Applying Rapid Crowdsourced Playtesting to a Human Computation Game

Pratheep Kumar Paranthaman
Elon University
pparanthaman@elon.edu

Anurag Sarkar
Northeastern University
sarkar.an@northeastern.edu

Seth Cooper
Northeastern University
se.cooper@northeastern.edu

ABSTRACT

Player engagement and task effectiveness are crucial factors in human computation games. However, collecting data and making design changes towards these goals can be time-consuming. In this work, we incorporate rapid crowdsourced playtesting via the ARAPID (As Rapid As Possible Iterative Design) system to iterate on the design of a human computation platformer game. For each level in the game, the player's goal is to collect items relevant to a given scenario while avoiding irrelevant items. We extended the visualization modules in the existing ARAPID system to include a multi-level data visualization and item collection task effectiveness plot. A designer from the project team used the system to iterate on the game's level design, with the goal of increasing relevant and decreasing irrelevant items collected by players. A large-scale test with the game versions created during the iterative analysis found that the designer was able to use ARAPID to improve the specified goal parameters.

KEYWORDS

human computation, crowdsourcing, playtesting, rapid iteration, game analytics

ACM Reference Format:
Pratheep Kumar Paranthaman, Anurag Sarkar, and Seth Cooper. 2021. Applying Rapid Crowdsourced Playtesting to a Human Computation Game. In *The 16th International Conference on the Foundations of Digital Games (FDG) 2021 (FDG'21), August 3–6, 2021, Montreal, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3472538.3472626

1 INTRODUCTION

Playtesting is an integral part of the game development lifecycle. However, organizing playtests and extracting feedback can be time-consuming due to the set of processes (recruiting users, organizing tests and collecting feedback) involved in the playtesting pipeline. The rise of crowdsourced providers and playtesting platforms [2, 23] has been supportive in organizing online playtesting. These existing online playtesting platforms involve either outsourcing the playtesting task to a third party or conducting the playtests through dedicated crowdsourced platforms (e.g. [2]). This decouples the testing and development phases and as a result, introduces a needless latency in the development cycle.

Previous work developed a prototype system called ARAPID (As Rapid As Possible Iterative Design) [22] to streamline the playtesting process within the Unity game development engine. ARAPID allows game designers to launch user tests, visualize playtest outcomes and make design revisions, all within Unity. In previous work, ARAPID focused only on simple entertainment games (2D and 3D platformers) with a single level.

In this work, we extended the previously developed ARAPID system for use with a human computation game (HCG) and carried out an experiment involving iteratively playtesting the game and evaluating the changes made. The HCG used was *Iowa James*, a platformer HCG where the player collects items relevant to a given scenario and avoids irrelevant items. This work covers:

Extension of ARAPID to a multi-level HCG – Our primary goal was to use ARAPID to address the task effectiveness aspect of HCG design. We implemented three new visualizations in ARAPID: multi-level data visualization, a player item collection task effectiveness plot, and consolidated player trajectories. These were intended to support the game designer to iterate on the design parameters of *Iowa James*.

Iterative analysis – A project member designed three new levels for the game, and another project member used ARAPID to improve the HCG. The goal of the test was to improve the total correctness (relevant minus irrelevant items collected) of the players by determining the best counts and placements of HCG item locations in three levels of the game. The project member used ARAPID to conduct 8 iterations using data from 50 playtesters per iteration.

Large-scale evaluation – After all the iterations, we identified the worst and best versions of the game, based on total correctness, and ran a large-scale test by randomly assigning 400 players to either the first, worst or best version. For the analysis, we considered three parameters from this test: 1) correctness, 2) number of levels completed, and 3) gameplay duration. We observed a statistically significant difference between all three versions in total correctness, with the best version from the iterations also having the best correctness during this test. We also found a difference among all three versions in number of levels completed and a difference between the best version and the other two for gameplay duration.

In this work, we demonstrate the use of ARAPID to improve player effectiveness in a human computation game through several iterations.

FDG'21, August 3–6, 2021, Montreal, QC, Canada

ACM ISBN 978-1-4503-8422-3/21/08…$15.00
https://doi.org/10.1145/3472538.3472626

2 RELATED WORK

2.1 Human Computation Games

Games are increasingly used as a means for human computation. For such games, which must be both engaging to players and effective at contributing to a real-world problem, the design process is especially challenging. An iterative approach involving playtesting can help with this design process [7]. From the initial work of the ESP Game [30], games have been applied in a variety of ways for image labeling, categorization, data mapping, information retrieval, solving scientific puzzles and analysis [6, 17]. Additionally, games have been employed in a broad range of domains for human computation, crowdsourcing, and citizen science. These range from neuron tracing [18] and software verification [19] to capturing facial expressions [27]. Most closely related to the game used in our work, *Gwario* [25] is a platformer in which the player collects items to categorize them into different scenarios. Siu and Riedl [26] created a reward system using a cooking-themed HCG and performed a comparative analysis between the choice of rewards and randomly assigned rewards. The results suggest that giving choice of rewards could provide better task completion and player experience. *Disguise*, a purpose driven game developed by Ahmed et al. [1] facilitated in evaluating visualization algorithms and this also focused on eliminating the latency involved in the development process of visualization research. HCGs are also used for training purposes; e.g. Clark et al. [5], created an HCG to evaluate optical coherence tomography scans for effective diagnosis of patients with Age-Related Macular Degeneration (AMD). Instead of an explicit training on the process, the game mechanics were crafted effectively to support the evaluation process. HCGs not only contribute to solving crucial problems but also act as a vital tool for data mapping and information gathering.

2.2 Playtesting and Game Analytics

Playtesting is a crucial part of the game development process [10] and has thus inspired a large body of work. Zook et al. [32] use an active learning approach to automatically tune in-game parameters to achieve specified design objectives based on gameplay data. Prior work has also looked at improving playtesting for games via utilizing AI-based methods to help automate the process. Holmgard et al. [15] developed psychologically grounded player models using a variant of Monte Carlo Tree Search (MCTS) and used them to automatically evaluate level playability. García-Sánchez et al. [12] used evolutionary algorithms to automate playtesting in *Hearthstone* by evolving different card decks which are then evaluated using the agent. Gudmundsson et al. [13] employed a more general approach to playtesting by training deep neural networks on player data in order to be able to evaluate levels by using predictions of player actions to simulate human behavior. Within the domain of HCGs, Horn et al. [16] proposed a hybrid approach combining MCTS and skill chains to automate playtesting. In their work, they designed AI agents of varying skill levels and used them to analyze the difficulty progression in the puzzle HCG *Foldit*.

Prior work has also looked at different ways of visualizing playtesting data such as by combining quantitative and qualitative data [20] as well as by aggregation [31]. Project Vixen [8] provides a plugin within Unity to manage the visualizations in-editor and also supports dynamic interaction of playtesting data. Representing the datapoints of a large scale data set can be challenging and at times the crowding of data can make the visualizations hard to interpret. Feltwell et al. [9] approached the the issue of data visualization in large and crowded data sets like heatmaps and activity histogram.

Using crowdsourcing for gathering information on user experience was proposed by Birk and Mandryk [4] and there exist many crowdsourced user testing and design platforms. Games for Crowds by Guy et al. [14] enables collaborative design and playing games via the collective intelligence of crowds of users. CrowdStudy by Nebeling et al. [21] is a toolkit for crowdsourced automated usability testing while The UX Crowd [28] is a usability testing platform that combines crowdsourcing with voting-based evaluation.

Game analytics helps designers understand playtesting data patterns, game logs, events, and trends. Companies like GameAnalytics [11] and Playfab [3] offer solutions for analytics and data visualization. Unity has an in-built analytics tool that can offer livestream data and heatmaps [29].

3 OVERVIEW

3.1 ARAPID System

The previously developed ARAPID system (a thorough description can be found in [22]) aims to leverage online crowdsourcing and custom game analytics to extract playtesting feedback instantly and to visualize the data in Unity. The functionality of ARAPID has been wrapped and integrated as a plugin within the Unity game development engine. The system consists of three main components:

Control Panel – a custom interface in Unity through which game designers can launch playtests and visualize the test outcomes in Unity's scene view. The control panel comprises several data visualization options such as player trajectories, game completion ratios, trap markers, and survival analysis plots.

Online Data Coordinator – this module handles game versions and gameplay data in the backend. We used Amazon DynamoDB, a NoSQL database, for storing gameplay data and Amazon S3 for storing game versions.

Recruiter – an online crowdsourced platform used to recruit players and assign them tasks. In this module, we used Amazon Mechanical Turk (MTurk) and integrated the functionalities of it within the control panel through open-source Javascript SDK from Amazon Web Services. This module allows the game designers to configure playtests, launch HITs (Human Intelligence Tasks), and approve HITs directly from the control panel in Unity.

Using ARAPID, the designer performs cyclical steps of edit (design changes and adjustments to the level layout), test (use the updated version of game and conduct a crowdsourced playtest), and analyze (visualize the test outcomes in Unity scene view through several plots in ARAPID). When players complete an assignment (gameplay task) in MTurk, they receive a secret code that can later be used to receive payment. For each assignment in this work, we paid $0.40 regardless of whether the players completed or failed the game. This payment was allocated based on the estimated average gameplay duration of 90 seconds (resulting in $16/hr).

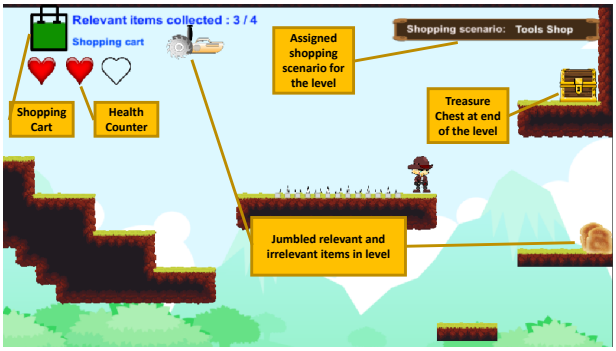

Figure 1: Labeled screenshot of Iowa James HCG.

3.2 Iowa James

The game used in this work, *Iowa James* [24] (see Figure 1), is a platformer based on *Gwario* [25]. Both are similar to *Super Mario Bros.*, but they introduce a human computation aspect by replacing the traditional collectibles with shopping items (e.g. clothes, vegetables, chocolates) and a scenario for each level, thus using gameplay to gather common-sense knowledge about these items based on which ones the player collects. We used three levels from *Iowa James*, designed with increasing complexity such that level 1 comprises no traps, level 2 comprises spikes on various platforms and the ground, and level 3 comprises a combination of spikes and rising spikes on platforms and the ground. In each level, multiple collectible items are placed at various locations and the goal of the players is to collect the relevant items corresponding to the given shopping scenario (level 1 - grocery store; level 2 - clothing store; level 3 - tools shop) and ignore the irrelevant items and reach the treasure chest at the end of the level.

In the version of *Iowa James* used in this work, it was known which items were relevant and irrelevant. Each level in the game has a number of *item locations* where items appear. Each item location is randomly assigned either a relevant or an irrelevant item. The count of total items in the level is an even number so that half of the items in the level are relevant and the remaining half are irrelevant. Players are assigned a shopping cart and a count of relevant items they need to collect for each level. When they fill the cart with the required number of collectibles, a treasure chest at the end of the level unlocks which then leads to the next level. Players have three lives to complete each level. Whenever players step into a trap (spikes and rising spikes) or collect an item irrelevant to the shopping scenario, they lose a life and their current position resets to the start of the particular level. In addition to this, every time a player collects an irrelevant item, the game provides feedback indicating that the item was irrelevant to the current scenario.

3.3 ARAPID Visualizations

Along with the existing visualizations in the ARAPID system, we incorporated three new visualizations for *Iowa James*: task effectiveness plots, multi-level survival analysis plot, and level-wise consolidated player trajectories. These visualizations were created to support the design iterations of this multi-level Iowa James HCG.

Task effectiveness plots – these show the total number of relevant and irrelevant items collected for each item location in all 3 levels.

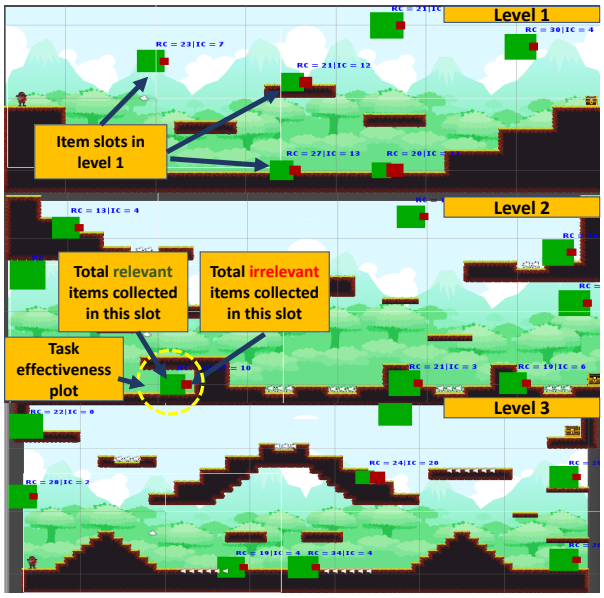

Figure 2: Example of ARAPID's task effectiveness plot which is displayed in each level of the game in Unity's scene view and shows the total relevant (green) and irrelevant (red) items collected by players for each item location.

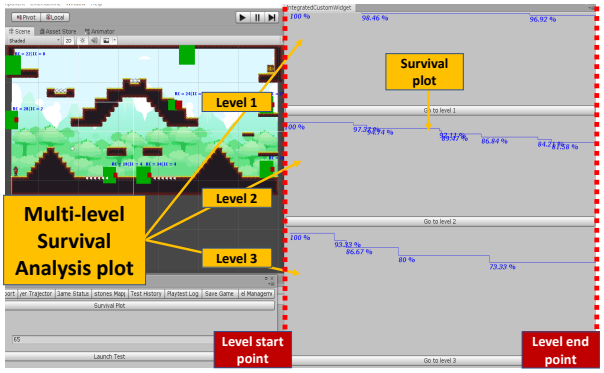

Figure 3: Example of ARAPID's multi-level survival analysis plot.

Green and red boxes are assigned for relevant and irrelevant items respectively. The scale of the boxes depends on the number of relevant items/irrelevant items collected. The designer uses this plot during iterations to determine the task effectiveness of each item location. An optimal outcome is a higher scale size of green boxes compared to red boxes for each item location (see Figure 2).

Multi-level survival analysis plot – a survival analysis plot shows the percentage of players progressing to various points in the level. We created a widget that encompasses the survival analysis plots of all three levels in one place. This plot informs the designer about the percentage of player success on all three levels e.g., in Figure 3 - 96.92% of players completed level 1, 81.58% of players completed level 2, and 73.33% of players completed level 3. Whenever a player loses all their lives in a level, there will be a drop in the plot denoting the decrease in completion proportion.

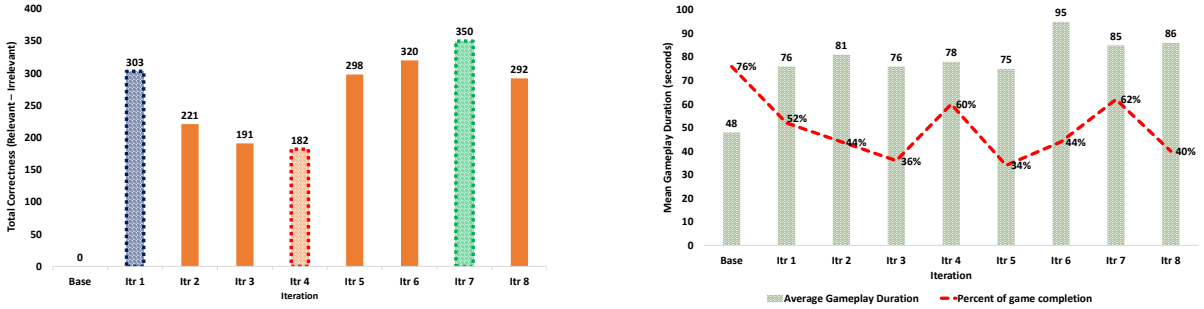

Figure 4: (left) Total correctness for each iteration is the sum of relevant minus irrelevant items collected for all 50 players in the test. Itr 7 (highlighted in green) denotes the best iteration, itr 4 (highlighted in red) denotes the worst, and itr 1 (highlighted in blue) denotes the first. These three iterations were used in the large-scale test for a comparative study. (right) Mean gameplay duration and percentage of players to complete all three levels for each iteration.

	Iteration								
	Base	1	2	3	4	5	6	7	8
Level 1	0	6	4	4	4	10	6	6	6
Level 2	0	8	6	6	4	10	8	8	8
Level 3	0	8	6	6	4	10	8	8	8

Table 1: Number of item locations placed in all three levels in each iteration.

Consolidated player trajectories – this plot shows the overall player trajectories in each level and informs the designer about the paths taken by players in the level. This feature is applicable by opening a particular level and clicking a button on the ARAPID control widget to pull the data, whereas the task efficiency and multi-level survival analysis plots are automated and once the playtest starts they will update every five seconds.

4 ITERATION

4.1 Method

Our aim for system evaluation was to improve player task effectiveness in *Iowa James* by using ARAPID. For this, one of the project members (PM-A) designed three new level layouts with increasing difficulty that had no item locations; a different project member (PM-B) used ARAPID to iterate on item locations and improved task effectiveness in those three levels. PM-B and other project members (other than PM-A) did not have experience with the specific level layouts used in this work before starting iterations. Between some iterations, PM-B would discuss their approach and playtesting results with another project member; however, after designing the levels, PM-A had no involvement with the game design or iterations. These project members are paper authors.

The goal assigned to PM-B was to place collectible item locations in each level and maximize the players' total correctness (relevant minus irrelevant) of all item locations in three levels through iterative process. During iterations, *total* correctness of all players was considered since each playtest had the same number of players.

In each iteration, the test specifications set for PM-B were: 1) add or remove collectible item locations in each level; due to the game's base logic (half of the total items in a level were relevant and the other half were irrelevant) for populating items in level, PM-B was allowed to place only even numbered item locations ranging from 2 (minimum) to 10 (maximum) in each level, 2) use ARAPID visualizations (task effectiveness plot, multi-level survival analysis plot and consolidated player trajectories) and determine ideal positions for item locations in each level, 3) adjust the position of item locations to maximize total correctness, and 4) update the game version after each iteration and upload the new game version for the next iteration. For every iteration, PM-B recruited 50 players per playtest. Due to the fact that the last few assignments can take a disproportionately long time, we recruited 15 additional players for each playtest (i.e. a total of 65 players) [22]. During evaluation of the results, we considered gameplay data of the first 50 users that were recorded. Players who played an iteration were not able to play subsequent iterations.

4.2 Process

A summary of the iterative process is given in Table 1 and Figure 4. PM-B initiated the test with the base level layout and this was considered as iteration 0, because neither the item locations were added nor any changes were made to the levels. PM-B used this iteration to understand how exactly the players were traversing through the three levels and determine item locations in each level for the next iteration. After iteration 0 (base), PM-B started iteration (abbreviated as itr) 1, by adding item locations in all three levels. At the end of the editing phase, PM-B ended up adding 6, 8, and 8 item locations in levels 1, 2, and 3, respectively (abbreviated as 6/8/8). On the subsequent iterations, PM-B decreased the item locations at all levels and as a result of it, total correctness started to reduce on every iteration after itr 1 and outcomes (total correctness) were the worst at itr 4 (with an item location configuration of 4/4/4).

For itr 5, PM-B maximized the item counts compared to the previous iterations and this is to see if increasing item counts could impact the total correctness. In itr 5, PM-B went with 10 items at each level (10/10/10) and at the end of itr 5, PM-B observed that the total correctness drastically improved compared to the previous three iterations (itr 4, itr 3, and itr 2). But still the total correctness was slightly lower than itr 1 (this trend can be seen in Figure 4), leaving itr 1 as the best iteration so far.

As the total correctness was low even after maximum item locations (10/10/10) for each level, PM-B decided to revert back to the item locations configuration used in itr 1 (6/8/8). So, in itr 6,

	Correctness	Level Completion	Gameplay Duration (s)
First (Itr 1)	6	2	66.5
Worst (Itr 4)	5	3	62.0
Best (Itr 7)	9	3	73.0

Table 2: Median values of measures for large-scale test.

	Correctness	Level Completion	Gameplay Duration (s)
Omnibus	$p < .001$ $\chi^2(3) = 42.68$	$p < .001$ $\chi^2(3) = 26.74$	$p < .05$ $\chi^2(3) = 8.98$
First – Best	$p < .001$ *Z = -3.66*	$p < .05$ *Z = -2.30*	$p < .05$ *Z = -2.18*
Worst – Best	$p < .001$ *Z = -6.18*	$p < .05$ *Z = -2.42*	$p < .05$ *Z = -2.90*
First – Worst	$p < .05$ *Z = -3.86*	$p < .001$ *Z = -5.22*	*n.s.* *Z = -0.93*

Table 3: Results of statistical analyses performed for the large-scale test.

PM-B used the same item locations configuration as in itr 1 with slight modifications to the locations of items that received poor correctness scores in three levels in itr 1. Results of itr 6 showed a considerable increase in correctness score, which was higher than the previous iteration (itr 5), but still lower than itr 1.

At this point, PM-B decided to tune the system by keeping the count of item locations to be the same as itr 1, but adjusting the item locations to achieve a higher score than itr 1. Itr 7 involved the same item locations count (6/8/8) with modified item locations. In the result of itr 7, the total correctness was much higher than all of the previous iterations and itr 7 turned out to be the best iteration thus far. PM-B observed that item locations configuration of having the maximum (10/10/10) on all levels was not the best for achieving total correctness. Also, reducing the item location configuration below 6/8/8 would reduce the correctness score as well.

Further, PM-B identified 6/8/8 as the best counts for item locations on respective levels to achieve better correctness score. Since the best case was derived based on the best score achieved in itr 7, PM-B further extended one more iteration with 6/8/8 with slight adjustments to the locations of item locations to cross-validate if the correctness score can be further increased from itr 7 with item locations configuration 6/8/8. PM-B initiated itr 8 with 6/8/8 and a few adjustments to the item locations. The results of itr 8 reduced the correctness score much more than the previous two iterations and this may have been due to the changes of item locations in itr 8. Thus, PM-B concluded the study with 8 iterations and identified itr 4 as the *worst* and itr 7 as the *best* iterations (see Figure 4).

5 LARGE-SCALE EVALUATION

5.1 Method

To get a better comparison of the versions of the game produced during the design iterations, we ran a large-scale evaluation of three versions. In this large-scale test, we included the *first* iteration (itr 1), the *worst* iteration (itr 4), and the *best* iteration (itr 7). Itr 1 was the first version that PM-B actually started from the base version, and after itr 1, the correctness score started to decrease until itr 4 so we wanted to investigate the difference involved between first (itr 1), worst (itr 4) and best (itr 7) cases. For this test, we uploaded the three game versions (*first, worst, and best*) to Amazon S3. We kept the recruitment count to 400 players.

5.2 Analysis

The 400 players were randomly assigned to one of the three versions (first, worst, and best). 156 users received the first version, 130 users received the worst version, and 114 users received the best version. After completion of the large-scale test, we conducted comparative statistical tests between the three versions. For statistical analysis, we looked at the per-player correctness (relevant minus irrelevant items collected), and to additionally examine gameplay measures of engagement, we considered number of levels completed and gameplay duration. We compared these three parameters between the three versions of the game (first, worst, and best).

First, we conducted a Shapiro-Wilk test for normality and Levene's test for homogeneity of variances. The normality and homogeneity of variances have been violated ($p < 0.05$) in correctness (Shapiro-Wilk Test: W(400) = 0.960, $p < 0.001$; Levene's test: F(2,397) = 61.89, $p < 0.001$) and number of levels completed (Shapiro-Wilk test: W(400) = 0.778, $p < 0.001$; Levene's test: F(2,397) = 21.52, $p < 0.001$). Whereas in case of gameplay duration, normality has been violated (Shapiro-Wilk test: W(400) = 0.750, $p < 0.001$), but the Levene's test for homogeneity of variances turned out to be non-significant (F(2,397) = 1.35, $p > 0.05$).

Next, we conducted an Omnibus Kruskal-Wallis test on total correctness, number of levels completed and total gameplay duration and found differences across versions to be statistically significant. We then conducted a pairwise Wilcoxon rank sum test for post-hoc analysis with the Bonferroni correction to investigate the actual difference between groups. We observed a statistical significance between all three versions (first, worst, and best) in correctness and number of levels completed. In gameplay duration, statistical significance was observed only between first (itr 1) with best (itr 7) and worst (itr 4) with best (itr 7), and comparison between first (itr 1) with worst (itr 4) turned out to be non-significant (see Table 3).

Table 2 shows the median values of correctness, number of levels completed and gameplay duration of three versions used in large-scale test and figure 5 shows box plots. There were some outliers in gameplay duration due to the fact that no additional players were recruited, so the last couple of players took much longer to complete the tests. Also, the patterns observed between three versions in the correctness box plot denote the level of outcomes variations involved between the three versions. This analysis supports that the different iterations had different correctness distribution; we found the best iteration (itr 7) had the highest median correctness out of the three versions in the large-scale test.

6 DISCUSSION

Iterative analysis – Our primary goal was to explore the use of rapid iteration via ARAPID to improve task effectiveness in *Iowa James* by iterating on the game's design starting from its base version. Thus our major focus during iterations was on total correctness.

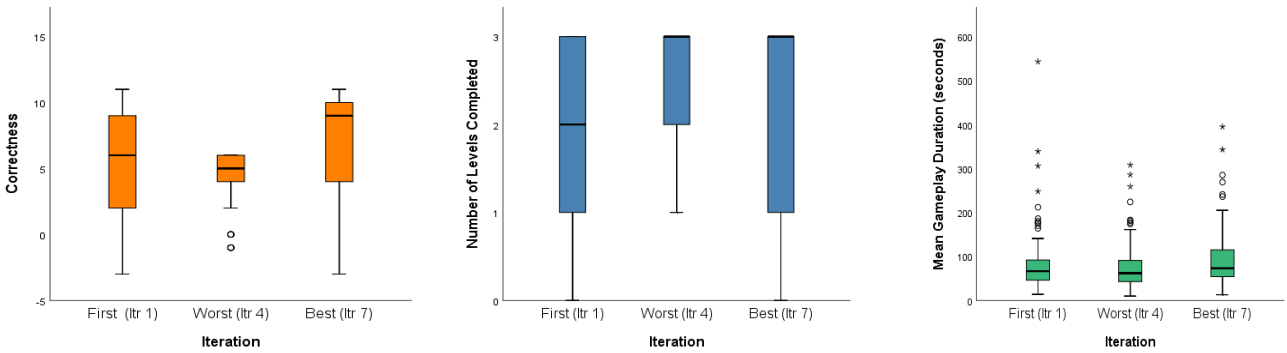

Figure 5: Plots of data from large-scale test, showing quartiles and outliers. (left) correctness; (middle) level completion; (right) gameplay duration.

We observed that when reducing the item locations in each level, the total correctness started to decrease, as depicted in Figure 4. PM-B started the iteration with 6/8/8 item locations in the three levels, and on subsequent iterations, these numbers were reduced. As an effect of this, correctness score constantly decreased from itr 1 and finally landed at itr 4 (with 4/4/4 item locations) which had the worst score compared to all previous iterations. In fact, itr 4 had the worst correctness score during iterations as shown in Figure 4. This makes sense as there were simply fewer items to collect in the iterations with fewer item locations.

After itr 4, item locations were increased or were kept the same as itr 1 (from itr 5 to itr 8), so the correctness score started to increase from itr 5 to itr 7. In itr 5, the item locations were raised to the maximum on all levels (10/10/10 item locations), resulting in a drastic increase in correctness in itr 5 immediately after itr 4; however, itr 5 still had lower correctness than itr 1 during playtesting, which indicated that maximizing the number of item locations may not be the best strategy for improving correctness. The best iteration (itr 7) had the same number of item locations as itr 1, but with some changes to the locations of the item locations which had lower correctness in itr 1. Thus, during iterations, it appeared that not only the number of item locations matter but also their location in the level affect correctness. If some items are placed in certain locations in levels, it may be hard for players to collect or avoid them (e.g. needing to go far out of their way to collect a relevant item, or having to land on an item to complete a jump making it hard to avoid an irrelevant item), which would impact the correctness in a negative way. An ideal location for an item location should be the one in which the players should have control to collect or avoid an item as needed. Thus, it is essential for the designer to use the three visualizations in ARAPID and determine good locations for items in the iterative process. During iterations it appeared that: 1) decreasing the item locations too much would decrease correctness, 2) simply increasing item locations to the maximum number would not necessarily yield better correctness, and 3) a balance of number of item locations and good locations could increase correctness.

In this game, a player's lives is affected by two factors: 1) collecting irrelevant items, and 2) stepping into traps, specifically in level 2 and level 3. These two factors can cause a player to lose the game. We computed the percentage of players who completed the entire game (all three levels) in all iterations. Figure 4 shows that in the base version of the game, 76% of players completed the entire game. Despite there being no items locations in the base version, 24% of players failed in level 2 and level 3 because of running into traps. Adding up item locations started to reduce the game completion percentage as shown in Figure 4, with the completion ratio from itr 1 to itr 8 ranging between 34% – 62%, which was lower than the base version (76%). The base version had higher game completion percentage denoting the fact that having no item locations in levels would increase the chance of game completion, as there is no crucial task involved in the game other than evading the obstacles. Whereas in itr 1 to 8, players not only had to manage to collect only relevant items, but also avoid the traps, and this combination induced more game complexity and affected the overall game completion percentage.

Itr 4 had the fewest item locations in the entire iteration series, and an increase in game completion percentage was noted in itr 4 when compared to three previous iterations (itr 3, itr 2, and itr 1). In contrast to this, itr 5 had the most item locations at all levels (10/10/10 item locations), so the game completion percentage recorded in this iteration was 34% and this was the lowest out of all the iterations. This aspect denotes that setting up the maximum item count on all levels could reduce the game completion percentage. In itr 7, the item locations were adjusted to provide more control for users to collect or avoid an item compared to all of the previous iterations, so this had an effect on game completion percentage. It can be noted from Figure 4 that itr 7 not only had the highest correctness score but also had a better game completion percentage (62%) compared to the itr 1.

Large-scale evaluation - Looking at correctness, there was a statistical significance between all three versions tested (first, worst, best). Although the first version produced during iteration was fairly good in terms of correctness, further iteration to explore the design space was able to improve on this in a later iteration (best).

We also conducted an analysis on number of levels completed by each player in the three versions of the large-scale test. There was a statistical significance observed between all three versions.

Table 2 shows that the median number of levels completed in the worst (itr 4) and best (itr 7) versions is 3, and this denotes that most players completed the entire game in these two specific iterations, while for the first version (itr 1) the median is 2. This also suggests that reducing number of items and adjusting the item locations can improve the level and game completion ratios.

Additionally, we conducted a comparison of gameplay durations between the three versions in the large-scale test. A statistical significance was observed between first (itr 1) and best (itr 7) and between worst (itr 4) and best (itr 7), and no statistical significance was observed between first (itr 1) and worst (itr 4).

7 CONCLUSION

In this work, we extended the existing ARAPID system to be applied to a human computation game and demonstrated that through rapid iteration on base game design, ARAPID can be used to successfully improve the effectiveness of an HCG. Existing playtesting frameworks involve either outsourcing of user tests to a third-party or executing the tests completely outside the development environment on crowdsourcing platforms. The ARAPID system manages to recruit users, organize playtests, and inform the game analytics directly within the development environment. This system facilitates the rapid closure of iterative cycles involved in the development pipeline. Previous work [22] tested the ARAPID system with simple 2D and 3D platform games with only one level in each game. In this work, the extended features of the existing ARAPID system were used to improve the task effectiveness in a multi-level HCG.

For future work, we plan to include an additional analysis by allowing the designer to adjust more aspects of the levels, such as trap positions in addition to the item locations and targets to maximize the game completion percentage along with correctness. Another limitation of the current work is that the entire framework was tested with one project member; thus in the future, we plan to explore recruiting multiple game designers and conducting a comparative analysis between the approaches used by different designers to iterate on game designs using ARAPID.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant no. 1652537.

REFERENCES

[1] N. Ahmed, Z. Zheng, and K. Mueller. 2012. Human Computation in Visualization: Using Purpose Driven Games for Robust Evaluation of Visualization Algorithms. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2104–2113.

[2] Amazon Mechanical Turk. 2019. MTurk homepage. https://www.mturk.com/.

[3] Azure Playfab. 2019. Azure Playfab homepage. https://playfab.com/.

[4] Max Birk and Regan Mandryk. 2016. Crowdsourcing Player Experience Evaluation. In *GDC Games User Research Summit 2016*. San Francisco, CA, USA.

[5] C. Clark, M. Ouellette, and K. Csaky. 2019. Training Players to Analyze Age-Related Macular Degeneration Optical Coherence Tomography Scans using a Human Computation Game. In *2019 IEEE 7th International Conference on Serious Games and Applications for Health*. 1–7.

[6] Seth Cooper, Firas Khatib, Ilya Makedon, Hao Lu, Janos Barbero, David Baker, James Fogarty, Zoran Popović, and Foldit players. 2011. Analysis of Social Gameplay Macros in the Foldit Cookbook. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. 9–14.

[7] Seth Cooper, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, David Salesin, David Baker, Zoran Popović, and Foldit Players. 2010. The challenge of designing scientific discovery games. In *Proceedings of the 5th International Conference on the Foundations of Digital Games*. ACM, 40–47.

[8] Brandon Drenikow and Pejman Mirza-Babaei. 2017. Vixen: interactive visualization of gameplay experiences. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 3:1–3:10.

[9] Tom Feltwell, Grzegorz Cielniak, Patrick Dickinson, Ben J. Kirman, and Shaun Lawson. 2015. Dendrogram Visualization as a Game Design Tool. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. 505–510.

[10] Tracy Fullerton. 2008. *Game design workshop: a playcentric approach to creating innovative games*. Morgan Kaufmann.

[11] GameAnalytics. 2019. GameAnalytics homepage. https://gameanalytics.com/.

[12] Pablo García-Sánchez, Alberto Tonda, Antonio M. Mora, Giovanni Squillero, and Juan Julián Merelo. 2018. Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone. *Knowledge-Based Systems* 153 (2018), 133–146.

[13] Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. 2018. Human-like Playtesting with Deep Learning. In *IEEE Conference on Computational Intelligence in Games*.

[14] Ido Guy, Anat Hashavit, and Yaniv Corem. 2015. Games for Crowds: A Crowdsourcing Game Platform for the Enterprise. In *Games for Crowds: A Crowdsourcing Game Platform for the Enterprise*.

[15] Christoffer Holmgard, Michael Cerny Green, Antonios Liapis, and Julian Togelius. 2018. Automated Playtesting with Procedural Personas wit Evolved Heuristics. In *IEEE Conference on Computational Intelligence in Games*.

[16] Britton Horn, Josh Aaron Miller, Gillian Smith, and Seth Cooper. 2018. A Monte Carlo Approach to Skill-based Automated Playtesting. In *AIIDE*.

[17] Firas Khatib, Seth Cooper, Michael D. Tyka, Kefan Xu, Ilya Makedon, Zoran Popović, David Baker, and Foldit Players. 2011. Algorithm discovery by protein folding game players. *Proceedings of the National Academy of Sciences* 108, 47 (2011), 18949–18953.

[18] Jinseop S. Kim, Matthew J. Greene, Aleksandar Zlateski, Kisuk Lee, Mark Richardson, Srinivas C. Turaga, Michael Purcaro, Matthew Balkam, Amy Robinson, Bardia F. Behabadi, Michael Campos, Winfried Denk, H. Sebastian Seung, and EyeWirers. 2014. Space-time wiring specificity supports direction selectivity in the retina. *Nature* 509, 7500 (May 2014), 331–336.

[19] Heather Logas, Jim Whitehead, Michael Mateas, Richard Vallejos, Lauren Scott, Dan Shapiro, John Murray, Kate Compton, Joseph Osborn, Orlando Salvatore, Zhongpeng Lin, Huascar Sanchez, Michael Shavlovsky, Daniel Cetina, Shayne Clementi, and Chris Lewis. 2014. Software verification games: designing Xylem, The Code of Plants. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*.

[20] Pejman Mirza-Babaei, Gunter Wallner, Graham McAllister, and Lennart E. Nacke. 2014. Unified Visualization of Quantiative and Qualitative Playtesting Data. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*.

[21] Michael Nebeling, Maximilian Speicher, and Moira C. Norrie. 2013. CrowdStudy: General Toolkit for Crowdsourced Evaluation of Web Interfaces. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*.

[22] Pratheep Kumar Paranthaman and Seth Cooper. 2019. ARAPID: towards integrating crowdsourced playtesting into the game development environment. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 121–133.

[23] PlaytestCloud. 2019. PlaytestCloud homepage. https://www.playtestcloud.com/.

[24] Anurag Sarkar and Seth Cooper. 2019. Using a disjoint skill model for game and task difficulty in human computation games. In *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. 661–669.

[25] Kristin Siu, Matthew Guzdial, and Mark O. Riedl. 2017. Evaluating singleplayer and multiplayer in human computation games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 1–10.

[26] Kristin Siu and Mark O. Riedl. 2016. Reward Systems in Human Computation Games. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. 266–275.

[27] Chek Tien Tan, Hemanta Sapkota, and Daniel Rosser. 2014. BeFaced: a casual game to crowdsource facial expressions in the wild. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. 491–494.

[28] The UX Crowd. 2020. The UX Crowd homepage. https://www.trymyui.com/site/the-ux-crowd.

[29] Unity Analytics. 2020. Unity Analytics homepage. https://docs.unity3d.com/Manual/UnityAnalytics.html.

[30] Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Vienna, Austria, 319–326.

[31] Gunter Wallner, Nour Halabi, and Pejman Mirza-Babaei. 2019. Aggregated visualization of playtesting data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*.

[32] Alexander Zook, Eric Fruchter, and Mark Riedl. 2019. Automatic Playtesting for Game Parameter Tuning via Active Learning. In *arXiv preprint arXiv:1908.01417*.