Gamified Exploratory GUI Testing of Web Applications: a Preliminary Evaluation

Gamified Exploratory GUI Testing of Web Applications: a Preliminary Evaluation

Tommaso Fulcini and Luca Ardito
Politecnico di Torino, Italy
tommaso.fulcini@polito.it
luca.ardito@polito.it

***Abstract*—In the context of Software Engineering, testing is a well-known phase that plays a critical role, as is needed to ensure that the designed and produced code provides the expected results, avoiding faults and crashes. Exploratory GUI testing allows the tester to manually define test cases by directly interacting with the user interface of the finite system. However, testers often loosely perform exploratory GUI testing, as they perceive it as a time-consuming, repetitive and unappealing activity. We defined a gamified framework for GUI testing to address this issue, which we developed and integrated into the Augmented testing tool, Scout. Gamification is perceived as a means to enhance the performance of human testers by stimulating competition and encouraging them to achieve better results in terms of both efficiency and effectiveness. We performed a preliminary evaluation of the gamification layer with a small sample of testers to assess the benefits of the technique compared with the standard version of the same tool. Test sequences defined with the gamified tool achieved higher coverage (i.e., higher efficiency) and a slightly higher percentage of bugs found. The user's opinion was almost unanimously in favor of the gamified version of the tool.**

***Index Terms*—GUI testing, Gamification, Software Testing, Software Engineering**

I. Introduction

Software testing is one of the most critical phases of the software development process. It allows detecting defects in advance in the produced code, avoiding releasing code affected by bugs and security issues, whose repair cost increases once the software is launched on the market.

Testing can be performed at any level and with many techniques (i.e. functional, structural), from low level (unit testing) to high level (system testing). Adopting a particular technique does not exclude another: they can be combined to ensure correctness at different layers.

Testing a system through its Graphical User Interface (i.e., GUI Testing) becomes crucial for software domains such as web applications because of the growing complexity of the provided graphical user experience and the fact that most of the input and output is conveyed through graphical components of the GUI hierarchy. At this level, the system is evaluated by traversing the interface simulating scenarios of final users and using properties of the components as oracles to prove the presence of defects.

Despite being an essential activity, GUI testing is often neglected or superficially performed by testers; the main reasons are two: the first is due to its highly time-consuming nature, primarily when conducted manually by testers, the second is owing to the complexity of configuration and integration of automated testing tools with existing IDEs, and the high cost in maintaining automated test scripts [1].

There are two main techniques to carry out system testing: case-based testing and exploratory testing (ET). The former requires the execution of the test cases to be preceded by a rigorous definition of the goal that each test case should reach and its steps to completion. The latter demands the construction of the test case to the ability of the tester to explore the interface as it is, without following a script. ET, which will be the one we will discuss in this paper, is typically performed by a human who chooses which interactions to perform among all the possibilities offered by the GUI and allows the tester to reach higher flexibility and to get an overall picture of the quality of the system in a faster way [2]. Despite the benefits brought by this technique, testers have to carry out a repetitive job, prone to attention losses that may result in poor test quality and limited effectiveness, i.e. low numbers of defects discovered.

Our vision is to keep the human factor, i.e. the tester, in the loop of creating and executing test cases but increasing commitment, motivation, and performance.

Gamification is a promising technique that consists in applying concepts, elements, and mechanics that are proper to games to a different discipline [3], in this case, Software Engineering. This technique has been proven to be able to boost users' performance and enjoyment when game design principles are correctly applied [4] [5].

As many other studies on gamification applied to the Software Engineering field showed encouraging results, both in the educational and industrial field, we decided to implement a gamification plugin to be installed on a GUI testing tool to make the vanilla test environment more enjoyable to achieve both higher user satisfaction and better performance compared to manual and automated scripted testing.

This paper discusses a preliminary analysis of the game elements we applied to a prototype tool for manual exploratory testing of web applications. Our objective is to verify the benefits of a gamified approach when applied in the test case creation phase and to measure its impact on the efficiency and effectiveness of exploratory testing tools.

The remainder of the paper is organized as follows: Section II defines the concept of gamification and reports other

applications of that practice in the software engineering domain; Section III describes the gamification plugin that we implemented; Section IV reports the preliminary evaluations of our prototype with its experimental settings and the result we reached; Section V deepens the current limitation and the validity threats of the works; finally Section VI concludes the paper and provides a roadmap for future work.

II. Related Works

In the past ten years, the application of gamification has undergone a growing interest [6]: in almost all academic disciplines, gamified approaches have been proposed. In the field of Software Engineering, gamification is a particularly suitable approach since Software Engineering activities are strongly human-centered. Between all the sub-disciplines of Software Engineering for which we can find documented attempts of adoption of gamification techniques, software testing has only received 13% of the total research interest [7]: this figure highlights how the efforts made have been focused mainly on other software engineering fields than testing; with development and requirements having the highest percentage of research items and on a par with project management.

Octalysis, defined by Yu-Kai Chou, is one of the most used frameworks to quantitatively assess how gamification is applied. It identifies eight core drives representing the different aspects of human behavior that gamification can stimulate, divided into four categories: right-brained, the ones related to creativity and social aspects and left-brained, associated with logic and calculations. The framework plots the core drives into an octagon, allowing to visually determine if the elements inserted are well balanced horizontally. It enables the gamification level assessment in any software by measuring the elements of each component, with a prediction of the feeling perceived by the user [8]. In this paper, we adopted the Octalysis framework as an item for assessing the game design of our plugin in order to establish if the chosen game elements were the right ones to fulfill our goals.

The benefits of gamification have been proven to be quite interesting in the software testing context, particularly in the teaching process of the testing concepts.

Some innovative tools were proposed to support the students. For example, Code Defenders [9] allows students to practice mutation testing, also allowing the creation of mutated test suites that can be used in actual use cases. This gamification tool introduces two roles, attacker and defender; the former has to create mutants based on existing code able to pass all the test cases both for the starting test suite and the defender's one. Defenders have to enrich the starting test suite to prevent the attacker's code from passing the test cases. The mutants produced by participants using the tool in the experimental settings described in [10] have also been compared to mutants generated with existing automatic mutants generation tools. The results showed that the mutants manually generated with Code Defenders were able to reach a higher coverage, confirming the benefits of the gamification environment. Our plugin differs from Code Defenders in that the injection of mutants is done at the GUI level by using visual artifacts rather than unit level. Furthermore, we used the mutant injection technique with the purpose of evaluating the tester's performance, while Code Defenders aims to support the creation of test suites able to detect mutants.

Another promising gamified approach to a different testing discipline is the one proposed by Scherr et al. in [11], where the authors describe an innovative approach to acceptance testing of mobile applications based on the use of animojis to determine the feeling of the user who is testing the system. The envisioned approach involves the automation of user evaluation for acceptance testing. The playful element lies in the possibility of showing the user in real-time its corresponding animoji. The animoji makes testers' enjoyment higher, as they can look at themselves in the role of a "cartoon" and makes them aware of the data automatically collected, reassuring them that no sensitive data is taken. This gamification feature leverages people's enjoyment in seeing a moving avatar mimicking their facial expressions.

Ferreira Costa et al. [12] [13], used a gamification framework to teach exploratory testing techniques. Each student received a fictitious name of an existing pirate and took part in a pirate story told during exploratory testing lessons. Different stages were faced, each of which the student learned or applied concepts, as in traditional lessons. After the lessons, an actual test session was carried out. In the end, each student had to fill in the report of the enclosed session. The students then exchanged reports and evaluated each other under the supervision of an expert who then validated all the reports. This validation included rewarding the participants with points (used to buy elements to customize the pirate's profile) and other playful resources to enrich the tester's avatar and medals. According to the domain expert, students were reportedly more engaged and would have done more test sessions and produced good test reports. This work, which is regarding the same testing technique of our paper, uses avatars as a way to certify the progress of the tester similarly as we did in our plugin.

III. The Prototype

Our work is based on a conceptual framework that was presented by Cacciotto et al. [14]. In the present paper, we implement the framework's four elements (final score, progress bar, injected bugs and exploration highlights) and add two additional ones (namely, the avatar and shop system and the achievement badges).

We implemented our prototype as a plugin for Scout, a tool for Augmented exploratory manual GUI testing of web applications proposed by Alegroth et al. [15]. The tool allows the tester to perform system testing for web applications using a capture and replay paradigm. Testers can create a test suite by executing test scenarios against the SUT through the tool's GUI that completely mimics the original application interface. In this way, the original SUT GUI is enriched with superimposed information (e.g., highlighting of clickable buttons, path and coverage information) that supports the tester in the creation of the test scenarios.

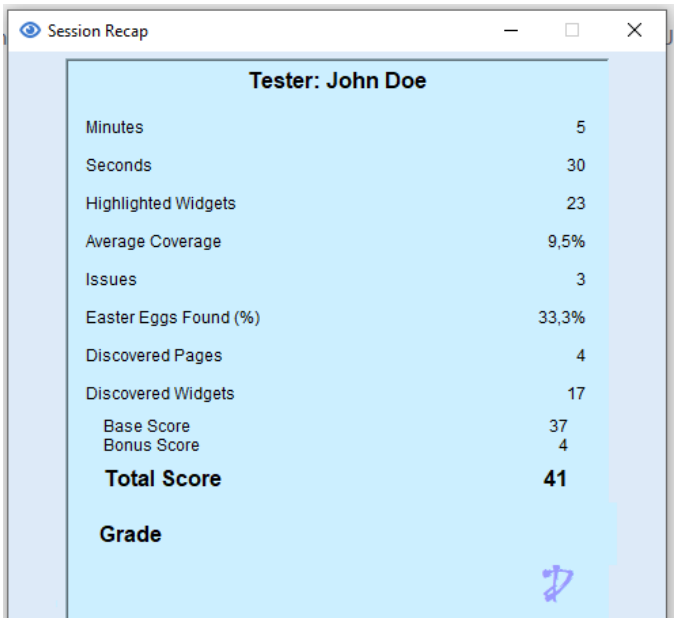

Fig. 1: Results screen, showing the metrics measured for the session and the related score

We represent a testing session with a tree structure. Every node represents a web page and includes data related to the interactions performed by the tester on that particular page. Each session is uniquely identified by the *testerID* and the instant in which it is terminated.

Our prototype implements six different game mechanics:

1) *Final Score and leaderboard*: The main gamification element of the proposed framework is a mechanism that assigns a score to each testing session. A sample prompt showing the final score to the tester is reported in Fig. 1. The score is computed based on the following lower-level metrics:
   - *Coverage component*: Computed as the ratio of interacted widgets over the total widgets of a page, averaged over all visited pages;
   - *Exploration component*: The component depends on the percentage of pages visited and widgets interacted for the first time by the current tester, among all users that have already completed a test session on the same SUT;
   - *Diversity component*: based on the ratio between the total number of interactions and the number of different widgets interacted, it quantifies the diversity of the test case executed by the tester;
   - *Time component*: based on the duration of the test session, it rewards longer sessions in which the tester should explore the GUI more thoroughly;
   - *Problems component*: based on the number of issues reported by the tester during the exploration of the SUT.

   Depending on the score obtained, the tester receives feedback about its performance, in the shape of a grade from D (lowest) to S (highest). A detailed description of how the final score is computed based on the individual components is reported in the conceptual framework by Cacciotto et al. [14].
   The scoring mechanism and leaderboard can be considered as a means to encourage and motivate the user of a system to achieve better performance [16]. Furthermore, these two game elements provide a basis for other more complex game mechanics, besides the fact that the score is also the most widespread playful element used in gamification applied to SE [7].
2) *Progress Bar*: A live visual feedback that shows the number of widgets interacted by the current tester as a green line. The progress bar is rendered over a red one, which represents the total number of widgets on the current page (see fig. 2). Each time a new interaction in the current page occurs, the green portion of the progress bar expands.
   For those pages that at least another tester has previously visited, a blue line is shown between the green and the red to represent the maximum coverage ever reached on that page, i.e. the current high score to beat.
   This element, along with the previous one, is reported in the Octalysis framework under the core drive ”Development & Accomplishment” as a stimulating element that provides satisfaction for the progress made and the reward obtained. By the time we are writing the current paper, as far as we know from the systematic mapping of Porto et al. [7] this is the first time this element has been introduced in a SE discipline.
3) *Injected Bugs*: A visual mutation introduced to emulate real bugs. One widget is selected randomly once per each page, among all the clickable widgets with hyperlinks present on that page. If such a widget is clicked, the destination page will contain the loaded dynamic bug. The bug is represented on-screen as a yellow oval. Injected bugs represent a way to evaluate the tester's ability to find actual bugs in the SUT and the effectiveness of the testing tool. In the literature, the injection of additional, random and dynamic mutants is the only way to compare the performance of a tester in finding incorrect elements in the GUI of the SUT since the

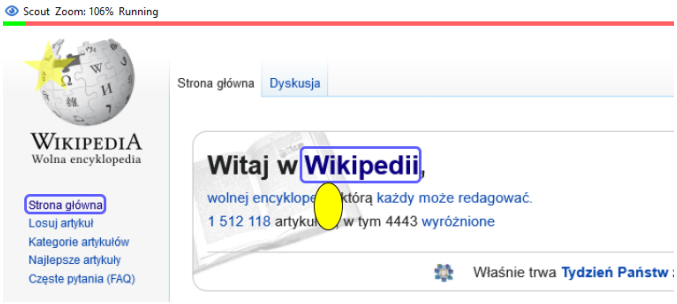

Fig. 2: Graphical feedback of the tool: progress bar, injected bug and *star* to indicate a newly discovered webpage

presence of actual bugs cannot be known apriori. This technique also allows establishing how reactive is the tester in discovering faults by navigating through the hyperlinks. In this research, we mocked the mutant/fault injection by using a visual artifact that lies in a random position of the screen. Since mutation testing is mainly used at the unit testing level, the grounds behind the introduction of this element is the attempt to apply the concepts to system testing as suggested by Zhu et al. [17].

This element can be classified in the Octalysis as an "unpredictability" factor. In fact, the generation of a mutant is similar to the easter egg feature due to their hidden nature and the "hunt" that the tester has performed to find the element. According to Porto et al. [7], this element as well has been applied for the first time in the Software Engineering domain in this prototype.

4) *Exploration Highlights*: A live feedback which highlights a page newly discovered with a semi-transparent star in the top-left corner of the page (see fig. 2). It allows marking a page that has never been tested before, showing the user how deep the current test session's exploration of the SUT provided with respect to all previous ones. This kind of visual feedback has not been previously used in related literature, nor does available frameworks of gamified mechanics mention it. Nonetheless, we consider this element as equivalent to the progress bar in terms of the possible stimulation provided to the users.
5) *Avatar and shop system*: For each tester, a profile window has been created. Here the tester can customize the profile with the preferred avatar by choosing between the available ones. Some avatars are unavailable by default as they are unlockable by spending a predefined amount of virtual currency. The tester has to score in tests sessions or complete the corresponding achievements' goals to collect virtual currency. The avatar and shop systems are shown in fig. 3 respectively on the upper left corner and in the right part of the screen. The avatars are statically generated using the open license and open source Github project Avataaars Generator[1]. The current avatars shopping system has some limitations as, once the tester owns all the available ones, the urge to collect the virtual currency runs out. For this reason, we reported some possible improvements in the future works section.

   Avatars belong to the "Ownership" core drive in the Octalysis and rely on the innately human need to empower their presence and own properties. This need pushes users to care for what they own in the platform and to increase their possessions. This gamification element has been adopted by many studies in the literature and – with some specific categories of people, as in [18] – proved to be an important aid for the user even from a psychological perspective.
6) *Achievement badges*: A sort of "medal" that certifies that the tester has met some particular objectives and requirements. It has the purpose of gratifying the work done by the tester (in a particular session or with its behavior in a set of sessions), rewarding him with an amount of virtual money depending on the difficulty of that particular achievement, and explicit acknowledgements on the results reached. An example of achievements is shown in fig. 3 in the bottom-left part of the screen.

   This element is also one of the most used, right behind the scoring system and leaderboards [7], for this reason, we decided to use it. Achievements belong to the "Development & Accomplishment" category in the Octalysis, a left-brain tendency: the user wants to complete all the achievements to collect all the visual badges assigned. Some ready-to-use open source solutions are available for unit testing [19].

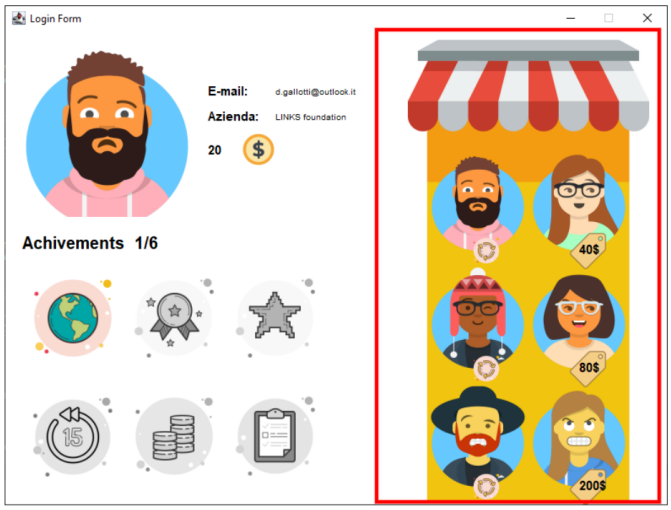

Fig. 3: Profile page of the tester, with the chosen avatar, the achievements' badges and the avatar shop

The tester profile is meant to be public to let other testers see the progress made, see how the profile has been customized, which achievements have been reached, stimulate the competition between them, and push each tester to perform better than the others.

We applied the Octalysis framework to evaluate our prototype tool. Fig. 4 shows the result of the application of the Octalysis graphically. At the same time, the prediction reported a balanced tool between left-brain and right-brain Core Drives, which means to have a good balance between Intrinsic and Extrinsic Motivation. The only reminder was about Extrinsic Motivation factors, which, when poorly designed, could negate the Intrinsic ones.

We continued our work by validating the plugin based on this promising forecast.

[1]https://github.com/fangpenlin/avataaars-generator

Fig. 4: Evaluation of the gamification plugin using the Octalysis framework

IV. Preliminary Evaluation of the Tool

We report the design, goal, research questions, metrics, and procedure adopted for the study following the Goal Question Metric (GQM) paradigm [20], as summarized in table I.

The *goal* of the study was to evaluate the impact of efficiency and effectiveness of the application of gamification concepts on GUI testing of web applications. The results of the study are interpreted according to the perspective of web *software testers* and *researchers*.

A. Research Questions and Metrics

In this section, we detail the Research Questions that we have defined to pursue the goal of our study and the metrics that we adopted to answer them.

RQ1 - Efficiency: What is the impact of gamified mechanics on GUI testing efficiency?

Efficiency, for a software testing tool, technique, or methodology, can be defined as the time required for obtaining a test suite able to provide sufficient coverage of the SUT [21]. In our case, we measure efficiency from another point of view by measuring the following metrics over a fixed time frame:

- *Coverage*: we measure coverage as the percentage of widgets interacted or checked inside each page of the web application, averaged over all the pages explored by the tester;
- *New widgets ratio*: number of new widgets discovered (i.e., never found in previous sessions by any tester) by the current tester, normalized by the total number of widgets encountered during the test session;
- *New pages ratio*: number of pages visited by the tester that was not already *discovered* in previous testing sessions by any tester, normalized by the total number of pages visited in the test session.

TABLE I: GQM Template for the study

Object of Study	Application of gamification concepts
Purpose	Evaluate the benefits for software testing
Focus	Efficiency, Effectiveness, User Experience
Context	GUI testing of web applications
Stakeholders	Software testers, researchers

As this study is conducted in the field of GUI testing, it was not possible to adopt the *statement coverage*. It happened because the Scout tool is agnostic of the technology with which the system is made and performs end-to-end testing in a pure black-box fashion, accessing only the HTML representation of the shown pages. For this reason, we have chosen the widget coverage metric as it was the only available way to quantify the percentage of each screen that was covered.

RQ2 - Effectiveness: What is the impact of gamified mechanics on GUI testing effectiveness?

The *Effectiveness* of a testing technique can be defined as the number of faults that the technique will find [22]. To measure effectiveness in our case, we resorted to measuring the percentage of *Injected Bugs* of those randomly injected in the web application by the tool successfully identified by the testers. We intended a bug as found or identified each time the page with the bug was loaded in the GUI and shown to the tester, meaning that even if the corresponding visual artifact was not rendered, the bug was counted as found.

RQ3 - User Experience: How do testers perceive the applied gamified mechanics?

The last research question deals with the user's perception of the gamified mechanics. We analyzed the participant's answers to a survey about their user experience to answer this question.

B. Experimental Setting

The Gamification Layer's experimental assessment applied to Scout was conducted following the AB/BA design with a sample of ten people, who spontaneously offered to participate in the experiment under receiving a small reward (a meal coupon) when completing the experiment. Participants were both young workers in the software development branch (20% of them) and students from the master's degree in computer engineering and computer science, all in the 20 to 30 age range. Despite their different backgrounds, only 10% of the sample had not had previous Object-oriented programming experience. 70% of them had at least three years of programming experience, mainly with PHP and Javascript.

Participants were divided into two groups: group A had to test the first SUT (a simple company showcase website[2], from now on referred to as *Showcase*) using the vanilla version of the tool, with the gamification plugin disabled. Then, after receiving a brief introduction regarding the applied game mechanism, they had to perform the second session on a second SUT (a popular e-commerce website[3], from now

[2]https://magi-giovanni-funghi-e-tartufi.webnode.it
[3]www.subito.it

TABLE II: Survey Questions

ID	Question (Type)
1.1	Age range (Multiple choice)
1.2	Are you a student or an employee? (Multiple choice)
1.3	Have you ever worked as a professional in object-oriented programming? (Multiple choice)
1.4	How many years of experience do you have in Object-oriented programming? (Open)
1.5	How many years of experience do you have in web application programming? (Open)
1.6	Do you have previous experience in web application testing? (Multiple choice)
1.6b	Which tools did you use for testing web applications? (Open)
2.1	The tool functioning and mechanics were clear to me (Likert)
2.2	How useful did you perceive the progress bar mechanic in order to perform more interaction on the page?(Likert)
2.3	How useful did you perceive the leaderboard mechanic in the sense of competition aroused?(Likert)
2.4	How useful did you perceive the presence of a final session score in order to enhance exploration?(Likert)
2.5	How useful did you perceive the star signalling for discovering a new page? (Likert)
2.6	How useful did you perceive the presence of mocked dynamic bugs in encouraging the search in different pages? (Likert)
2.7	How useful did you perceive the achievements in the paid attention during the session? (Likert)
2.8	How useful did you perceive the presence of unlockable avatars in pushing you to reach a higher score? (Likert)
2.9	Do you believe that the used tool can be easily integrated into an existing testing environment (working or studying)? (Likert)
2.10	Overall, which version do you prefer? (Likert)
2.11	Which of the following elements do you believe are essential in a testing session? (Checkbox)
2.12	What were the principal issues you encountered during the sessions? (Open)
2.13	Have you got any suggestions, comments or unsolved doubts on the presented tool? (Open)

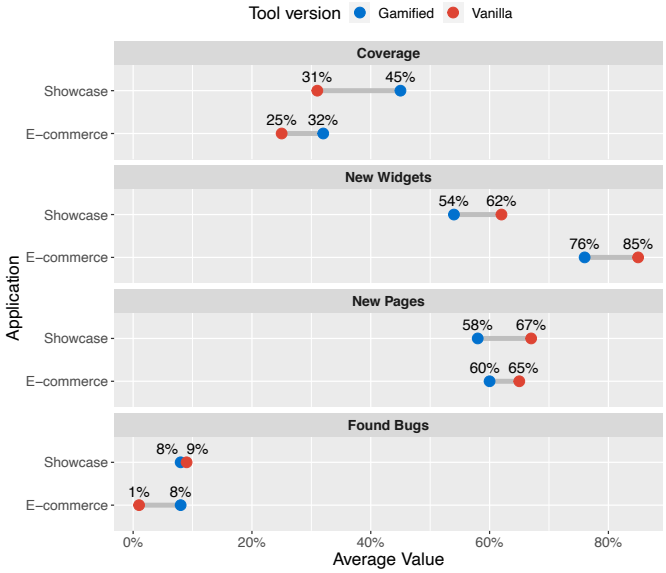

Fig. 5: Results of the preliminary evaluation

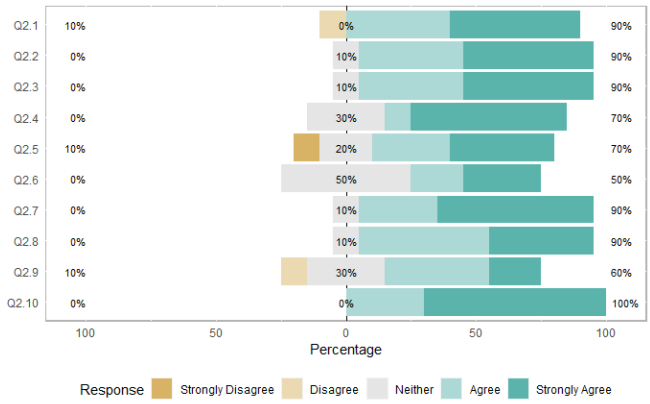

Fig. 6: Answers to the Likert questions of the survey

on referred to as *E-Commerce*), with the gamification plugin enabled. Group B performed the sessions on the SUTs in reverse order.

We have chosen this setting for the experiment because we wanted to avoid introducing any application bias; for this reason, we adopted this A/B scheme. We could not apply the same scheme to the vanilla vs gamified dimension due to the narrowness of the sample; this introduced a validity threat that we will discuss afterward in the corresponding section.

Both sessions had a starting state simulating a previous usage of the tool to let users compare their score with past testers, having a high score to beat in a set of pages. To compare the tool's effectiveness with and without gamification, we counted the amount of identified bugs by the testers also in the vanilla version of the tool, even though the visual artifact were not rendered. With this setup, we could analyze how testers react to the gamified aspects added by the plugin and the influence of the introduced competition with other players.

The test sessions were limited to 30 minutes. At the end of each session, the generated test cases were automatically analyzed to extract the metrics of interest to answer the first two RQs. After the end of the second session, the participants were asked to answer an online survey, to collect demographic information and impressions about the tool used and the gamified aspects they used. The survey structure is reported in Table II.

C. Results

Fig. 5 reports the results of the empirical evaluation, divided by application and version of the tool used (vanilla vs gamified). For each of the four considered metrics, we considered the average over the set of experimental sessions that the respondents performed.

Regarding *RQ1*, we observe an increasing trend in the average coverage when transitioning from the vanilla to the gamified version of the tool, with an average increase of 7% for the *E-Commerce* experimental object and 14% for *Showcase*. These results likely reflect a positive impact of the addition of graphical feedback for the coverage during the test session execution, which effectively encouraged the testers to perform more operations on the widgets during the test sessions.

This result is countered by the decrease in the percentage of new widgets (-9% for the *E-Commerce* SUT, and -8% for *Showcase* SUT) and new pages (-5% for the *E-Commerce* SUT, and -9% for *Showcase* SUT) over all those traversed during a test session. These results suggest that testers have focused their attention on exploring already visited pages, preferring to reach a higher coverage in a lower number of pages rather than exploring more pages more superficially.

Regarding *RQ2*, we observe a slight decrease of the percentage of bugs found over the total injected for the *E-Commerce* SUT, and a steady increase in the metric for the *Showcase* SUT. The small experimental sample size can justify such divergence in the two measures.

Regarding *RQ3*, we report the average answers to the questions of the survey in fig. 6. We observe that all the averages are positive towards the usage of gamified mechanics in the practice of exploratory GUI testing, compared to the vanilla version of the tool. Specifically, question 2.10, which used a Likert score from 1 to 5, where 1 was associated with the *Vanilla version* and 5 with the *Gamifed version*, obtained a 4.8 average score, with all the preferences to the tool with gamification plugin.

The lowest average Likert scores were obtained by the three survey questions: one regarding the integration of the tool in the existing environment, the other two regarding the usefulness and intuitiveness of the star signaling new pages, and of the presence of mocked dynamic bugs (respectively 3.7, 3.9 and 3.8 average scores). These results about gamification elements are in line with what was observed for RQ1 and RQ2, for which no benefits in exploration and bug finding effectiveness were measured. The low result about the integration of the tool shows how users had difficulty in making a context switch from their current testing tool to Scout: this means that additional effort is required in order to broaden the adoption of Scout, regardless of the version used (vanilla or gamified).

We report some constructive notes regarding the question about encountered issues. Two participants did not see the star mark for the newly discovered pages, having seen pages solely already discovered. Some other issues were about Scout and not the Gamification Plugin; these were primarily difficulties in performing some interactions and suggestions to improve the tool per se. The answers to the final open question of the survey also hinted at the addition of more explicit mechanics to visualize and track the discovery of new widgets and pages in addition to a more appealing profile window, enriched with more statistics about the performances of the tester in the various tested project.

V. Threats to validity

Threats to Construct Validity.

In the adopted experimental setting, even if the two considered SUTs belong to the same domain, the comparability of the results obtained for them is limited since the set of pages and widgets present in the web applications, as well as the nature of the interactions that can occur, is substantially different.

Additionally, the application of the gamification layer only in the second test session could impact the results. Using the same tool in both sessions may introduce a *learning effect* in the testers, with the possibility to achieve better results independently of the presence of gamified mechanics. We still believe that it is reasonable to study the effect of gamification when applied to a known environment. In a real-world scenario, the adoption of the proposed plugin would be an addition to the classic environment of the tester, and it is implausible that testers start their activity directly with the gamified tool.

To better understand the influence of gamified components, further evaluations shall adopt a full-factorial AB/BA pattern, alternating in the first session the usage of the gamified tool in group A to one without gamification for group B [23].

The way the bug injection is currently implemented provides only a statistic that considers the pages that contain a bug, counting it each time as a found bug, without relying on the ability of the tester to recognize different categories of bugs. We believe that the provided mock-ups still simulate the presence of real bugs, putting the exploration made by a tester to the test.

Threats to External Validity.

The further problem is that no support is provided for the version of Scout used for the execution of JavaScript code snippets inside the testing environment. For such reason, the interactions that can occur have been limited to cut off all the functionalities that would have led to misbehaviors, like the appearance of pop-ups, hyperlinks forcing the scroll of a page and the execution of scripts inside the browser. This assumption narrows the set of testable SUTs with the tool and therefore limit the generalisability of the results. Additional limitations to generalisability are caused by the way the interaction trees are constructed by the tool, which so far cannot consider the operations on widgets with dynamically generated IDs and properties.

VI. Conclusion and Future Work

In this paper, we described the incorporation of gamification mechanics in the practice of exploratory GUI testing. By examining a set of four efficiency and effectiveness-related metrics on ten test sessions and comparing the results obtained with and without the gamification, we observed positive effects of the applied concepts on the coverage reached by the testers on the visited pages. On the contrary, testers tended to visit fewer pages in their test sequences when using the gamified version of the tool, hinting at a reduced tendency to explore in-depth the web application instead of increasing the coverage for already visited pages.

The user experience of the testers with the gamified version of the tool was instead unanimously positive. These results suggest that a more refined calibration of the provided visual feedback and gaming components provided can provide beneficial results for both coverage and bug-finding capabilities of exploratory GUI testing, stimulating a deeper exploration

and a more comprehensive selection of test inputs for the web applications under test.

We noted that the most impactful element was the progress bar, as each participant considered it an essential mechanic, the effect being a trend to achieve higher coverage despite deep exploration of the test domain. According to the questionnaire, achievements were also an appreciated element, but no apparent effect on the produced test case was seen; only a higher motivation was stimulated in the sample: many achievements required multiple testing sessions to be accomplished. The lower-rated element was the exploration highlight, with some reported cases clearly stating that they had never seen the star visually present in the left upper corner.

Lastly, the work described in the present manuscript is intended as a pilot study for a large-scale empirical experiment with graduate students. In this future experiment, we also plan to extend the analysis of the impacts of gamification on GUI testing by considering the effectiveness in finding actual mutants and the quality of test cases in terms of similarity to real usage scenarios of the SUT.

We also foresee incorporating and validating additional game mechanics, such as challenges and quests, to exploratory GUI testing. We also plan to refine how the current game elements are applied, expanding the avatar customization, starting with buying small items or expressions to add to the basic avatars to the complete customization of the avatars or with the release of exclusive avatars available just for a limited amount of time. The current bug injection implementation in the future should be enhanced to provide a visual mutation compliant with the classification of Alegroth et al. [24] Also, the coverage computation could be adapted to a different definition, avoiding the count of the many text area and limiting it to the elements the test can interact with; this implies the possibility of a redesign of the scoring system. Whether the future results will be encouraging, the plugin could be transformed into a stand-alone testing tool or a browser plugin to verify its effectiveness.

REFERENCES

[1] M. Leotta, D. Clerissi, F. Ricca, and P. Tonella, "Chapter five - approaches and tools for automated end-to-end web testing," ser. Advances in Computers, A. Memon, Ed. Elsevier, 2016, vol. 101, pp. 193–237.

[2] J. Itkonen and K. Rautiainen, "Exploratory testing: a multiple case study," in *2005 International Symposium on Empirical Software Engineering, 2005.*, 2005, pp. 10 pp.–.

[3] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining gamification," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011*, vol. 11, 09 2011, pp. 9–15.

[4] G. Fraser, A. Gambi, M. Kreis, and J. M. Rojas, "Gamifying a software testing course with code defenders," in *Proc. of the ACM Technical Symposium on Computer Science Education (SIGCSE)*, ser. SIGCSE'19. ACM, 2019, to appear.

[5] K. Berkling and C. Thomas, "Gamification of a software engineering course and a detailed analysis of the factors that lead to it's failure," in *2013 International Conference on Interactive Collaborative Learning (ICL)*, 2013, pp. 525–530.

[6] M. Trinidad, M. Ruiz, and A. Calderón, "A bibliometric analysis of gamification research," *IEEE Access*, vol. 9, pp. 46 505–46 544, 2021.

[7] D. de Paula Porto, G. M. de Jesus, F. C. Ferrari, and S. C. P. F. Fabbri, "Initiatives and challenges of using gamification in software engineering: A systematic mapping," *Journal of Systems and Software*, vol. 173, p. 110870, 2021.

[8] Y. Chou, *Actionable Gamification: Beyond Points, Badges, and Leaderboards*. Createspace Independent Publishing Platform, 2015. [Online]. Available: https://books.google.it/books?id=jFWQrgEACAAJ

[9] J. M. Rojas and G. Fraser, "Code defenders: A mutation testing game," in *Proc. of The 11th International Workshop on Mutation Analysis*. IEEE, 2016, pp. 162–167.

[10] B. C. José Miguel Rojas, Thomas White and G. Fraser, "Code Defenders: Crowdsourcing effective tests and subtle mutants with a mutation testing game," in *Proc. of the International Conference on Software Engineering (ICSE) 2017*. IEEE, 2017, pp. 677–688.

[11] S. A. Scherr, F. Elberzhager, and K. Holl, "Acceptance testing of mobile applications: Automated emotion tracking for large user groups," in *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*, ser. MOBILESoft '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 247–251. [Online]. Available: https://doi.org/10.1145/3197231.3197259

[12] I. E. F. Costa and S. R. B. Oliveira, "A systematic strategy to teaching of exploratory testing using gamification," in *ENASE*, 2019.

[13] I. F. Costa and S. Oliveira, "The use of gamification to support the teaching-learning of software exploratory testing: an experience report based on the application of a framework," in *2020 IEEE Frontiers in Education Conference (FIE)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2020, pp. 1–9.

[14] F. Cacciotto, T. Fulcini, R. Coppola, and L. Ardito, "A metric framework for the gamification of web and mobile gui testing," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2021, pp. 126–129.

[15] M. Nass, E. Alégroth, and R. Feldt, "On the industrial applicability of augmented testing: An empirical study," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2020, pp. 364–371.

[16] E. D. Mekler, F. Brühlmann, K. Opwis, and A. N. Tuch, "Do points, levels and leaderboards harm intrinsic motivation? an empirical analysis of common gamification elements," in *Proceedings of the First International Conference on Gameful Design, Research, and Applications*, ser. Gamification '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 66–73. [Online]. Available: https://doi.org/10.1145/2583008.2583017

[17] Q. Zhu, A. Panichella, and A. Zaidman, "A systematic literature review of how mutation testing supports quality assurance processes," *Software Testing Verification and Reliability*, vol. To Appear, 05 2018.

[18] A. Mendoza-González, H. Luna-García, R. Mendoza-González, C. Rusu, H. Gamboa-Rosales, J. I. Galván-Tejada, J. G. Arceo-Olague, J. M. Celaya-Padilla, and R. Solis-Robles, "An approach to make software testing for users with down syndrome a little more pleasant," in *Proceedings of the XIX International Conference on Human Computer Interaction*, ser. Interacción 2018. New York, NY, USA: Association for Computing Machinery, 2018.

[19] B. Beck, "nose-achievements," https://github.com/exogen/nose-achievements, 2010-2016.

[20] V. R. Basili, "Goal question metric paradigm," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[21] S. Eldh, H. Hansson, S. Punnekkat, A. Pettersson, and D. Sundmark, "A framework for comparing efficiency, effectiveness and applicability of software testing techniques," in *Testing: Academic & Industrial Conference-Practice And Research Techniques (TAIC PART'06)*. IEEE, 2006, pp. 159–170.

[22] E. J. Weyuker, "Can we measure software testing effectiveness?" in *[1993] Proceedings First International Software Metrics Symposium*. IEEE, 1993, pp. 100–107.

[23] L. Madeyski and B. Kitchenham, "Effect sizes and their variance for ab/ba crossover design studies," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 420.

[24] E. Alégroth, Z. Gao, R. Oliveira, and A. Memon, "Conceptualization and evaluation of component-based testing unified with visual gui testing: an empirical study," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2015, pp. 1–10.