RITE Method on *Revision*

Abstract

This research examines running a variation of the RITE method on a game in development. Despite testing being run concurrently to development, the RITE method, after slight tweaks, still improve the usability of the game. Specifically, the amount of problems still decreased over time. The amount of new problems discovered also decreased over time, causing most of the problems encountered later on to be repeated problems. Finally, results of fixing the problems showed lower impact ratio than traditional but still maintained low re-fix ratio.

Introduction

Steve Cha, a third year MFA, is making a game for his thesis project called *Revision*. This is a metafiction narrative game, where the player plays through mini-game levels which are vignettes of his life. After the first playthrough, the player is taken back to a hub world, where she is given the power to go back and revisit the vignettes again, but this time with additional mechanics that reveal deeper meanings behind each level. However, problems arise with the introduction of the second set of mechanics, as players get stuck because of problems in the game system. The design team wants to know what usability problems prevent players from completing the game, and what changes can they make to fix them. And so, the RITE method is used as a technique to answer these questions. The RITE method is defined in "Using the RITE Method to Improve Products; a Definition and a Case Study", and slight adjustments were made to its methods to better fit the design team's needs. This paper will describe the methods and showcase the results from the usability tests.

Methods

The traditional RITE method was followed rather loosely for this study. First, the players were not given a clear set of tasks to complete on paper. Instead, players are placed in the game, where they had to finish a set of levels, each with a series of objectives for the player to complete. In the first half of the game, the player plays through the levels. In the second half, players collect notes hidden throughout each level. The game should communicate clearly to the players on what to do, and this aspect was tested also.

For each player session, the testing team observed the player playing the game, stepping in only when the team determines that the player could no longer continue on her own. The players were also asked to think out loud and so the testing team could record their thought processes. After each test, the collected data is presented to the design team, who made decisions on which problems to address by the next iteration, which occurs a week after the previous session. The decision on which problems to address is based on the ease of fixing and the impact of the problem on the user experience.

The primary source of data was tester's observations of players. Specifically, the amount of problems that arose during testing was measured. And because not all problems are created equal, they are divided into four categories for better analysis; the last two are added in addition to what the traditional RITE method defines:

- Error. Errors are incorrect player behaviors when completing a task.
- Failure. Failures are when players give up on completing a task, generally due to frustration.
- Complaint. This is when players point out problems during think-out-loud, but does not cause incorrect player behavior.
- Missed. Sometimes players finish the game without ever knowing a task exists, due to the game not communicating correctly.

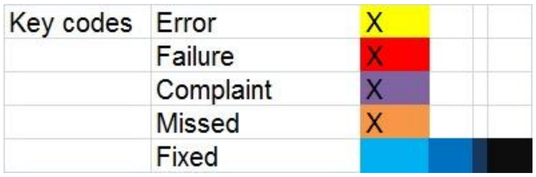
The testing team consists of 2 observers and a tester, all of whom are from the design team. They are present in person and taking notes for the entire duration of each task. Desktop video recording of each player's playthrough is also recorded for detailed post mortem analysis.

The biggest deviation of this experiment from the traditional RITE method was that development was happening alongside testing, due to time constraints. Because of this, the amount of time between each iteration was stretched to one week instead of one to two days. Also, the recorded data are influenced by the fact that new levels and mechanics were introduced in between testing iterations, which in turn introduce new problems. So at first glance, the number of problems over time is increasing. This deviation must be accounted for during analysis. In the end, over the course of seven weeks, seven usability tests were conducted, each with a week in between for fixing issues.

Results

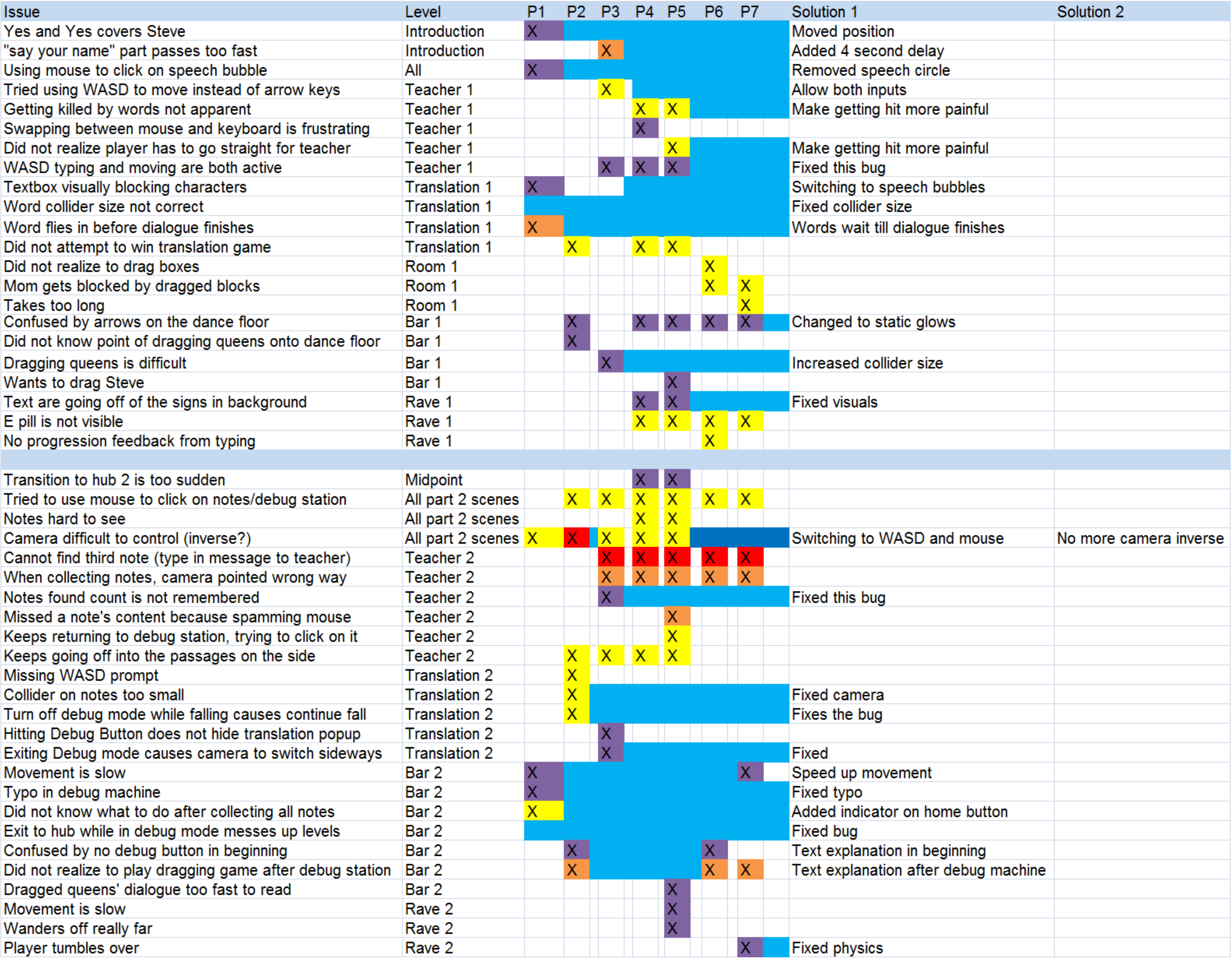
The data points from the tests are collected in Table 2. Table 1 contains the color keys for what is going on in Table 2.

Table 1. What the colors represent in Table 2.



Key codes	Error	X			
	Failure	X			
	Complaint	X			
	Missed	X			
	Fixed	🟦	🟦	🟦	⬛

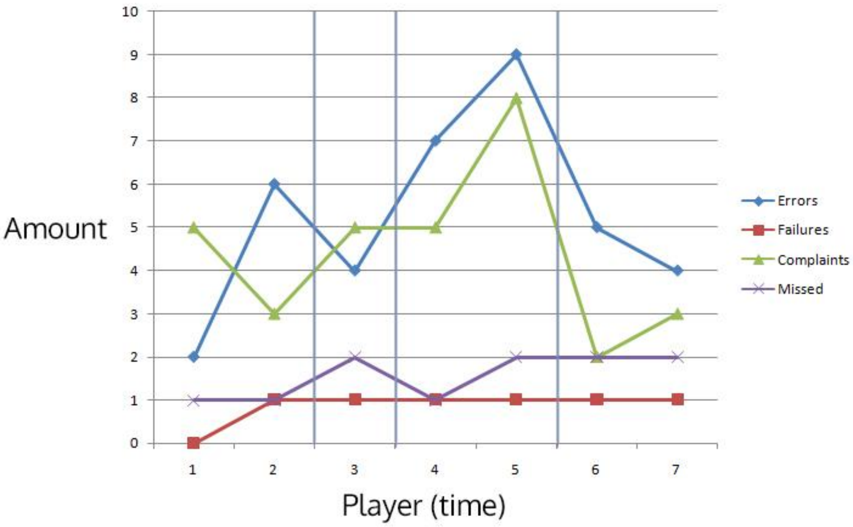
Table 2. Datapoints of problems encountered during testing by 7 players.



Issue	Level	P1	P2	P3	P4	P5	P6	P7		Solution 1	Solution 2
Yes and Yes covers Steve	Introduction	X	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Moved position	
"say your name" part passes too fast	Introduction			X	🟦	🟦	🟦	🟦	🟦	Added 4 second delay	
Using mouse to click on speech bubble	All	X	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Removed speech circle	
Tried using WASD to move instead of arrow keys	Teacher 1			X	🟦	🟦	🟦	🟦	🟦	Allow both inputs	
Getting killed by words not apparent	Teacher 1				X	X	🟦	🟦	🟦	Make getting hit more painful	
Swapping between mouse and keyboard is frustrating	Teacher 1				X						
Did not realize player has to go straight for teacher	Teacher 1					X	🟦	🟦	🟦	Make getting hit more painful	
WASD typing and moving are both active	Teacher 1			X	X	X	🟦	🟦	🟦	Fixed this bug	
Textbox visually blocking characters	Translation 1	X			🟦	🟦	🟦	🟦	🟦	Switching to speech bubbles	
Word collider size not correct	Translation 1	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Fixed collider size	
Word flies in before dialogue finishes	Translation 1	X	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Words wait till dialogue finishes	
Did not attempt to win translation game	Translation 1		X		X	X					
Did not realize to drag boxes	Room 1						X				
Mom gets blocked by dragged blocks	Room 1						X	X			
Takes too long	Room 1							X			
Confused by arrows on the dance floor	Bar 1		X		X	X	X	X	🟦	Changed to static glows	
Did not know point of dragging queens onto dance floor	Bar 1		X								
Dragging queens is difficult	Bar 1			X	🟦	🟦	🟦	🟦	🟦	Increased collider size	
Wants to drag Steve	Bar 1					X					
Text are going off of the signs in background	Rave 1				X	X	🟦	🟦	🟦	Fixed visuals	
E pill is not visible	Rave 1				X	X	X	X			
No progression feedback from typing	Rave 1						X				
Transition to hub 2 is too sudden	Midpoint				X	X					
Tried to use mouse to click on notes/debug station	All part 2 scenes		X	X	X	X	X	X			
Notes hard to see	All part 2 scenes				X	X					
Camera difficult to control (inverse?)	All part 2 scenes	X	X	X	X	X	🟦	🟦	🟦	Switching to WASD and mouse	No more camera inverse
Cannot find third note (type in message to teacher)	Teacher 2			X	X	X	X	X			
When collecting notes, camera pointed wrong way	Teacher 2			X	X	X	X	X			
Notes found count is not remembered	Teacher 2			X	🟦	🟦	🟦	🟦	🟦	Fixed this bug	
Missed a note's content because spamming mouse	Teacher 2					X					
Keeps returning to debug station, trying to click on it	Teacher 2					X					
Keeps going off into the passages on the side	Teacher 2		X	X	X	X					
Missing WASD prompt	Translation 2		X								
Collider on notes too small	Translation 2		X	🟦	🟦	🟦	🟦	🟦	🟦	Fixed camera	
Turn off debug mode while falling causes continue fall	Translation 2		X	🟦	🟦	🟦	🟦	🟦	🟦	Fixes the bug	
Hitting Debug Button does not hide translation popup	Translation 2			X							
Exiting Debug mode causes camera to switch sideways	Translation 2			X	🟦	🟦	🟦	🟦	🟦	Fixed	
Movement is slow	Bar 2	X	🟦	🟦	🟦	🟦	🟦	X		Speed up movement	
Typo in debug machine	Bar 2	X	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Fixed typo	
Did not know what to do after collecting all notes	Bar 2	X	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Added indicator on home button	
Exit to hub while in debug mode messes up levels	Bar 2	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	Fixed bug	
Confused by no debug button in beginning	Bar 2		X	🟦	🟦	🟦	X			Text explanation in beginning	
Did not realize to play dragging game after debug station	Bar 2		X	🟦	🟦	🟦	X	X		Text explanation after debug machine	
Dragged queens' dialogue too fast to read	Bar 2					X					
Movement is slow	Rave 2					X					
Wanders off really far	Rave 2					X					
Player tumbles over	Rave 2							X	🟦	Fixed physics	

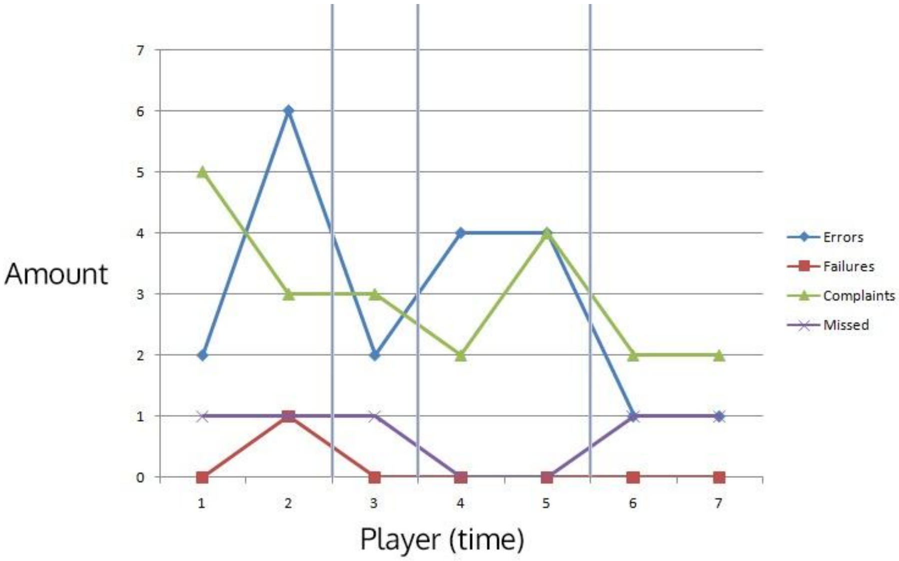
The datapoints are summarized in Figure 1, which counts up all four types of problems over time. Changes were implemented between each iteration.

Figure 1. Amount of problems over time for all levels.



From Figure 1, an interesting trend can be seen: the amount of problems seems to be increasing. This is due to new levels, and thus new problems, being introduced before player 3, 4, and 6. Because of this, it is more insightful to look only at the amount of problems encountered in the Translation and Bar levels, which are the only two levels that have been consistently in every single test session. The graph that represents this can be seen in Figure 2.

Figure 2. Amount of problems over time for the two consistent levels.



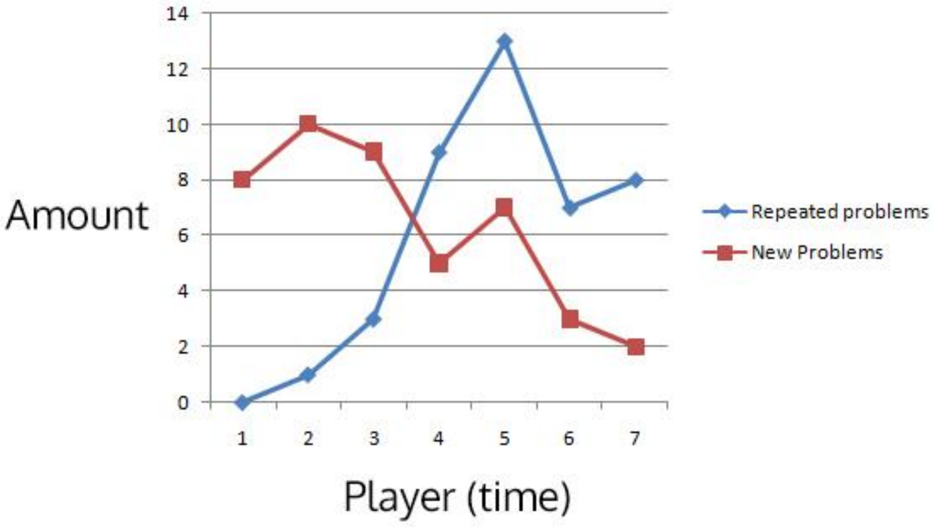
From Figure 2, it can be seen that there is a general trend of decline in both errors and complaints count, while the missed and failures count are relatively stable. Another interesting point to note is that for the last two players, they actually have the exact same count in types of problems. This could represent that the two players encountered the exact same problems, which could be great if the problems were not fixed readily on time and thus makes sense for more players to encounter them. But if they are running into different problems, then the design team would have a lot more work on hand.

In order to examine this deeper, two more definitions need to be established:

- Repeated problem - for a problem that a player encounters, a previous player has encountered it before
- New problem - This is the first player to see this problem

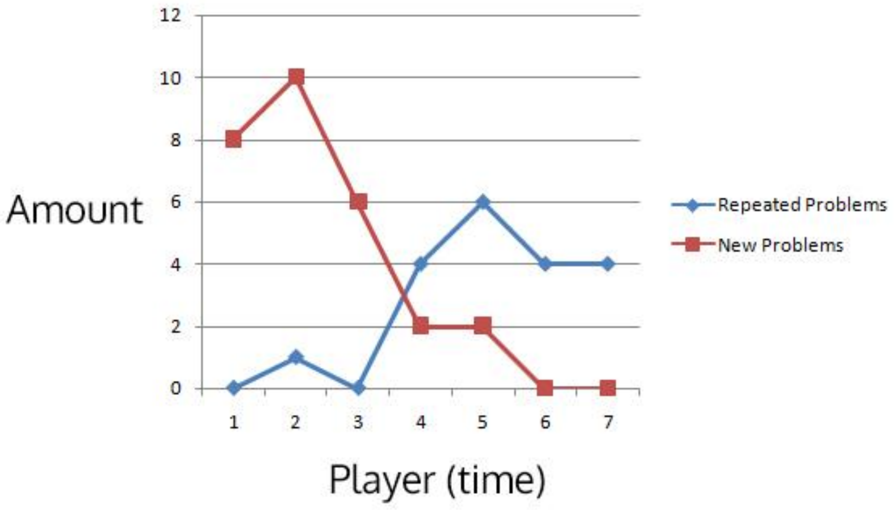
Figure 3 counts up the amount of repeated problems and new problems over time for all levels.

Figure 3. Amount of repeated and new problems over time for all levels.



The amount of repeated problems has a trend towards up over time, while the amount of new problems are decreasing. Looking back at Table 1, it can be seen that for the last two players, most of their repeated problems are ones that the design team has not yet been able to address yet. From this, it can be drawn that over time, the types of problems encountered during testing is mostly repeated problems. To double check this with a traditional RITE method, Figure 4 counts up the amount of repeated problems and new problems over time for only the two levels that were consistent throughout all sessions.

Figure 4. Amount of repeated and new problems over time for the two consistent levels.



The same trend from Figure 3 can also be observed in Figure 4. In fact, it may be argued that this trend is more pronounced, because the amount of new problems encountered for the last two players is zero.

Another point to note is that for the last two players, they encountered the same number of repeated problems, and both do not run into new problems. This is good news for the design team, as players are not stumbling into brand new problems that need to be addressed.

To analyze the effectiveness of the usability tests on fixes that the design team implemented, impact ratio is calculated and examined in Table 3.

Table 3.

Problems found	**Problems that received fix**	**Impact ratio**
47	25	0.531915

The impact ratio is rather low for a RITE study, as *Age of Empires II*'s tutorial has 97%, as opposed to 53% here. This is due to development of the game running concurrently with fixing, so the design team was unable to address many of the problems recorded from the usability tests.

Another ratio to examine the effectiveness of the usability test is to look at the re-fix ratio, calculated in Table 4.

Table 4.

All fixes	Changes that need re-fixing	Re-fix ratio
26	4	0.153846

The re-fix ratio is only 15%, which is pretty low, compared to *Age of Empires II*'s re-fix ratio of 20%. This shows that of the problems that the design team chose to address, they addressed it well.

Discussion

However, a pessimistic view on why re-fix ratio is so low could be explained by the design team cherry-picking the easy problems to fix, and not addressing the harder problems that require more trial and error. This could be true given that the impact ratio is so low. Fortunately, examinations of the datapoints show that most of the recurring unfixed problems are level-specific, so not fixing them will not greatly impact the system of the game as a whole and introduce new problems.

Conclusion

Despite the slight deviations from the traditional RITE method, which includes defining new types of problems and conducting tests alongside development, the results are still in line with the results of traditional RITE method. In particular, the tests have also resulted in the betterment of the game, as problem count are decreasing over time, and the amount of new problems players encounter are decreasing.

However, because development happened in parallel to testing, fixes were not implemented as well as in traditional RITE studies. But despite many of the problems remaining not fixed, they are generally narrow in impact so any fixes to them will not introduce too many new problems.

As of this very moment, the game is still not complete. The levels however are all finished, so the testing team can rely on a more traditional RITE method to conduct usability tests and perform fixes until the deadline of the game, as the process has been shown to be effective through the results.

References

Using the RITE Method to Improve Products; a Definition and a Case Study - Microsoft