Functional and non-functional requirements for a player performance dashboard in CS:GO

Nikola Martino
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
n.martino@student.utwente.nl

ABSTRACT

This paper lays the foundation for a dashboard that visualizes information and statistics for player improvement in CS:GO. Using state of the art literature and data analysis, the functional and non-functional requirements were extracted. Based on these requirements, a prototype dashboard was created.
The dashboard was tested and validated by professional CS:GO players, who provided feedback on the functional and non-functional requirements portrayed in the prototype dashboard.
The dashboard fulfilled the posed needs of professional CS:GO players for a performance feedback tool.
Based on the findings of this paper, further research on the usefulness of performance dashboards and data-driven feedback & improvement systems for CS:GO, could be conducted.

Keywords

Counter Strike, esports, Player Performance, Post-game Review, Real-time Review, Game Data Visualization.

1. INTRODUCTION

The competitive scene of computer games, form what is known as eSports, or also known as electronic sports. eSports have greatly grown in popularity [23] in the past decade, with *hundreds* of tournament games organized and *millions* of dollars in prize pools. It is a growing industry that only attracts more viewers and attention each year [23].

eSports most often consist of a group of players, forming a team and playing together to reach a common goal [7]. eSport teams are similar to traditional sports teams and require teamwork, communication, and coordination [7].

As in regular physical sports, a team's overall performance greatly depends on its members' individual and collective performance; how well they perform when playing alone and together [17]. Similar to traditional sports teams, there exist coaches that evaluates the teams' overall performance, motivate players, promote team spirit and cooperation [9]. Coaches have the responsibility to collect common strategies and performance statistics from opposing teams, by reviewing video replays of matches and individual performance of opposing players [15].

This paper will be mostly focused on one specific eSports game, namely Counter Strike Global Offensive (also known as CS:GO), a First Person Shooter game. CS:GO is the most popular (with the highest number of concurrent users) First Person Shooter game at the time of this paper, according to the Steam listing [20]. This allows the research to, at points, relate to other FPS games, which use CS:GO as a role model [18]. CS:GO provides data per played match, in the form of replays. Counter-Strike Global Offensive replays are match recordings, stored after the game has ended and that contain all the information per second of the game. The replay stores player positioning on the map throughout the lasting of the match, player statistics (kills, average headshot rate, and so on) per round [2].

CS:GO heavily relies on teamwork. Each member has a specific role within the team. We consider two main categories, informal roles [5] and formal roles [19]. The latter roles are universal and apply to all CS:GO teams. As in any other team, roles are important and directly affect team performance [3]. However, as CS:GO relies on teamwork, team performance is also affected by each member individually and collectively. Individual player's performance and match replays are what coaches look at and work with to build strategies and maximize team cooperation and coordination [9].

A player's game experience is not the only performance attribute that affects match outcome, but so do the player's team effectiveness, personality and skills [6].

Automation that makes possible the combination of player's individual attributes with the game data stored in a match replay, allows for a more compact overview of all game parameters and player attributes. This would ease the process of manually searching through this data, which is more time-consuming [22].

2. PROBLEM

This paper aims to answer the following question: "What are the functional and non-functional requirements for a post-game review dashboard of player performance?".

We focus on post-game as there are little possibilities for real-time data processing, and it is not quite feasible by players to focus on anything but the match they are playing. Major eSports leagues do not allow use of any external type of software that affects player performance in any way, as it will most likely result in a game ban [13].

Most of the existing dashboards are of commercial use and do not provide any self-improvement statistics to the user, or offer only overall match statistics. In order to tackle self improvement, we want to create a system that

35^{th} *Twente Student Conference on IT* July 2^{nd}, 2021, Enschede, The Netherlands.

provides feedback to the user, based on the matches they have played.

"We all need people who give us feedback. That's how we improve." - Bill Gates.

With such a system in mind, we split the main research question into three sub-questions:

- **Question 1:** What are the functional requirements for a player performance dashboard?
- **Question 2:** What are the non-functional requirements for a player performance dashboard?
- **Question 3:** To what extent can the use of the player performance dashboard, theoretically, improve general player performance?

3. RELATED WORK

Related papers were found using the following keywords: "fps game player performance; esports player performance; cs:go gameplay; eSport roles; eSport data analysis; coach importance in teams; team players; team efficiency" and by digging through references.

Currently, there are tools and websites that provide real-time match updates, such as HUDStats and GameScore-Keeper. Both sites allow API (application programming interface, that allows the communication between two platforms) connectivity, for a fee, that ease the creation of a dashboard, as data will no longer have to be processed by the dashboard creators, but can be fed from these sites. Usage of API, however, offers less manageability, as one can only use data made available from these sites. Reliability is also another issue that may appear, as the hosting of these providers may not be available around the clock, may experience technical issues, or the data itself may not be entirely accurate [24].

Research has also been done on player reaction speed and how it applies to in-game performance. A setup that allows such testing opportunities is possible [21], with a small delay in data collection that can be adjusted post-analysis. One setup involved examining eye movement and on-screen positioning, along with the most commonly used keys, making a distinguishment in player knowledge and experience [12]. However, such data would not be beneficial when comparing players of the same league, as the outcome would be similar in almost all cases [12].

4. METHODOLOGY & APPROACH

This paper is based on Mader's & Eggink's development framework, based on three iterative phases:

- Ideation: Writing on paper of the basic dashboard requirements, both functional and non-functional.
- Specification: Writing on paper of the user cases and must have's of the dashboard (answering RQ 1 & 2).
- Relization: The creation of the dashboard prototype with the requirements from the specification.
- Evaluation: Running a quantitative and qualitative user testing, to evaluate the usefulness of the dashboard (answering RQ 3).

To begin with the ideation process, we must initially extract all possible data from a CS:GO match. These procedures are further explained in the following sections.

4.1 Data Collection & Processing

CS:GO calls 64 updates per second, which means, all the data, is stored at least 64 times per second. A match usually lasts longer than 30 minutes (1'800 seconds), thus, we have a minimum of 115'200 data entries to look at (30 minutes * 60 seconds * 64 calls per second). This is only for a single match. Due to such an insane amount of data, it is not possible to store it in a database, without wasting multiple megabytes of space.

CS:GO, however, makes it possible for users to download the demo file of a match. Demo files are the post-recordings of a match, containing all information about the match itself.

To process the data, we parse the demo file content. To achieve this, CSGO Demos Manager [10] was used and made it possible to have an in-depth view of the content of a replay.

From the extracted demo content data, two data categories stood out:

- Directly compiled data: consisting of player and overall match data at the end of the match (kills, deaths, assists, average damage per round (ADR), kill/death ratio (KD), kills per round (kpr), enemies flashed (ef), utility damage (damage you cause with utilities, UD), aces (killing all 5 enemies), clutches (being the last player alive of your team, and killing all enemies, winning the round), MVPs (most valuable player of the round), quad-kills, bullet accuracy (how accurate one's shooting is), first bullet accuracy (how accurate one's first bullet is)). This data can be directly retrieved off the demo without having to use any processing, thus why "compiled".
- Raw data: consisting of a mass of information, not compiled and with lack of documentation. The need to make use of such data, along with the need to provide feedback to the users, made for such data be manually processed and categorized into "wrong-doings" and "well-doings", based on how they affected round outcome. Wrong-doings and well-doings are further explained below.

4.2 Design Ideation

The dashboard has to have an interface that the user can easily navigate around and find all the necessary statistics to create a personal performance overview. To create a final product that better approaches the needs of the user and makes the product more suitable for its intended use [4], we used a user-centered design.

CS:GO is composed of three elements: users, teams and matches. Thus, the dashboard should include information about these elements:

- User data: Data specifically oriented for the user, such as user statistics (kills, deaths, assists, enemies flashed, weapon spray pattern, flashes duration).
- Team data: Data oriented for the team the user has played with.
- Overall match data: General data, providing overall game statistics such as duration, the overall number of kills, and player's statistic boards (that contain user-oriented statistics).

The dashboard will display data gathered, which during the initial evaluation phase, proved to be the most resourceful for self and team improvement. The data for

both the prototype and end production dashboards will be collected from parsed demo files. The end production dashboard will also follow the basic design principles that the users who tested the prototype, felt most comfortable with.

4.3 Design Specifications

This section contains the main functional requirements and non-functional requirements planned for the dashboard. The prototype will be based off these requirements.

The functional requirements are written below as scrum user stories. As the dashboard focuses only on users, the functional requirements only have users as a role.

- As a user, I want to be able to create a team, add members to my team, remove members from my team, so I may view statistics of the players I play with as a group.
- As a user, I want to be able to load match statistics, so I may look at the overall and player statistics of that match.
- As a user, I want to be able to view the overall match statistics, or statistics per round.
- As a user, I want to be able to be able to view the match preview for a selected match, within the dashboard.
- As a user, I want to view the statistics of any player that played in a match.

The above requirements, make the initial set of functional requirements for a player performance dashboard, as these requirements proved to be useful for self-improvement after user evaluation. We consider these as functional requirements as the analysis and display of such data should be the primary goal of the dashboard.

The dashboard also implements a set of design choices that have proven to make browsing and interaction easier [1]:

- Clear and intuitive user interface: users should be able to easily and quickly find their way around the interface, without spending more time than needed to get accustomed to the design. Colors have a great impact [1], especially red and green, since they represent a positive and negative action. Such colors can be used for one to understand that green drives more in-depth into the interface, while red, rolls the user back.
- Clear and direct titles: titles should be self intuitive and explanatory. The user should be able to understand what a button does by reading its title; where not possible, on-screen and easily accessible assistance should be provided in the form of tool tips. Information should be easily accessible, so that any user can find themselves around the dashboard without wasting too much time to get used to the interface.
- Positive feedback on input: in case of errors, the feedback should be helpful and understandable from anyone, to make it possible for the dashboard to be used by anyone, regardless of their technical background.

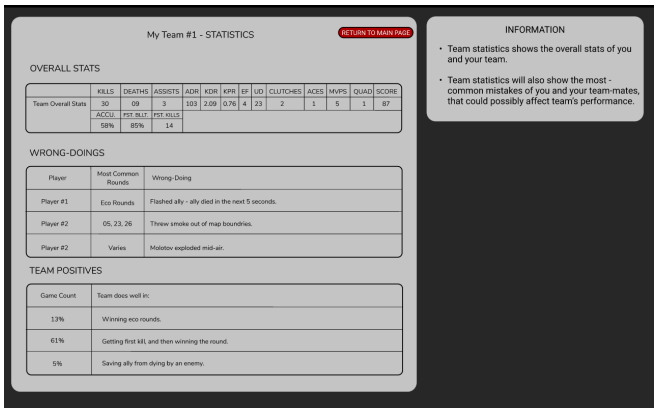
These non-functional requirements are derived from literature review around direct manipulation interfaces[1] [11]. These design choices are recommended, but not mandatory for a dashboard to fulfill its performance role.

4.4 Prototype Realization

Taking the functional and non-functional requirements from the specification section, we created the initial dashboard prototype design using Figma [8].

4.4.1 Team Creation & Statistics

Amongst the dashboard features, included was the possibility for users to create a new team, add or remove members on this team and view overall team statistics. The interface was meant to be direct, without redundant and over extensive information (Pic. 4.4.1.1).



Pic. 4.4.1.1: Team Statistics Prototype Page

Following specification phase, the prototype consisted of two statistic categories, team and players.

Team statistics consisted of:

- The overall matches statistics: Directly compiled data. We wanted to display all the data possible to be retrieved from the demo file to the user, thus why these overall statistics were shown.
- The overall wrong-doings of the team. We analyze the match data and extract statistics that the player may not notice in-game and that can later identify through the dashboard. For team statistics, we group the data by rounds where this behavior is most commonly detected. When grouping by rounds, we identify these four primary round patterns:
  - Eco-round (economy-round): When the team buys light equipment due to lack of money [14].
  - Half-buy: When the team spends only half of the money (usually follows an eco-round [14]).
  - Variation: When the AI cannot find a specific round pattern, it will tag the wrong-doing with ”Varies”.
  - Specific round numbers: When the AI detects a specific round pattern, but not any of the above, then it will say the exact round number (i.e. round 3, 6, 12).

[1]”Direct manipulation is a human–computer interaction style which involves continuous representation of objects of interest and rapid, reversible, and incremental actions and feedback” - Wikipedia.

The wrong-doings of a player are grouped as:

- Flashing self: When the player throws a flash that blinds the player himself.
- Flashing allies: When the player throws a flash that blinds their ally.
- Throwing a smoke or incendiary outside of map bounds.
- Throwing an incendiary on top of smoke: this causes the incendiary to not catch fire and thus, rendered useless.
- Incendiary exploding mid-air: if an incendiary travels for a long time, it explodes mid-air, rendering it useless.

To provide some feedback on what the team does right, we again analyzed the data and determined a set of actions that can be considered as positive achievements. The occurrence probability is also included per achievement, based on the total number of matches played as a team.

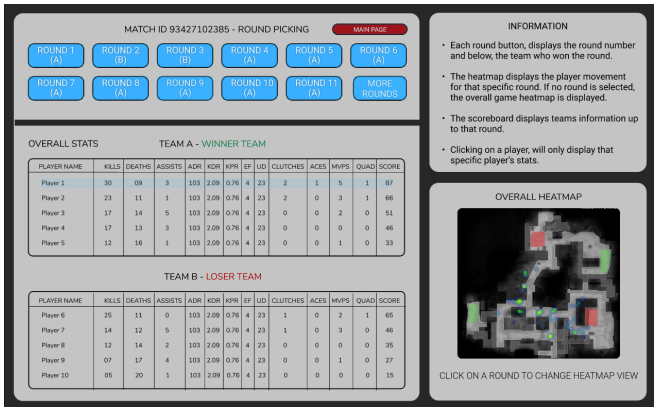
Current positive actions are:

- Saving an ally from dying: when a player saves an ally, who is being targeted by an enemy, from dying.
- Getting the first kill of the round, and winning the round.
- Successfully clutching a round: when a player is the last man standing versus one or more enemies and manages to win the round.
- Winning an eco-round [14] against a full-buy [14] enemy team.

4.4.2 *Match Statistics*

The landing page, first displayed to the user when the dashboard is accessed, will present the latest user matches, in descending order by date, making it easier to access the latest match on record.

When opening a match, the overall statistics and scoreboard will be displayed. Round selection buttons will be positioned on top, a small information box will be displayed on the top right and a 2D match preview will be displayed on the bottom right (Pic. 4.4.2.1). The match preview will display all game events [2] but from a 2D perspective.



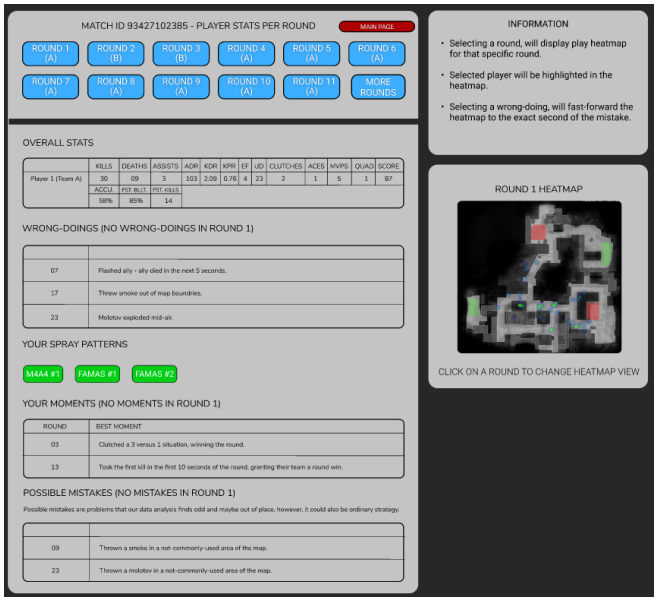
Pic. 4.4.2.1: Match Statistics Prototype Page

The statistics displayed in the scoreboard are general game statistics: kills, deaths, assists, average damage per round (ADR), kill/death ratio (K/D), kills per round (KPR), enemies flashed (EF), utility damage (UD), clutches, aces, MVPs, quad kills and total score.

When a round is selected, the above statistics are displayed **for all the rounds, up to the selected one**, i.e. if a player had 7 kills in the first 5 rounds, then on selecting round 5, the kill count will be 7.

4.4.3 *Player Statistics*

When clicking a user on the scoreboard, the user's statistics for that specific match will open. In the user statistics, we give an overall of the match statistics and based on the analyzed data, wrong-doings, highlights (positive actions), and possible mistakes (Pic. 4.4.3.1).



Pic. 4.4.3.1: Player Statistics Prototype Page

When a player is constantly shooting with their weapon, the action is called spraying (as in, spraying bullets). In CS:GO, there is a specific pattern to follow for each weapon, so the bullets are always directed at the target. Analyzing the data from the demo, we were able to retrieve the direction of the bullets and portray a spray. Utilising some artificial intelligence techniques, we can compare the in-game spray pattern with the game spray pattern and determine accuracy. This data provides improvement feedback to the user, as one can compare their spray pattern with what it should actually be, and as a result, improve shooting accuracy in matches to come.

The mistakes are the same as for team statistic mistakes in section 7.1. However, in this case, the mistakes are round-specific. Each mistake will be indexed by the round it took place in. Clicking on the mistake will fast-forward the match preview to that exact time and position of the selected mistake taking place.

Based on the matches the dashboard will have access to, it collects data on the most commonly used areas (hot spots), based on processed matches. Such data consists of the most common areas for smokes, flashes, or incendiary landings. If the player's smoke, flash, or incendiary is outside of those hot spots, then it will be considered as a possible mistake. However, a throw outside of hot

spots could also be strategic, thus, why it is tagged as a "possible" mistake. Possible mistakes are to be perceived as warnings of possible incorrect behavior.

4.4.4 Prototype Evaluation

We conducted quantitative and qualitative user testing of the dashboard prototype to determine the usefulness of the functional requirement, as well as the quality of the interface (and non-functional requirements as a result).

To test the prototype, we selected a number of people that were playing on an averagely high level, making up the total 14% of CS:GO players, according to TotalCSGO[2].

Participants were given 1 minute to navigate the prototype and learn as much as possible about it. Time was limited to only 1 minute, to simulate a realistic environment. The longer it takes a user to get used to a website, the more chances there are for the user to leave the website [16]. Giving the user only a minute to learn and understand the interface, gave us more feedback to work with for the interface difficulty.

Once the 1-minute time frame ended, participants were requested to follow a set of tasks. We measured the time it took to complete each task, to later identify tasks which took longer than average and adjust the interface accordingly. Results are displayed in the **results** section of this paper.

The tasks asked to be fulfilled were limited to all the features of the prototype:

1. Manage the players of one of the teams.
2. Open the statistics of one of the team.
3. Return to the main page.
4. Open the last match on record, by date.
5. Open the round 1 statistics.
6. Open Player 1 statistics.
7. Open Player 1, Round 1 statistics.
8. Return to the main page (where all matches are shown).

Once the tasks were completed, two questions about the quality of the user interface were asked.

1. How hard did you find the tasks to execute from 1 to 10?
2. How hard did you find the interface to get used to from 1 to 10?

We then explained the content to the participants. As the prototype did not include all the statistics described above, we have to manually explain it to them. After explaining the data and statistics, both displayed and to be displayed, we asked a few questions about the prototype content and participant's CS:GO experience.

1. Do you play in a semi-pro/pro level?
2. How seriously do you play CS:GO on a scale from 1 to 10?
3. What is the highest rank you have achieved in CS:GO?
4. Do you play in a standard team?
5. Do you find the provided statistics useful from a scale of 1 to 10, for self-improvement?
6. If you were to play 10 CS:GO matches, how often would you use the dashboard?
   (a) Every 1-2 (90%-100% percent of the times)
   (b) Every 2-3 games (70%-80% percent of the times)
   (c) Every 4-6 games (40%-60% percent of the times)
   (d) Every 7-9 games (10%-30% percent of the times)
   (e) Never
7. **IF IN A TEAM:** Would you use the dashboard for team improvement (yes-no)
8. **IF IN A TEAM:** Do you find the statistics provided useful for team improvement?

[2]TotalCSGO is a site dedicated to CS:GO and that contains technical and practical information about the game.

5. RESULTS

In this thesis, we aimed to discover the necessary functional and non-functional requirements for the creation of a performance dashboard, that would help players in self-improvement. Through prototyping and testing, using a user-centered approach, we managed to gather a set of results, from which, we derived the functional and non-functional requirements.

5.1 Prototype Testing Results

User testing and post-testing survey had a total of 11 participants. Participants varied in age, being between 20 to 25. All participants were ranked in the top 14% of CS:GO ranks. 73% of the participants played CS:GO on a serious level, 45% was part of a team, and 18% play or had played CS:GO on a semi and professional level.

5.1.1 Task Execution Times

From the prototype testing, we recorded an average total task completion time of 31.68 seconds.

Slower times were initially detected in the execution of the first task (management of the players of the team). This was a result of bad titling of the team's section, which initially, was "Teams". This section was later re-titled to "Team Management" and dropped action fulfillment time by an average of 2 seconds.

Even slower times were recorded for task seven (opening round one statistics for player one). As a result of previously being in round one, participants intuitively believed that opening player statistics in that round would open that player's statistics for round one. This was later fixed and the question changed to opening round two instead.

Task	1	2	3	4
Time	5.23s	3.24s	1.57s	3.64s
Task	5	6	7	8
Time	3.67s	3.62s	6.69s	3.97s

Table 1: Average task completion times in seconds

5.2 Non-functional Requirements

27% of the participants rated their user experience with a 1 out of 10 in terms of difficulty, with 1 being very easy and 10 being very difficult, 54% gave a score of 2 and 19%, a score of 3. The overall user experience and task-performing ability were considered easy by all participants.

The non-functional requirements, as a result of the positive feedback given about the difficulty of the interface, remained mostly unchanged.

On-screen tips were implemented to make it easier for the users to guide themselves through the interface, as a result of the two tasks that took longer than expected to execute.

5.3 Functional Requirements

Asked about the personal player data, statistics displayed, and their usefulness for self-improvement, 54.5% of the participants, of which 18% had played on a semi or professional level, rated with a score of 8 out of 10. 45.5% of the participants gave a score of 9 out of 10. All participants considered the data useful for self-improvement.

45% of the participants, who were part of a team, considered the processed data displayed in the team management section, as useful for training foundation.

Given the overall positive feedback, no changes were made to the functional requirements listed in **section 4.3** of this paper. No additions, possible to be implemented with the data on-hand, were recommended by the participants.

6. CONCLUSION

To conclude, a player performance dashboard does, theoretically, improve player performance. Insight on CS:GO data processing possibilities was gained as a result of extracting data off a replay file. We categorized and processed this data, splitting it into compiled data and raw data. Using Mader's & Eggink's development framework, we idealized, specified, realized and evaluated a player performance dashboard. During ideation, we wrote the basic requirements for a player performance dashboard. Making use of the categorized data and literature review, we laid down the design specifications, functional requirements and non-functional requirements. These requirements were used for the realization of the initial player performance dashboard prototype. We conducted a qualitative and quantitative user testing, along with a survey, to validate the usefulness of functional requirements and the difficulty of the interface. Interface testing participants were given a set of timed tasks to complete and answer a set of questions post-testing. Tasks completed above average time, were reviewed again and changes were made to both the non-functional requirements and interface. After completing the tasks, participants were explained the purpose of the dashboard, all the data processed and displayed. Participants were then asked to evaluate the usefulness of such data for self-improvement. After multiple evaluations, the final functional and non-functional requirements were concluded.

When deriving the functional requirements, we made sure to make use of all the data possible to be extracted from the match replays. The non-functional requirements, or the design specifications, were based on the direct manipulation interfaces principle, where the interface should be direct, simple to use and intuitive.

Main research question and sub-questions were answered as a result. Most of the participants rated the interface as very simple to use and also saw the processed data as useful for self-improvement. All of the participants saw the potential of such a dashboard for self-improvement, given the functional and non-functional requirements described in this paper. This paper results are based on specific population and remain accurate for this specific population's attributes.

7. DISCUSSION

We believe that given time limitations, we were able to achieve significant results. Functional and non-functional requirements were initially drawn as a result of data processing and literature review and were not meant to be final. User testing results saw participants satisfied with both the functional and non-functional requirements, leaving them unchanged.

In comparison to other existing dashboards, that are mostly focused on only providing overall statistics, we implement action feedback for the user to improve on. We had quite some data to initially extract and then process and organize.

7.1 Limitations

Due time restrictions, we were limited to a small population size for user testing. However, a larger population would not make a difference, for as long as the same population characteristics are used.

Dashboard creation was limited to a prototype, whereas functional requirements were limited to manual data processing possibilities. Future work, as described below, could see the creation of an automated system for data processing, as well as a functional dashboard.

7.2 Future Work

Different testing possibilities, practical and close-up interviews, have been left for future work, due to lack of time. Future developments could further expand the data used in this paper, process more data and further increase amount of feedback the user receives.

There are a few ideas that, if time allowed, we would have desired to implement and further collect information about.

Two continuation possibilities would be:

- Creating an actual, working, production dashboard and running a more suitable testing method. Presentation of actual, existing, and applicable data would be more helpful than dummy data that was used in this paper's prototype. Testing could be done via a field experiment: letting participants play Counter-Strike as they would normally do and have them use the production dashboard to learn more about their mistakes after every gaming session. At the end of the experiment, one would have actual practical feedback on whether such a dashboard improves performance or not.

- Contacting and working with a professional team for the creation of a production dashboard, using these same principles and requirements, but extending to fit the team's needs. Working with professionals would guarantee more functional requirements and ideas to extend this paper. Given that most teams have their own data analysts, one would be able to focus more on the creation of the dashboard, instead of the data interpretation.

This paper lays the fundamental functional and non-functional requirements for a performance dashboard, which can then be expanded in many different directions, on researcher's discretion.

8. REFERENCES

[1] Adream Blair-Early, M. Z. *User Interface Design Principles for Interaction Design*. 2008.

[2] Alliedmods.net. *Counter Strike Global Offensive: Game Events; Wiki*. June 2021.

[3] Barbara, S. *Team roles and team performance: Is there 'really' a link?*, vol. 70. The British Psychological Society, 1997.

[4] Chadia Abras, Diane Maloney-Krichmar, J. P. *User-Centered Design*. 2004.

[5] Drenthe, R. *Informal roles within two professional Counter-Strike: Global Offensive eSport teams*. PhD thesis, May 2016.

[6] Driskell, J. E., G.-G. F. S. E. . O. *What makes a good team player? Personality and team effectiveness.*, vol. 10. 2006.

[7] Edwawrds, L. *What is Esports and How Does it Work in Education?*

[8] Figma. May 2021.

[9] Hedstrom, R. *Making Your Team Work: How Coaches Can Transform Groups into Teams*. 2012.

[10] https://csgo-demo manager.com/. *CSGO Demo Manager - tool to parse CS:GO Demo files*. June 2021.

[11] Hutchins, E. L., Hollan, J. D., and Norman, D. A. *Direct Manipulation Interfaces*, vol. 1. Taylor Francis, 1985.

[12] Khromov, N., Korotin, A., Lange, A., Stepanov, A., Burnaev, E., and Somov, A. *Esports Athletes and Players: A Comparative Study*, vol. 18. 2019.

[13] Liquipedia. *forsaken*. MAY.

[14] Lopez, J. *Understanding Economy in CS:GO*. March 2020.

[15] Martin, S. *Scout Reports: How to effectively scout your opposition*. June 2020.

[16] Nielsen, J. *How Long Do Users Stay on Web Pages?* September 2011.

[17] Parker, G. M. *What Makes a Team Effective or Ineffective?* 2008.

[18] ProGameTalk. *15 Top Games Similar To CS:GO*.

[19] Rambusch, J., Jakobsson, P., and Pargman, D. *Exploring E-sports: A case study of game play in Counter-strike*, vol. 4. Digital Games Research Association (DiGRA), 2007. [ed] B. Akira.

[20] SteamPowered. *Steam FPS games listed by popoularity*. May 2021.

[21] Stepanov, A., Lange, A., Khromov, N., Korotin, A., Burnaev, E., and Somov, A. *Sensors and Game Synchronization for Data Analysis in eSports*, vol. 1. 2019.

[22] Thomas, L. *GGPredict is looking to revolutionise CS:GO coaching with AI*. October 2020.

[23] Witkowski, E., Hutchins, B., and Carter, M. *E-Sports on the Rise? Critical Considerations on the Growth and Erosion of Organized Digital Gaming Competitions*. IE '13. 2013.

[24] Wynings, D. *Why Don't All Websites Have an API? And What Can You Do About It?* September 2017.