles, for example, by trying all possible combinations on the keypad shown in Figure 1b. However, this is unlikely to be successful: The high number of combinations makes this an infeasible tactic. For this reason, we argue that the puzzles in *Daedalus* require insight to solve.

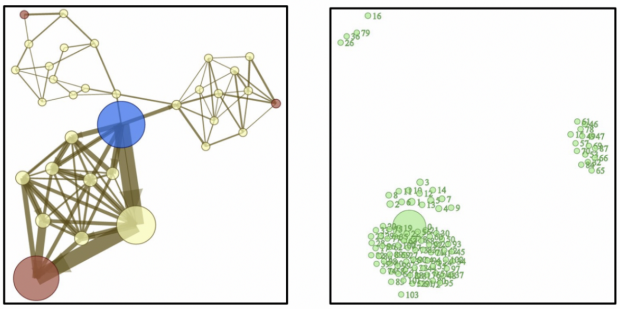
In this work, we will only consider the first stage of *Daedalus*' five stages, which consists of six puzzles: few dots, many dots, cypher wheel, safari slideshow, glyph painting, and safe. However, the approach can easily be expanded to the full game. There is a particular order in which these puzzles have to be solved to clear the first stage. Players can start by solving the few dots and/or safari slideshow puzzle first as they both are not dependent on other puzzles. The glyph painting puzzle can only be solved after the safari slideshow. The cypher wheel puzzle requires solving the few dots puzzle followed by the many dots puzzle. As a result, few dots, many dots, and cypher wheel can be considered a set and the other set would be safari slideshow and glyph painting. These sets can be solved in parallel. The final safe puzzle can only be solved after all five puzzles are solved. It is important to note that this order information is not presented to players.

3.2 Visualization System: *Glyph*

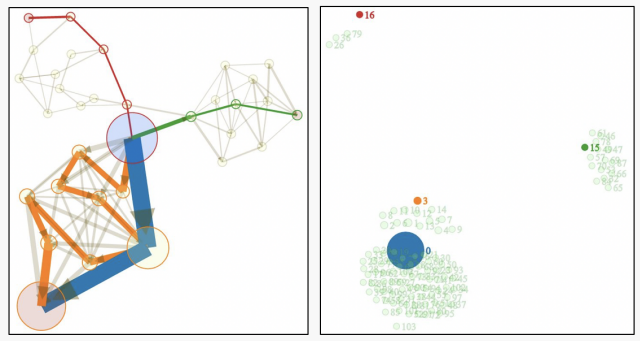
An essential component of our method is a visualization system called *Glyph* [33]. *Glyph* is used to understand players' decision-making strategies over time, by tracing their sequences of actions. The system allows us to visualize patterns from many players at the same time and thus compare how different players solved problems, adapted their strategies, or adjusted their course of action. Figure 2 shows the interface of *Glyph* visualization with example data.

Glyph is composed of a dual-view interface that shows data from two related perspectives: A state graph and a sequence graph. A state graph (left in Figure 2a) is a node-link representation that shows a sequence of abstract behaviors and abstracted states. In *Glyph*, we used nodes on this graph to represent different game states, at which players make decisions and execute actions. The links between these nodes are actions that players took to get from one state to another. As shown in the Figure 2a (left), we assigned colors to the nodes, indicating the start state in blue and end states in red. In general, a game state captures all information associated with in-game entities that affect players' choice of actions.

To facilitate comparison of individual action sequences, the state graph is augmented with a sequence graph. A sequence graph (right in Figure 2a) showcases the popularity (number of players following a sequence) and similarity of different paths/patterns exhibited by users. Each node in the sequence graph represents a full play trajectory, with the size being an indication of popularity. The number on top of a node in the sequence graph is the identifier



(a) No highlight



(b) Player trace comparison

Figure 2: Figure 2a shows a *Glyph* screen where no pattern is highlighted. Figure 2b, in contrast, shows a *Glyph* screen highlighting four different sequences or patterns for solving a game level, which has three different solutions. These patterns are highlighted in red, blue, green, and orange, shown in the sequence graph (right) with the corresponding traces shown in the state graph (left)

of the sequence. Furthermore, the distance between each node provides a visual representation of similarity/dissimilarity.

By inspecting Figure 2b, one can see three different ways that players developed solution traces within a game level. To understand how such information can be gleaned from the visualization, inspect the sequence (right) and state (left) graphs shown in the figure. The sequence graph shows the clustering of the sequences displayed in distinct spaces. As discussed above, distance is used to show how different the sequences are. The fact that there are three distinct clusters tells us that there are three distinct solution paths.

To inspect this further, the visualization system allows users to interact with the graphs, highlighting different sequences to inspect the decision-making steps in detail. The state graph in figure 2b (left) shows a rendering of four different paths towards solutions for the level. This interactivity allows us to use *Glyph* to discover underlying patterns and irrelevant variances.

It is not a trivial task to examine the uniqueness and commonality of patterns in most of the visualization systems. It may be possible to compare two traces in some visualization systems, but comparing ten traces is not a trivial task as almost all of them focus either

on individual traces or overall aggregation. *Glyph* allows for easy understanding and investigation of player behavior by providing answers to questions such as "What are the top 10 most popular sequences?", or "How is the most frequent losing pattern compared to winning patterns?". *Glyph* is based on objective data, scalable, and provides a synchronized interactive visualization of both an aggregated overview and individual traces [33]. Furthermore, *Glyph* is unique in allowing users to navigate and interpret the sequence of states visited together with population clusters which is needed to allow for iteration on abstraction. These features together with its demonstrated application in analyzing player behavior [25, 34] led us to opt for Glyph in this study.

3.3 Dataset: 280 Players

We developed this abstraction method as part of making *Daedalus* and demonstrated how was it was useful to understand player patterns. We collected data for analysis by running a study in the Summer of 2018. This involved recruiting people from various venues including game clubs and classrooms at our institution and neighboring universities. The data collected comprised of their in-game actions and the messages they sent to certain channels of the *Slack* workspace. The in-game actions are comprised of the screens the players visited in the game and whether or not a submitted solution is correct. Individual players' actions are logged whilst playing *Daedalus*, and thus we are able to analyze each team member in both isolation and with their group. The data also contains chat from messages between the teams. This chat data is not included in the analysis except for validation of some of the assumptions we made (discussed further in the next section). We recruited 280 players in total, including people who completed the game (126 players) and those who did not (154 players). Data from all these participants is included in the analysis.

4 ABSTRACTION METHOD

Figure 3 shows the unabstracted data in *Glyph*, i.e. the information on all the screens visited by players in the game. The large state space makes it practically impossible for a human expert to see the decisions players are making.

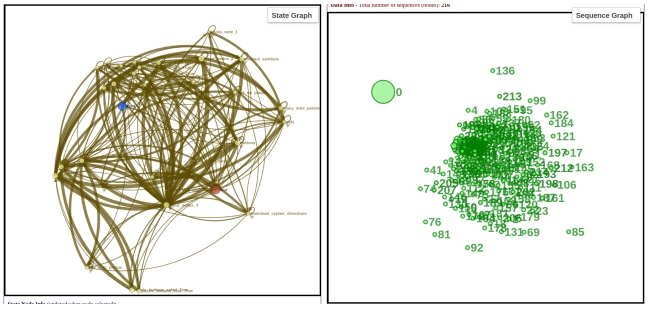


Figure 3: Visualization of the unabstracted play traces

To address this we developed our abstraction method. Figure 4 shows a schematic overview of our abstraction method. The visualization via *Glyph* is the backbone of the whole process. The abstraction process and distance measure development processes both rely on visualization of data to tune the state (node) representation or distance measurements, respectively. We will start by explaining the iterative process of abstraction followed by showing the final state representation. Then we discuss the development of the *ideal trace* to compare patterns against, in order to see how participants converge or diverge from this ideal behavior and where that happens over time. After that, we move on to describe the distance measurement process.

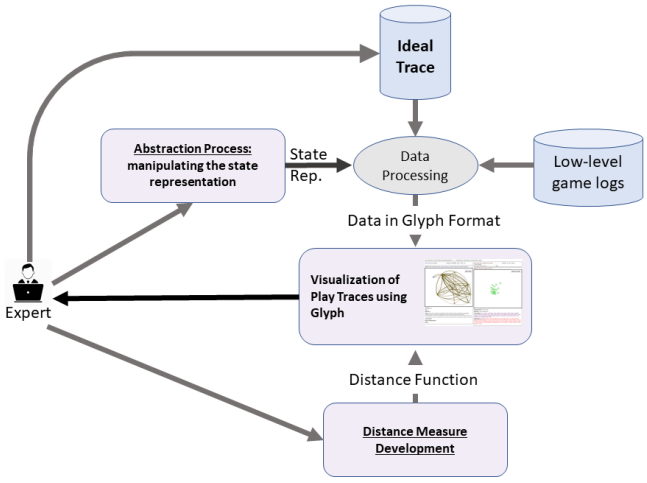


Figure 4: An overview of our abstraction method

4.1 Abstraction Process

With the goals of recognizing patterns of play and identifying puzzles that are engaging (in terms of multiple failed attempts leading to eventual success), frustrating (failed attempts leading to eventual withdrawal) and relatively easy to solve (none to few failed attempts leading to success), the following process is followed.

Initially, experts prepared a list of states (nodes), e.g. solved nodes, failure nodes, screens/cues visited etc. required to achieve the objectives of analysis. Here, by experts we mean a team of 5 members consisting of 2 game designers (who are part of a game company) and 3 game user researchers. This representation is then encoded with the low-level game logs into states, which in turn are fed to *Glyph* to visualize the nodes and links. The process of adjusting the state representation is an optimization process, whereby the expert is trying to minimize the noise and explosion of the state space while also using a state representation that allows them to explore problem-solving strategies. The abstraction rules at the end of each iteration are then put into a python script[1] to automatically convert raw data logs into abstracted sequences. The iterative order and logic used for conversion are described below. Experts are involved in this stage to ensure that the representation is one that is interpretable and can produce meaningful results.

We first looked at abstracting how players solved puzzles by the number of puzzles solved. This was achieved by removing all the

[1]The python script used for this abstraction process can be found at https://github.com/nitheshj/data-abstraction.

solved puzzle identifiers such as *puzzle_buttons_safari_true* etc., which are logged when particular puzzles are solved. Then a single node *solved_x*, with *x* being the number of solved puzzles, is inserted in place of the final solved puzzle identifier. Although this rule condenses the play sequence, it was not useful for analyzing solved puzzles as the objective is to look at the granular details of which puzzles are solved in what order along with the attempts it took players to eventually solve them. So we resorted to a separate node for each solved puzzle by replacing the identifiers with *solved_x*, where *x* is the name of the puzzle, for easy identification and clarity.

We used similar logic to look at every occurrence of a failed attempt to solve a puzzle, and represented this with the *failed_node* state. This was implemented by replacing all failed attempt identifiers with a single node. It should be noted that the failure node is not specific to any puzzle but just a generic one common to all puzzles in order to bring the number of nodes further down. This does not distinguish players attempting the same puzzle multiple times from those attempting once. To resolve this, we came up with *failed_1*, *failed_2*, etc. states, which show sequential failure attempts. However, this approach results in a large number of different nodes, making the state graph balloon into an unreadable size. To resolve this, we included *failed_once* and *failed_many_times* states. These first two iterations helped us identify the puzzles solved by each player and the number of times they attempted to solve each puzzle.

However, there were still a large number of nodes making it impossible to interpret the resulting visualization. We noticed that many nodes logged to identify players movement across puzzles, such as *main_dark_room*, *ladder_down* etc. We prepared a list of all such nodes and replaced them with a common *navigation* node using the python script discussed above. This abstracts the player movement and collapsed a 13 identified navigation nodes into single type of node, making the visualization more interpretable.

Another important category of abstraction are the cues to solve the puzzles. It is not possible for players to solve the puzzles without these cues so every instance of a player observing a cue is logged. In some cases more than one cue is needed. There are 10 such cues logged, such as *pick_up_plaque*, *download_cypher_wheel*, *video_safari* etc. To abstract these type of logs, we started with grouping them based on puzzles like *many_dots_cue*, *safari_cue* etc. but that did not significantly reduce the number of nodes as it would be proportional to the number of puzzles. Instead, we used an approach to abstract them into two nodes, *relevant_cue* and *irrelevant_cue*, using the knowledge of the game. This was possible because the puzzles can only be solved in a particular order. So by encoding this order information into a python script, all the nodes corresponding to cues are converted into either relevant or irrelevant cues based on the order in the trace.

When visualized with *relevant_cue* and *irrelevant_cue* nodes, we observed the instances of players solving puzzles without visiting relevant/irrelevant nodes. With the help from teammates players are able to solve the puzzles without visiting the relevant cues, e.g. if any of the teammates are stuck at a particular puzzle the ones who solved this puzzle shared their solution with the rest to fast track the team's progress. We verified some of these instances by looking at the time stamps of the play traces and conversations in respective slack channels of the teams. To consider this in the analysis we added *no_relevant* using the python script when a player goes through a solved puzzle node without viewing the corresponding cue required to solve the puzzle. The final state representation we obtained at the end of this iterative process and used for the analysis is discussed in detail in the following section.

4.2 State Representation

After several iterations as described above, using game knowledge to collapse nodes, and visualizing the results several times, we came up with the following abstracted state representation that encompassed the information we required:

- *Navigational Nodes*: All the screens that allowed players to navigate between different puzzles, or in same cases between stages, are clubbed together into the **navigation** node.
- *Solved Nodes*: To represent the number of puzzles solved by an individual player, we introduced states that indicate the name of the puzzle completed. An example is **solved_x**, where *x* is the name of the puzzle in the game.
- *Cue Nodes*: We used the order information in which the puzzles must be solved to figure out whether or not the player is visiting a relevant cue before they solved a puzzle, and use that to abstract the cue screens into two states: **irrelevant_cue** and **relevant_cue**.
- *Failure Nodes*: To analyze player failures, we looked at the number of actions between two successive failures and if they were below a certain threshold, we collapsed the series of failures into the state **failed_many_times**. We found that a threshold of three actions between successive failures works best, representing a good balance between capturing the repeated behavior and not overcompensating and collapsing failures that might be unrelated. A single failed attempt is represented as **failed_once**.
- *Help from Teammate Node*: To capture players that are receiving help from teammates, we added the state **no_relevant**. We accomplished this by looking at the last *x* actions taken before solving the puzzle, and seeing if there were any relevant cues. If there are not, we assumed that they received the solution from a teammate. We found that x=8 works best; this gives some leeway for players who looked at other screens before realizing which one was relevant, while still capturing players who "suddenly" solved the puzzle correctly.
- *Gave Up Node*: This is to show the number of puzzles a player solved before abandoning the game, so that the differences between the play traces can be meaningfully accentuated. Thus, we added a state **gave_up_x** at the end of the traces of those who did not solve all puzzles, where *x* is the number of puzzles they solved before reaching the end. We also added the state **gave_up_without_trying** to separate players who did not make any effort at all.

4.3 Ideal Trace

A necessary component of our method is developing the ideal trace to compare patterns against, in order to see how participants converge or diverge from this ideal behavior and where that happens over time. This can also be the behavior that the designers intend to provoke in players during gameplay. In our approach, the ideal trace is set by experts in the abstracted state representation. Generally,

there are two approaches in creating the ideal trace: The experts can either (1) handcraft the trace, or (2) select a found trace that satisfies some performance criteria and set it as the ideal trace. The first approach will be suitable for cases when an optimal behavior path is known, such as when solving a puzzle with specific known steps. The second approach works better for cases when there are more unique ways than just one to perfectly carry out the task, such as tasks where high performance is key. Instead of having to manually set individual components in the ideal trace, experts only need to consider the existing best-performing traces based on their performance measures and pick the one that looks "ideal" to them.

Daedalus's ideal trace falls into the first category as the optimal path is known, i.e. players pass only through the relevant cue(s) for each puzzle as that puzzle becomes available to solve, then solve that puzzle immediately after. This signifies the ideal player's ability to quickly identify the appropriate cues for each puzzle and solve them in the right order.

4.4 Distance Measure Development

Sequences obtained from *Daedalus* gameplay data vary highly in terms of length and order of states. To accommodate for these variances, we opted to use Time-Warp Edit Distance (TWED) [28] as our distance measure. TWED has many desirable properties: It is a metric and so follows the triangle inequality, which makes comparing distances easier, and most importantly it has been shown to outperform many popular distance measures for time-series classification [40]. We tried using various values for α, the elasticity parameter, and deletion cost λ, and settled on $\alpha = 0.001$ and $\lambda = 1$.

Initially, we explored what would happen if our state distance function weighed everything equally, i.e. if the state distance function returns 1 whenever there is a mismatch, and returns 0 at any other time. This proved to be off the mark by a large margin: Traces were mostly assessed by their sequence length because there was more room for discrepancies within longer sequences than shorter ones. Thus, we needed to penalize certain states over others, allowing those differences to overpower differences in length. After some iterations, we converged onto the following scheme:

(1) If one of the sequences has state **gave_up_x** and the other does not, then the distance is proportional to **x** or the number of puzzles solved before abandonment.
(2) If one of the sequences has state **gave_up_without_trying** and the other does not, then the distance between them is highest 600.
(3) If one of the states is **no_relevant_cue** and the other is not, then the distance is 20.
(4) If one of the states is **failed_many** and the other is not then the distance is 40.
(5) If one of the states is **failed_once** and the other is not, then the distance is 2.
(6) Otherwise, if there is a mismatch the distance is 1.

The idea behind (1) and (2) is to separate players who left the game from those who completed the first stage. Because we needed to differentiate players in terms of their problem-solving abilities, it made sense to penalize those who gave up. The differences in (1), and between (1) and (2), are motivated by the fact that the earlier the game is left, the lower the individual in question's problem-solving capacity ought to be scored.

(3), (4), and (5) exist to accentuate differences in traces of players who completed the stage, or were close to completion. The largest penalty of the three is (4), which is to penalize those who attempted to brute force a solution, while (5) should ensure that players who did not fail a puzzle land closer to the ideal trace than players who did fail. (3) exists to further discriminate between those players who used a solution given to them by teammates and those players who did not. Because players are encouraged to share answers, the penalty for (3) is not that high, but it is there to punish reliance on teammates without looking at how the puzzle was solved.

We envision the iterative process as optimizing a function $w_1s_1 + w_2s_2 + ... + w_is_i$, where w_i are weights for the sequences s_i in the TWED distance function. The abstraction process is defining which nodes are required in the sequences and the distance metric function is optimizing the weights. All the sequences and distances are then fed into the visualization to evaluate them. Here the evaluation measure is twofold: (1) if the qualitative difference between player patterns (in the state graph) is exhibited and (2) if the number of patterns shown (in the sequence graph) is distinct enough and not cluttered. If the nodes in the sequences and weights are too small, we will see no clutter but no distinct patterns either; if the nodes and weights are too large, then we will see a cluttered a graph and thus no differences. Both the abstraction process and distance measure development are essentially part of the optimization of the same objective function. So it is required to iterate between these abstraction and distance measure processes to achieve both qualitatively different patterns and non-cluttered distinct sequences.

Figure 5 shows the abstracted data visualized in *Glyph* highlighting example sequences following different paths. We observed 6 different types of player patterns based on the paths followed.

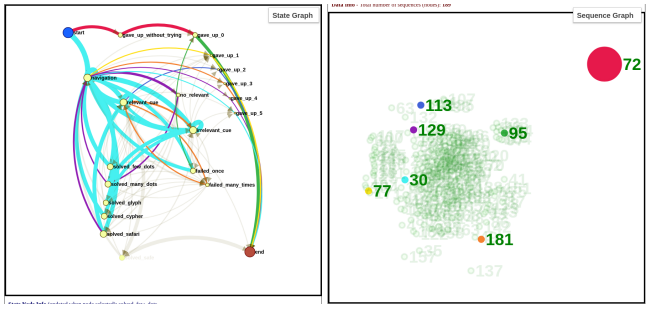


Figure 5: The abstracted representation visualized in *Glyph*

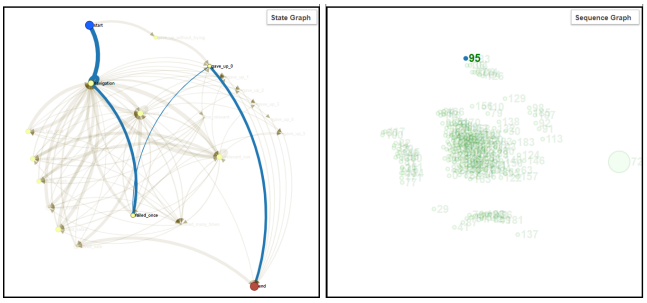
5 ANALYSIS RESULTS

Here, we exemplify the analysis results from applying the abstraction method to *Daedalus* to observe patterns in player progressions. This allows to better understand how individuals navigate through the space of possibilities in the game and exhibit different behaviors during the course of their play, which can be helpful in the (re-)design of the game. Specifically, we demonstrate how our method helped elucidate specific patterns that can differentiate how players solve problems. We identified six patterns of

play. Each pattern is discussed with an example figure and final abstracted sequence of that pattern. The number in parentheses to the side of each state in the final abstracted sequence (see Figure 6 for example) indicates the number of times the state is visited consecutively. In our supplemental material, we included the complete and large-scale screenshots of the figures presented here. Further, the interactive visualization of the *Daedalus* game data can be found at: https://nitheshj.github.io/GlyphARG/index.html.

5.1 Early Dropouts

This type of players left the game very early in the game. Trace 95, shown in green in Figure 6, is an example that shows a player who failed and gave up early, and thus is incredibly short. Looking at the states that the players went through, we observed that the player started the game, did not visit any cue nodes, made one failed attempt at solving a puzzle, and left the game. As players that followed this trajectory did not visit any cues, we can assume that these players did not display any discontinuity in thinking, i.e. they attempted to solve the puzzle in one particular way, and when failed they quit playing instead of trying other ways .



Abstracted Sequence: navigation(4), failed_once(1), gave_up_0(1)

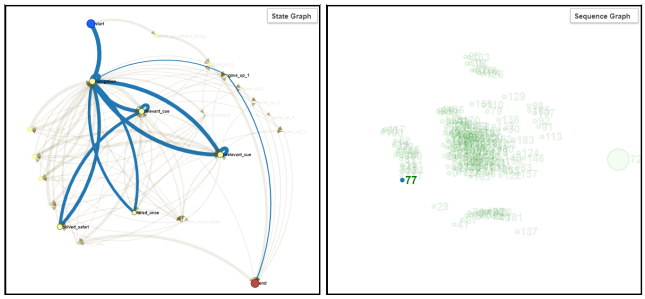
Figure 6: An example of early dropouts-Trace 95

5.2 Arduous Dropouts

In contrast to Early Dropouts, the "Arduous Dropouts" category shows players who were trying to explore and get as much information as possible before attempting any solution, but who still ended up dropping out. Trace 77 is an example (see Figure 7). This player's sequence mostly consist of navigation between (exclusively irrelevant) cues, and then entering an answer. Their answer was incorrect and thus the player's mental model of the problem was likely changed. This is further demonstrated by the fact that they received help from a friend before solving the puzzle. This suggests that the player changed their strategy after their failed attempt at solving the puzzle, and reached out for assistance from teammates. After receiving and entering the correct answer for the "safari" puzzle, they gave up playing. It is not clear why they did so, but perhaps the failure of their first strategy and the fact that they had to think of the problem in a different way frustrated them.

5.3 Dodgers

Many players in the "Dodgers" pattern of play used the strategy of getting help from teammates. Trace 129 is an example of this



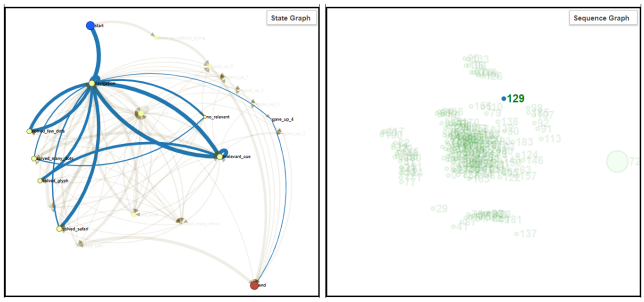
Abstracted Sequence: navigation(2), irrelevant_cue(3), navigation(3), irrelevant_cue(3), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(3), navigation(3), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(1), navigation(3), irrelevant_cue(1), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(3), failed_once(1), navigation(2), solved_safari(1), relevant_cue(3), navigation(1), gave_up_1(1)

Figure 7: An example of arduous dropouts-Trace 77

type (see Figure 8). They appear to begin with a similar tactic of exploration to that of Trace 77 in Arduous Dropouts, but they did not solve without consulting a teammate. This player solved four puzzles before leaving the game. It is very likely that despite getting the answers to four of the puzzles, they did not display any discontinuity in thinking. They did not solve any of the puzzles themselves, and thus it is likely that their representation of the puzzle did not change; since they never "failed", the way they thought about the game did not alter. This strategy could be prompted by their uncertainty, and feeling that they did not have enough insight to make a solid attempt. The repeated help from teammates also indicates that players in this trace were likely falling behind. At some point they decided to try exploring again, to see if they could gather enough information to feel confident in solving the puzzle. But all the cues they visited were irrelevant to the puzzle they needed to solve. The fact that they quit after a couple of trials indicates their frustration at being unable to play on their own.

5.4 Resolute Finishers

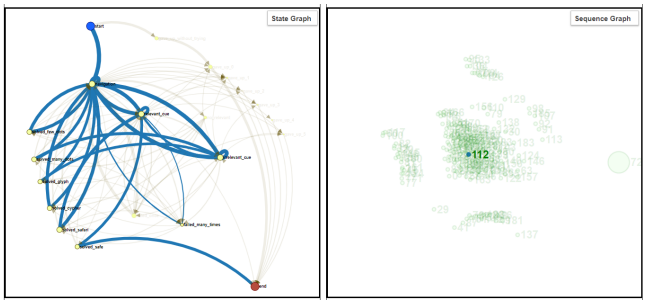
Trace 112, shown in Figure 9, is an example of the "Resolute Finishers" pattern of players, who solved puzzles both on their own, and with the assistance of others. They started exploring in a way similar to Traces 129 (Dodgers) and 113 (Arduous Dropouts), before getting help with the safari puzzle. Then they solved the "few dots" puzzle after observing a relevant and an irrelevant cue. This can likely be attributed to their teammates' help. However, the player attempted several incorrect solutions for the "many dots" puzzle before finally getting the right answer. This indicates that there was a discontinuity in their thinking. Furthermore, it indicates that there has been some restructuring of the representation of the problem, as they had to think of the cues differently to get the right answer. The mix of getting help and self-reliance indicates that players like this were likely heavily involved with the game, and that they communicated with their teammates often. They continue



Abstracted Sequence: navigation(2), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(4), navigation(3), irrelevant_cue(2), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(3), irrelevant_cue(1), navigation(3), solved_few_dots(1), navigation(2), solved_safari(1), navigation(2), no_relevant(1), solved_many_dots(1), navigation(1), irrelevant_cue(3), navigation(2), solved_glyph(1), irrelevant_cue(1), navigation(1), gave_up_4(1)

Figure 8: An example of dodgers-Trace 129

to receive help up until the end, which shows that the teamwork in this group did not falter. This trace shows a different approach to getting help than Traces 129 and 113: Instead of complete reliance on their teammates to do the heavy lifting, we see signs of collaborative problem-solving.

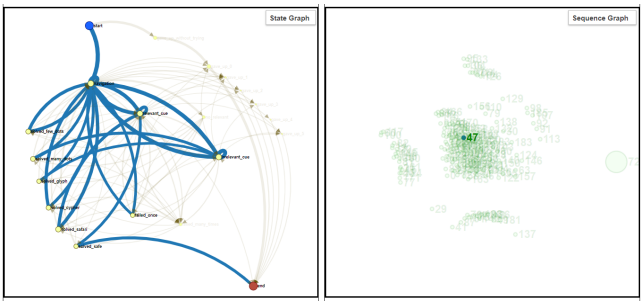


Abstracted Sequence: navigation(2), irrelevant_cue(5), navigation(8), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(1), navigation(3), irrelevant_cue(1), navigation(3), solved_safari(1), relevant_cue(2), navigation(1), irrelevant_cue(1), navigation(3), solved_few_dots(1), navigation(1), irrelevant_cue(1), relevant_cue(1), navigation(1), relevant_cue(2), navigation(5), relevant_cue(1), failed_many_times(1), navigation(3), solved_many_dots(1), irrelevant_cue(3), navigation(6), irrelevant_cue(1), navigation(2), solved_glyph(1), irrelevant_cue(1), navigation(2), solved_cypher(1), navigation(3), relevant_cue(1), solved_safe(1)

Figure 9: An example of resolute finishers-Trace 112

5.5 Elite Players

"Elite players" are the type of players that completed the game with minimal effort and help from teammates. The ideal trace falls under this category in the current study – but note that this may not be the case in general, if the ideal trace is designed with an intention to objectively measure other patterns such as discussed above. Trace 47, shown in Figure 10 is an example of an Elite Players trace. Typically, this set of players explore for a bit and then attempt to link cues with puzzles. After a failed attempt, they try a different approach. This indicates their discontinuity in thinking and continuously adapting to different approaches in problem-solving.



Abstracted Sequence: navigation(3), irrelevant_cue(1), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(3), navigation(8), failed_once(1), navigation(2), irrelevant_cue(3), navigation(3), solved_few_dots(1), navigation(2), solved_safari(1), relevant_cue(1), navigation(1), relevant_cue(2), navigation(3), solved_many_dots(1), irrelevant_cue(3), navigation(2), solved_glyph(1), irrelevant_cue(1), navigation(2), solved_cypher(1), navigation(3), relevant_cue(1), navigation(1), relevant_cue(1), navigation(3), relevant_cue(2), navigation(3), relevant_cue(1), navigation(1), relevant_cue(1), navigation(6), relevant_cue(1), solved_safe(1)

Figure 10: An example of elite players-Trace 47

5.6 Mixed Patterns

Finally, certain players exhibited different patterns over the course of play and hence are located in the grey areas between different patterns. One such example is Trace 88 (see Figure 11). This player got help from teammates and then proceeded to enter the correct solution to the same puzzle more than once indicating characteristics of both dodgers and resolute finishers. Similarly, there are players who displayed attributes of different groups over time.

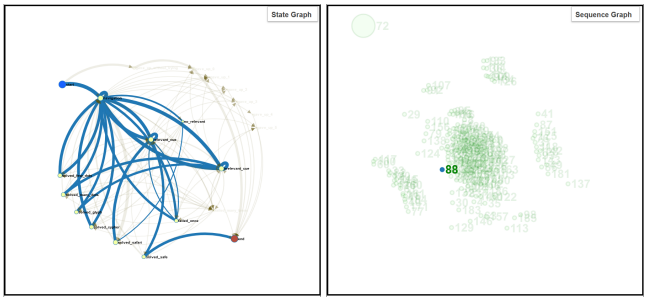
6 DISCUSSION

In this section, we first discuss the insights we gained about the player patterns with respect to the design of *Daedalus*. We then present a general set of guidelines for expert knowledge-based abstraction of game log data based on the lessons learned from this work; in other words, we attempt to abstract the abstractions.

6.1 Design Evaluation

By applying our method on data generated by the interactive fiction puzzle game *Daedalus*, we have been able to capture interactions and behavior of players by abstracting behavioral telemetry data. We were also able to meaningfully interpret the patterns of play exhibited. The question is now what this means, specifically for the design of *Daedalus*. In this section, we describe the design lessons we learned starting with analysis of expected player patterns, then move to unexpected players patterns. We end with discussing the balancing issues of *Daedalus*, and the applicability of our method.

6.1.1 Expected Player Patterns. First, the results highlight some expected play patterns, which are predicted by the designers of



Abstracted Sequence: navigation(2), irrelevant_cue(2), navigation(1), irrelevant_cue(3), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(1), irrelevant_cue(1), navigation(3), failed_once(1), navigation(4), failed_once(1), navigation(4), no_relevant(1), solved_safari(1), relevant_cue(3), navigation(2), irrelevant_cue(3), navigation(3), solved_few_dots(1), navigation(2), solved_many_dots(1), irrelevant_cue(3), navigation(2), solved_glyph(1), irrelevant_cue(1), navigation(2), solved_cypher(1), navigation(3), relevant_cue(1), navigation(1), relevant_cue(1), navigation(1), relevant_cue(1), failed_once(1), navigation(2), irrelevant_cue(4), navigation(3), relevant_cue(1), failed_once(1), navigation(2), irrelevant_cue(1), navigation(5), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), failed_once(1), relevant_cue(1), solved_safe(1)

Figure 11: An example of mixed pattern-Trace 88

Daedalus. The Early Dropout profile players are one example of this; our method also distinguished players who quit playing *Daedalus* not immediately, but after some significant puzzle solving efforts. Another revealed behavior revealed is the "brute forcing" behavior, in which players attempted to solve puzzles simply by trying each possible answer combination. Although this behavior is expected, it occurred more than expected. Around 17% of the players who completed the first stage (i.e., 22 out of 126 players) tried to brute force more than once where possible. This behavior occurred even more in players who made some effort but did not complete the first stage at 23% (14 out of 60 players). We obtained these percentages by looking at the consecutive occurrences of *failed_many_times* node in the sequences. Finally, a particularly meaningful pattern that the analysis illuminated was the "help from friends" behavior, in which players managed to solve puzzles without actually visiting the relevant information cue. While the play trace only shows that players reach a *solved* node without first reaching the *relevant_cue* node, based on our understanding of *Daedalus*' design and players, we inferred that these players received the puzzle answers from their teammates. This behavior is something we expected as a possibility while designing *Daedalus*, but our method now incontrovertibly shows that it actually occurs.

6.1.2 Unexpected Player Patterns. An unexpected observation is the difference in difficulty levels between the two sets of puzzles (see Section 3.1 for details on sets). The number of players that completed the few dots, many dots, and cypher wheel puzzles are 146, 145 and 140, respectively. For the other set, the numbers that completed the safari slideshow and glyph painting puzzles are 166 and 130, respectively. This big difference between puzzle completion numbers from the latter set indicates the difficulty of the glyph painting puzzle. This is exacerbated when we consider the difficulty players faced when they begin with the safari slideshow puzzle as starting point as discussed above. Furthermore, players have to recognize the order in which puzzles have to be solved to reach the end, making the game harder. This explains why 51% of the total 280 players left the game by the time of solving three puzzles. Our method facilitates identification of game elements and balancing issues that cause low retention which is not possible with the existing literature on churn and retention models [13, 57].

6.1.3 Balancing Daedalus. We identified several design and balancing issues as well. For instance, the fact that players drop out of playing *Daedalus* early even when successfully completing the game comes with a monetary reward indicates a problem: Apparently, these players enjoy playing *Daedalus* so little that not even the promise of money can keep them engaged. In relation to that, the fact that some players drop out of *Daedalus* after trying and failing a given puzzle several times suggests that some (early) puzzles might be more difficult to solve than the designers anticipated. For example, players have two starting points in the few dots and safari slideshow puzzles, as they both can be solved individually. Only 146 players were able to solve the few dots puzzle while 166 players solved the safari slideshow puzzle. Interestingly, 56% players failed one or more times in solving the safari slideshow puzzle when only 10% players failed in solving the few dots puzzle. This means the safari slideshow puzzle is both challenging and engaging at the same time but not an ideal starting point.

Although our method has been applied to understand the difficulty and engagement of different puzzles, it can be easily applied to different levels of a game to adjust the level progression and thereby improve player retention. This analysis has been successful in identifying both barriers hindering player progress and puzzles causing players to drop out. In both cases, the analysis reveals points of attention for *Daedalus*: A general notion of poor onboarding for the former one, and specific indications of poorly tuned puzzles for the latter. The problem pertaining to onboarding can be resolved by providing players rewards or incentives with accomplishments [19].

Our abstraction method proved to be useful in identifying both expected and unexpected player patterns by visually analyzing gameplay data. Previous approaches, such as CMDS applied in [2, 26, 52], are particularly advantageous in spatio-temporal data from RTS or MOBA games. Although Wallner et al. [51] successfully applied CMDS for a puzzle game, it requires a dissimilarity matrix which may not be available in all cases. Our method has great potential for application of rapid user experience testing of games during design phases because of its generalizability.

Our method demonstrates how to systematically evaluate "why" (not) certain player behaviors emerge by making visualization of player traces more manageable (interpretability & transparency). Obtaining such manageable play traces sets our method apart from existing literature in deriving meaningful insights about game design such as flow, balancing, mechanics, etc. The method is particularly suitable for games with problem-solving, decision-making, and reasoning (inductive/deductive) as their core gameplay and that do not have a highly variable state space.

6.2 Abstraction Guidelines

We came up with a set of guidelines for abstracting raw data to analyze player behavior. We attempted to generalize these guidelines to fit other game genres and visualization systems, using the

lessons we learned from our work. The seven resulting guidelines – discussed below – can be useful when analyzing both individual and team gameplay behaviors.

6.2.1 Use the objectives of the analysis to identify the critical nodes on which the analysis hinges. One of the objectives of the presented study was to look at the patterns and progression of players. We are interested in looking at players who leave the game, and the last puzzle they attempted before leaving. This way, we will be able to identify if any particular puzzle is causing excessive frustration in players, causing them to quit. Hence, we came up with the nodes *gave_up_x*, where *x* is the name of the puzzle. In general, it is important to look for mechanics that should be tested for, such as arriving at a location or communicating with a PC/NPC, and to have the related nodes included in the abstraction. These can be specific to e.g. frequency, time, or space. Having an underlying theme of investigation (such as identifying patterns, or finding balancing problems) would help in having a smaller number of nodes, which helps in making the sequences even more interpretable.

6.2.2 Define the set of goals that players have to accomplish during play, and merge any sub-goals into goals based on the granularity of the analysis. In the current study, players have to complete all the puzzles to complete the game. Consequently, nodes corresponding to all the puzzles have been inserted to be able to see players' rate of progression and order of completion: The *solved_x* nodes, where *x* is the name of the puzzle, serve this role. Developing a hierarchy of goal structure and prioritizing the goals that are related to the objectives (in guideline #1) based on the expert knowledge will allow identification of possible abstractions. It should be noted that it is occasionally important to have nodes that indicate players' perception of the goals, e.g. when the goals or objectives are intentionally unclear. In these cases, the loop of hypothesize-experiment-evaluate-understand can be collapsed into a single node indicating a type of attempt to achieve a particular goal.

6.2.3 Create a set of actions, rules, objects, and attributes required to accomplish the goals in guideline #2, and check if there is a class structure for condensing them. To be able to complete the game, players in *Daedalus* have to navigate and look at different cues available for solving the puzzles. Every puzzle has a corresponding cue available for solving them. Instead of having a node for each cue, we only had two classes, *relevant_cue* and *irrelevant_cue*, which are identified by looking at the puzzle players who looked at these cues subsequently attempted to solve. Players often have to follow certain rules, perform certain actions, or collect certain objects to attain particular attributes, which gives an opportunity to collapse all of them into a single node.

6.2.4 Insert nodes that indicate the consequence of an attempt, i.e., whether they fail or succeed. In our work, we created *solved* and *failure* as nodes indicating action consequences. This enabled us to inspect the player behaviors involved in completing the decision-making loop of action-consequence-feedback. Based on expert knowledge, it is often possible to identify and abstract the misleading, blind, or meaningless decision-making processes. This identification can be achieved in an iterative process. For example, our study observed that players are visiting *failed_many_times* without entering *relevant_cue* or even *irrelevant_cue*, indicating the brute-forcing approach.

6.2.5 Abstract player movement in space by dividing in zones and movement over time by dividing into time windows wherever possible. Movement across space and time can generally be easily abstracted, unless there is a significant event that hinders such abstraction. Again, this can be identified iteratively by carefully inspecting all the traces moving towards a particular location. The node *navigation* presented in the study represents the movement of players from one puzzle to another. It should be noted that as location and time (not part of intended analysis objectives) are irrelevant here, these elements are ignored in the abstraction.

6.2.6 Develop metrics based on the game for measurement of skills that are of interest. This common approach is followed by game user researchers and data scientists in industry. Based on the objectives of analysis, several variables of interest are developed to aggregate the data over time. For example, time spent per location could be a useful metric when looking for player types. Performance metrics such as the quest-completion rate or deaths/kill ratio could be of interest in battle games. Although no such nodes are used in our case, we included this guideline as it is a widely practiced method, particularly in industry [16].

6.2.7 Inspect for condensing nodes both horizontally and vertically in the set of nodes developed from the above guidelines. An example of vertical collation is the exploration loop described in guideline #2, which contains elements from guidelines 3,4,5 such as action, consequence, location details which can be merged into a single node, for say, *Failed_kill_attempt _at_Y*. Such abstractions are particularly useful if the behavior is expected in the majority of participants. Examples of horizontal collations can be abstractions such as merging series of sub-tasks to perform an action into a single node (as in guideline #4). Using expert knowledge and careful examination, such mergers can be identified and applied.

6.3 Limitations

Several limitations impact the generalizability of our work. First, our study was only conducted on the data of one particular game. It is thus unclear if the method could be used with other (commercial or serious) games quite as easily. Although the node definitions and distance measures are game and goal specific, we believe the method/results from *Daedalus* can be generalized to other puzzle games, which share the benefits of a limited state space which generally progress in one direction. However, data from other game types may be more difficult to use. For instance, RTS games involve a highly variable state space, dependent on the units produced and actions taken by players. This calls the broad generalizability of our method into question. Nevertheless, for understanding game data in such games knowledge-based abstraction would likely still be useful; it may therefore be more the case that our guidelines may be less applicable and new guidelines have to be developed.

Also, our method may fall short when analyzing other game types with richer feature sets and/or more dynamic and emergent gameplay, such as Massively Multiplayer Online Role-Playing Games (MMORPG) or MOBA games, in which it would be difficult to abstract from the visualization of play traces alone. Developing

abstraction rules for these complex domains is a difficult problem and may require special tools as discussed by Ahmad et al [1]. Additionally, it may not always be feasible or tractable to develop an ideal trace by hand. Developing a way to automatically create these ideal traces, given the game environment, would alleviate this issue considerably. However, it is unclear if this is achievable without a similar amount of expert intervention.

Finally, our analyses have not included every variable of interest for reasons of scope. We also reduced the role of teams to whether individuals received help from teammates and thus did not look at play patterns on the level of teams. Moreover, we limited our analysis to the first out of five stages. Finally, while we asked all participants to *only* communicate with each other over *Slack*, we may not have captured all forms of player communication. As such, our presented analysis is limited in understanding the full range of player behavior in *Daedalus*; however, the purpose of this paper was to demonstrate the method, not to analyze *Daedalus*.

7 CONCLUSION

In this paper, we demonstrated the utility of our knowledge-based iterative abstraction method for analysis of game data. We have shown how this analysis can condense a large data space into a manageable size, and how context-driven expert analysis of traces and patterns of play can provide design insights into the chosen game (both expected and unexpected). We believe this approach can be used to model other constructs (e.g., strategies instead of problem-solving) and game contexts – thus adding a new approach to the field of game analytics. More importantly, it is an approach that game user researchers and designers can use in their practice as it addresses limitations of previous work on data-driven visualization approaches in terms of their interpretability and transparency. To facilitate the adoption of our work, we provided a general set of guidelines for knowledge-based abstraction. Future work should consider how others make use of and expand on these guidelines.

ACKNOWLEDGMENTS

We want to thank our participants and the team who built Daedalus. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense, DARPA, or the U.S. Government.

REFERENCES

[1] Sabbir Ahmad, Andy Bryant, Erica Kleinman, Zhaoqing Teng, Truong-Huy D. Nguyen, and Magy Seif El-Nasr. 2019. Modeling Individual and Team Behavior through Spatio-Temporal Analysis. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* (Barcelona, Spain) *(CHI PLAY '19)*. Association for Computing Machinery, New York, NY, USA, 601–612. https://doi.org/10.1145/3311350.3347188

[2] Erik Andersen, Yun-En Liu, Ethan Apter, François Boucher-Genesse, and Zoran Popović. 2010. Gameplay analysis through state projection. In *Proceedings of the fifth international conference on the foundations of digital games*. ACM, 1–8.

[3] Natalia Andrienko and Gennady Andrienko. 2013. Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization* 12, 1 (2013), 3–24.

[4] Martin Ashton and Clark Verbrugge. 2011. Measuring cooperative gameplay pacing in World of Warcraft. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. ACM, 77–83.

[5] Myat Aung, Simon Demediuk, Yuan Sun, Ye Tu, Yu Ang, Siva Nekkanti, Shantanu Raghav, Diego Klabjan, Rafet Sifa, and Anders Drachen. 2019. The trails of Just Cause 2: spatio-temporal player profiling in open-world games. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–11.

[6] Alessandro Canossa, Ahmad Azadvar, Casper Harteveld, Anders Drachen, and Sebastian Deterding. 2019. Influencers in Multiplayer Online Shooters: Evidence of Social Contagion in Playtime and Social Play. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 259, 12 pages. https://doi.org/10.1145/3290605.3300489

[7] Alessandro Canossa, Truong-Huy D Nguyen, and Magy Seif El-Nasr. 2016. G-Player: Exploratory Visual Analytics for Accessible Knowledge Discovery.. In *DiGRA/FDG*.

[8] Olivier Cavadenti, Victor Codocedo, Jean-François Boulicaut, and Mehdi Kaytoue. 2016. What did i do wrong in my MOBA game? Mining patterns discriminating deviant behaviours. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 662–671.

[9] Luca Chittaro, Roberto Ranon, and Lucio Ieronutti. 2006. Vu-flow: A visualization tool for analyzing navigation in virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 12, 6 (2006), 1475–1485.

[10] David D Curtis and Mike Joseph Lawson. 2002. Computer adventure games as problem-solving environments. (2002).

[11] Anders Drachen and Alessandro Canossa. 2009. Analyzing spatial user behavior in computer games using geographic information systems. In *Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era*. ACM, 182–189.

[12] Anders Drachen, Alessandro Canossa, and Georgios N. Yannakakis. 2009. Player modeling using self-organization in Tomb Raider: Underworld. In *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, Milano, Italy, 1–8. https://doi.org/10.1109/CIG.2009.5286500

[13] Anders Drachen, Eric Thurston Lundquist, Yungjen Kung, Pranav Rao, Rafet Sifa, Julian Runge, and Diego Klabjan. 2016. Rapid prediction of player retention in free-to-play mobile games. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[14] Anders Drachen, Rafet Sifa, Christian Bauckhage, and Christian Thurau. 2012. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE conference on Computational Intelligence and Games (CIG)*. IEEE, 163–170.

[15] Christoph Eggert, Marc Herrlich, Jan Smeddinck, and Rainer Malaka. 2015. Classification of player roles in the team-based multi-player game dota 2. In *International Conference on Entertainment Computing*. Springer, 112–125.

[16] Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa. 2013. *Game Analytics: Maximizing the Value of Player Data*. Springer Publishing Company, Incorporated.

[17] Wu-chang Feng, David Brandt, and Debanjan Saha. 2007. A long-term study of a popular MMORPG. In *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*. ACM, 19–24.

[18] André R Gagné, Magy Seif El-Nasr, and Chris D Shaw. 2011. A Deeper Look at the use of Telemetry for Analysis of Player Behavior in RTS Games. In *International Conference on Entertainment Computing*. Springer, 247–257.

[19] Jacqueline Gaston and Seth Cooper. 2017. To Three or Not to Three: Improving Human Computation Game Onboarding with a Three-Star System. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5034–5039. https://doi.org/10.1145/3025453.3025997

[20] Fabian Hadiji, Rafet Sifa, Anders Drachen, Christian Thurau, Kristian Kersting, and Christian Bauckhage. 2014. Predicting player churn in the wild. In *2014 IEEE Conference on Computational Intelligence and Games*. Ieee, 1–8.

[21] Casper Harteveld, Erica Kleinman, Paola Rizzo, Dylan Schouten, Truong Huy Nguyen, Samuel Liberty, Wade Kimbrough, Paul Fombelle, and Magy Seif El-Nasr. 2019. Teamwork and adaptation in games (TAG) a survey to gauge teamwork. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–12.

[22] Nate Hoobler, Greg Humphreys, and Maneesh Agrawala. 2004. Visualizing competitive behaviors in multi-user virtual environments. In *Proceedings of the conference on Visualization'04*. IEEE Computer Society, 163–170.

[23] Britton Horn, Amy K. Hoover, Yetunde Folajimi, Jackie Barnes, Casper Harteveld, and Gillian Smith. 2017. AI-Assisted Analysis of Player Strategy across Level Progressions in a Puzzle Game. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (Hyannis, Massachusetts) *(FDG '17)*. Association for Computing Machinery, New York, NY, USA, Article 9, 10 pages. https://doi.org/10.1145/3102071.3102083

[24] Robin Hunicke, Marc LeBlanc, and Robert Zubek. 2004. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, Vol. 4. 1722.

[25] Chaima Jemmali, Erica Kleinman, Sara Bunian, Mia Victoria Almeda, Elizabeth Rowe, and Magy Seif El-Nasr. 2020. MAADS: Mixed-Methods Approach for the Analysis of Debugging Sequences of Beginner Programmers. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 86–92.

[26] Yun-En Liu, Erik Andersen, Richard Snider, Seth Cooper, and Zoran Popović. 2011. Feature-based projections for effective playtrace analysis. In *Proceedings of the 6th international conference on foundations of digital games*. ACM, 69–76.

[27] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N Yannakakis. 2010. Predicting player behavior in tomb raider: Underworld. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 178–185.

[28] P.-F. Marteau. 2009. Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 2 (Feb. 2009), 306–318. https://doi.org/10.1109/TPAMI.2008.76

[29] Ben Medler, Michael John, and Jeff Lane. 2011. Data cracker: developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2365–2374.

[30] Larry Mellon. 2009. Applying metrics driven development to MMO costs and risks. *Versant Corporation* (2009).

[31] Robert Moskovitch and Yuval Shahar. 2005. Temporal data mining based on temporal abstractions. In *ICDM-05 workshop on temporal data mining*.

[32] Dinara Moura, Magy Seif el Nasr, and Christopher D Shaw. 2011. Visualizing and understanding players' behavior in video games: discovering patterns and supporting aggregation and comparison. In *Proceedings of the 2011 ACM SIGGRAPH symposium on video games*. ACM, 11–15.

[33] Truong-Huy D Nguyen, Magy Seif El-Nasr, and Alessandro Canossa. 2015. Glyph: Visualization Tool for Understanding Problem Solving Strategies in Puzzle Games.. In *FDG*.

[34] Truong-Huy D Nguyen, Magy Seif El-Nasr, and Derek M Isaacowitz. 2015. Interactive visualization for understanding of attention patterns. In *Workshop on Eye Tracking and Visualization*. Springer, 23–39.

[35] Dmitry Nozhnin. 2012. Predicting churn: When do veterans quit. *Gamasutra, August 30th* (2012).

[36] Alain Saas, Anna Guitart, and Africa Periánez. 2016. Discovering playing patterns: Time series clustering of free-to-play game data. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.

[37] Anna Sapienza, Alessandro Bessi, and Emilio Ferrara. 2018. Non-negative tensor factorization for human behavioral pattern mining in online games. *Information* 9, 3 (2018), 66.

[38] Anna Sapienza, Yilei Zeng, Alessandro Bessi, Kristina Lerman, and Emilio Ferrara. 2018. Individual performance in team-based online games. *Royal Society open science* 5, 6 (2018), 180329.

[39] Matthias Schubert, Anders Drachen, and Tobias Mahlmann. 2016. Esports analytics through encounter detection. In *Proceedings of the MIT Sloan Sports Analytics Conference*, Vol. 1. 2016.

[40] Joan Serra and Josep Lluis Arcos. 2014. An Empirical Evaluation of Similarity Measures for Time Series Classification. *Knowledge-Based Systems* 67 (Sept. 2014), 305–314. https://doi.org/10.1016/j.knosys.2014.04.035 arXiv: 1401.3973.

[41] Yuval Shahar. 1997. A Framework for Knowledge-based Temporal Abstraction. *Artif. Intell.* 90, 1-2 (Feb. 1997), 79–133. https://doi.org/10.1016/S0004-3702(96)00025-2

[42] Rafet Sifa, Anders Drachen, and Christian Bauckhage. 2018. Profiling in Games: Understanding Behavior from Telemetry. *Social Interactions in Virtual Worlds: An Interdisciplinary Perspective* (2018).

[43] Sam Snodgrass, Omid Mohaddesi, Jack Hart, Guillermo Romera Rodriguez, Christoffer Holmgård, and Casper Harteveld. 2019. Like PEAS in PoDS: the player, environment, agents, system framework for the personalization of digital systems. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–15.

[44] Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M Rush. 2018. S eq 2s eq-v is: A visual debugging tool for sequence-to-sequence models. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 353–363.

[45] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2017. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 667–676.

[46] Ruck Thawonmas and Keita Iizuka. 2008. Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology* 2008 (2008).

[47] Ruck Thawonmas, Masayoshi Kurashige, Kuan-Ta Chen, et al. 2007. Detection of Landmarks for Clustering of Online-Game Players. *IJVR* 6, 3 (2007), 11–16.

[48] Clive Thompson. 2007. Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine* 15, 9 (2007), 15–09.

[49] Christian Thurau, Kristian Kersting, and Christian Bauckhage. 2009. Convex non-negative matrix factorization in the wild. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 523–532.

[50] Günter Wallner. 2013. Play-Graph: A methodology and visualization approach for the analysis of gameplay data.. In *FDG*. 253–260.

[51] Günter Wallner and Simone Kriglstein. 2011. Design and evaluation of the educational game DOG eometry: a case study. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*. 1–8.

[52] Günter Wallner and Simone Kriglstein. 2012. A spatiotemporal visualization approach for the analysis of gameplay data. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 1115–1124.

[53] Günter Wallner and Simone Kriglstein. 2013. Visualization-based analysis of gameplay data–a review of literature. *Entertainment Computing* 4, 3 (2013), 143–155.

[54] G. Wallner and S. Kriglstein. 2014. PLATO: A visual analytics system for gameplay data. *Computers & Graphics* 38 (2014), 341 – 356. https://doi.org/10.1016/j.cag.2013.11.010

[55] Yiting Wang, Walker M White, and Erik Andersen. 2017. PathViewer: Visualizing Pathways through Student Data. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 960–964.

[56] Ben George Weber, Michael John, Michael Mateas, and Arnav Jhala. 2011. Modeling player retention in madden nfl 11. In *Twenty-Third IAAI Conference*.

[57] Hanting Xie, Sam Devlin, Daniel Kudenko, and Peter Cowling. 2015. Predicting player disengagement and first purchase with event-frequency based data representation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 230–237.

[58] Pu Yang, Brent E Harrison, and David L Roberts. 2014. Identifying patterns in combat that are predictive of success in MOBA games.. In *FDG*.

[59] Geogios N. Yannakakis. 2012. Game AI Revisited. In *Proceedings of the 9th Conference on Computing Frontiers* (Cagliari, Italy) *(CF '12)*. ACM, New York, NY, USA, 285–292. https://doi.org/10.1145/2212908.2212954

[60] Georg Zoeller. 2010. Development telemetry in video games projects. In *Game developers conference*.