On Effectiveness and efficiency of Gamified Exploratory GUI Testing

On Effectiveness and Efficiency of Gamified Exploratory GUI Testing

Riccardo Coppola, Tommaso Fulcini, Luca Ardito, Marco Torchiano, and Emil Alègroth

Abstract—*Context*: Gamification appears to improve enjoyment and quality of execution of software engineering activities, including software testing. Though commonly employed in industry, manual exploratory testing of web application GUIs was proven to be mundane and expensive. Gamification applied to that kind of testing activity has the potential to overcome its limitations, though no empirical research has explored this area yet.
Goal: Collect preliminary insights on how gamification, when performed by novice testers, affects the effectiveness, efficiency, test case realism, and user experience in exploratory testing of web applications.
Method: Common gamification features augment an existing exploratory testing tool: Final Score with Leaderboard, Injected Bugs, Progress Bar, and Exploration Highlights. The original tool and the gamified version are then compared in an experiment involving 144 participants. User experience is elicited using the Technology Acceptance Model (TAM) questionnaire instrument.
Results: Statistical analysis identified several significant differences for metrics that represent the effectiveness and efficiency of tests showing an improvement in coverage when they were developed with gamification. Additionally, user experience is improved with gamification.
Conclusions: Gamification of exploratory testing has a tangible effect on how testers create test cases for web applications. While the results are mixed, the effects are most beneficial and interesting and warrant more research in the future. Further research shall be aimed at confirming the presented results in the context of state-of-the-art testing tools and real-world development environments.

Index Terms—Software Testing, Web Application Testing, Gamification

1 Introduction

Software testing is a critical activity in the software development process. Its main purpose is to reveal faults and defects, but it is also utilised to ensure reliability and conformance to functional requirements of software artefacts. Several techniques have been proposed for software testing, ranging from low-level unit testing of individual software components to higher-level exploratory testing through the software's Graphical User Interface (GUI).

In recent years, many tools and techniques have been proposed by research and industry to automate test generation in modern software development. However, while automation generally achieves high coverage in code-level white-box testing activities, automated test cases do not always generate realistic GUI-based sequences of interactions. By contrast, manual exploratory testing has survived as a costly, error-prone and tedious yet crucial activity. However, the required manual work does govern the amount of testing an organisation can support due to financial factors or human capital. The available support thereby affects the amount of testing and, in turn, the software's quality [1].

Gamification is defined as *"the use of game design elements in non-game contexts"* [2]; it is typically employed to increase the engagement, motivation, and performance of participants in solving a task to which game mechanics are applied [3]. Many works in the field have motivated, conceptualised, and evaluated gamification mechanics applied to different activities in Software Engineering. Such activities range from mutation testing to the management of the application development lifecycle [4]. Although gamification has primarily been proposed in the educational field, encouraging results have also been provided by industrial case studies [5].

Software testing is well suited for gamification since the activity revolves around finding defects/faults, completing test objectives or increasing test capabilities. Hence, activities can be quantified, measured and compared among testers/users in a game-like way. Gamification is seen as an opportunity to enable, and encourage, crowdsourced testing tasks. Increasing the number of concurrent testers will increase the number of test cases defined and executed on software artifacts [6]. As reported in the mapping study by de Paula Porto et al. [7], benefits of gamification in Software Testing include the incentive to log defects, a better motivation to perform activities, a higher opportunity of helpful user feedback, and a general improvement of the quality of the work performed.

This paper builds upon a previous conceptualisation of a framework for metrics and components for manual exploratory GUI testing of web applications and leverages a prototype tool implementing such conceptualisation [8].

The framework incorporates mechanics such as session scores, leaderboards, and live graphical feedback. To the best of our knowledge, it is the first attempt at applying these mechanics to the practice of tool-supported ex-

- *R. Coppola, T. Fulcini, L. Ardito and M. Torchiano are with the Department of Control and Computer Engineering, Polytechnic University of Turin, Italy. E-mail: first.last@polito.it*
- *Emil Alégroth is with the Blekinge Institute of Technology, Sweden. E-mail: emil.alegroth@bth.se.*

Manuscript received January 19, 2022.

ploratory GUI testing for web applications.

The main goal of the present paper is to report the findings of a large-scale experiment with graduate students to evaluate the impact of gamification mechanics on the effectiveness, efficiency, quality of generated test cases, and users' experience. The experiment is performed by using a prototypical implementation of gamified mechanics in a tool for GUI-based Capture & Replay generation of test sequences for web application, namely Scout. We deemed the graduate students representative of newly hired workers in the industry with limited experience, who are frequently employed in manual testing activities of finite products, while professionals with sounder domain knowledge are more indicated to be hired to work with systematic test case design methods [9]. A measure of the user experience provided by the framework can also be of interest since gamification can be seen also as a means to facilitate onboarding. Based on the results, we elaborate on the potential benefits and weaknesses of a gamified approach and propose extensions and aspects worth investigating in future research.

The remainder of the manuscript is structured as follows: Section II discusses the background of the practice of GUI testing of web applications and related works about gamification in the Software Engineering discipline; Section III presents our framework and the gamified mechanics it implements; Section IV describes the developed tooling, experimental setting, goals, hypotheses, threats and results; Section V reports and discusses the experimental findings; Section VI provides a higher-level discussion of the benefits and drawbacks that can be carried by using the gamified mechanics, and frames the results of the study in the context of the current technical limitations of the used tool. Section VII concludes the paper by providing pointers for additional evaluations and extensions of the current work.

2 Background and related work

2.1 Exploratory testing

Exploratory Testing (ET) is a form of manual testing widely used in practice – and popular among agile development teams – due to its recognised flexibility and defect finding ability [10]. In practice, ET consists of a process of continuous test design and execution, i.e. tests are created against the finished SUT, and test design is performed as an integral part of test execution instead of being conducted in a preliminary design phase. ET leverages the tester's domain and technical experience to uncover new, previously unknown faults in the software [11].

Although manual note-taking is perhaps the most common way of recording ET, tool support with Capture & Replay (C&R) features is also used to generate test scripts from the sequences of the testers' interaction with the System Under Test (SUT). These test scripts can later be re-executed by a dedicated test engine for regression testing purposes. The syntax of the generated scripts is most often outside the user's control, resulting in an additional technical knowledge learning burden for the tester. Other ET tools record the interactions as logs for manual analysis or re-execution.

Examples of C&R tools are WebRR, which uses DOM layout-based locators to identify elements to be interacted [12], or the general-purpose SikuliX tool, which uses screen captures of the GUI to drive the test execution [13].

One of the most recent advances in C&R testing is *Augmented* testing, which facilitates the act of gathering inputs against the widgets by dynamically superimposing textual and visual information over the SUT's GUI. Augmented testing has been defined by Nass et al. [14] for the Scout tool, which applies augmented testing to manual exploratory testing of web applications. During the capture phase, the tool records the tester's interactions in a test data model that is continuously analysed to extract new test data or scenarios that will be shown to the tester in the augmented GUI. The augmented GUI provides hints, textual test data, and coverage information. Recorded test cases can be replayed manually or automatically from the test recordings. The tool also allows to add assertions (e.g., the presence or content of an on-screen widget) during the definition of test sessions, which are later automatically verified when the tests are re-executed. The tool also allows to signal *issues* during the manual exploration phase, thus giving the possibility to the tester to signal rendering and content errors in the SUT. The tool provides an API to extend features through plug-ins.

2.2 Gamification

Gamification is currently a promising technological trend involving gaming mechanics in non-game environments, systems, and tasks. Many research tools have conceptualised gamification in related literature. Among these, one of the most comprehensive and adopted is the *Octalysis* framework, defined by Yu-Kai Chou. The framework identifies eight core drives that represent aspects of human behaviour that can be stimulated through gamification. Examples of such drivers are *development and accomplishment*, *social influence*, *empowerment of creativity*, *unpredictability and curiosity*. By measuring the elements of each driver, the framework enables assessment of the level of gamification in any software [15].

Several sources in the literature have presented gamified constructs and applied these to software testing. A systematic mapping study by Pedreira et al. identified gamification as an emerging trend in the last decade in all areas of Software Engineering. The approach has been applied primarily to project planning and system implementation activities. The most frequent game mechanics applied to software engineering were the adoption of point and score systems and user badges [2]. De Jesus et al. in [4] characterised the application of gamification to software testing. They found that the test case design, evaluation, and maintenance are the most interesting software testing practices from a gamification standpoint. An important finding of this mapping study for this paper regards the testing *level* at which gamification has been applied. While lower-level testing has been subject to gamification research, no documented effort was identified at a system level.

Furthermore, several tools have been developed for teaching software testing by using gaming aspects. An example is CodeDefenders, proposed by Fraser et al. [16]. The tool is designed to teach mutation testing through competition among two teams of testers: attackers, inject bugs into existing code, while defenders, who have to strengthen

the existing test suite, try to predict new bugs introduced by the attackers. Code Defenders showed increased user involvement in the testing activity and increased robustness of generated test suites in an academic course evaluation.

Elbaum et al. [17] introduced Bug Hunt, a web-based platform to introduce students to the concept of black-box and white-box unit testing. The tool incorporates game mechanics such as the presence of levels where the testers have to advance and the confrontation of the current students' scores against the average score of the classroom. The application of gamified mechanics was deemed beneficial since most involved students found the gamified tasks more engaging and interesting than traditional ones.

Parizi [18] included game concepts in traceability tasks with the tool GamiTracify, for software testing and maintenance; the game mechanics included points, achievements, reputation building, time progress and the presence of quests and challenges. The study included an evaluation that demonstrated that a higher precision was obtainable with the gamified tool compared to a non-gamified one. Laurent et al. [19] proposed a crowdsourcing platform for mutation labelling and detection, which applies gamification in the form of progressive presentation of mutants according to the testers' skill level, points, and a live leaderboard. The scoring gamification mechanism aims at incentivising task completion by assigning points every time enough other players agree on the labelling of a mutant.

Few works in the literature have presented evaluations of gamified mechanics in industrial contexts. Liechti et al. [5] presented a case study about continuous improvement, automated testing and feedback mechanisms for developers. The platform integrated an element of gamification in live graphical feedback of users' performance in the assigned task. The authors measured positive impacts, principally in terms of the enhancements in the testers' engagement. The authors also presented qualitative results hinting that the gamification features incentivised the involved participants to write more tests and increased their engagement.

In recent years, the application of the crowdsourcing technique to testing, the so-called *crowdtesting*, has become a trend [20]. This technique makes it possible to intensively test an application using a very large number of users who, by breaking down a large problem into microtasks, complete in a short time, what would take a single tester a great deal of effort and time. It is particularly suited for big problems that are difficult to automatise, as they require a human factor to be solved [21].

Exploratory GUI testing is an example of a fruitful field of application of *crowdtesting*: testers can be recruited using platforms such as Amazon Mechanical Turk[1] where they are asked to perform small tasks and to produce reports with the outcome of the assigned task, typically under a monetary remuneration [22].

Although *crowdtesting* could represent a viable way to improve test quality, leveraging only extrinsic motivators (those that are typical of crowdsourcing), it remains a very expensive approach that only keeps the user effectively involved for a short time due to over-justification effect, whereby the effect of an extrinsic motivator diminishes as the user becomes accustomed to it. Gamification conversely focuses on creating a playful environment for the user to improve engagement by providing both extrinsic and intrinsic motivators. The final objective of this study is to focus on gamification without taking into account crowdtesting mechanics.

1. https://www.mturk.com/

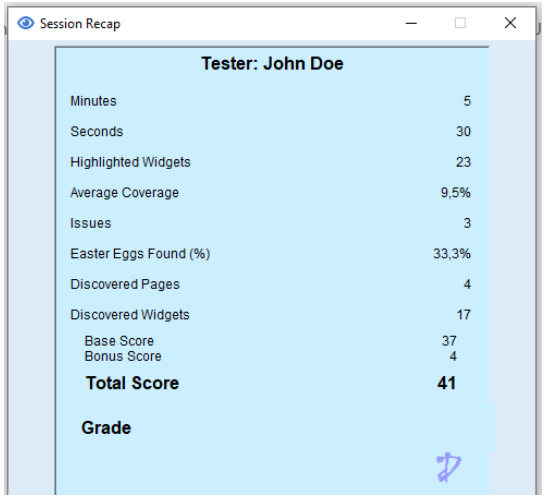

Fig. 1: Results screen, showing the metrics measured for the session and the related score

3 The Framework

This section describes our conceptual framework for the gamification of activities related to exploratory GUI testing of web applications. The framework has been originally introduced in our previous work [8].

Our conceptual framework has been implemented as a plug-in for Scout, described in the Background section of the present manuscript. The tool has been considered apt for the addition of gamification elements because of the presence of the augmented layer superimposed on the SUT, which could be leveraged to convey gamified mechanics.

In our framework, we model a testing session as a forest structure made of trees, each one representing a test case. Every node of the tree represents a web page, encapsulating data related to the interactions (i.e., clicks on interface elements, selection of menu items, or insertion of text in text boxes) performed by the tester on that particular page. Edges between nodes are interactions that trigger page changes in the SUT, whereas interactions that do not result in a page change are represented as arrows back to the same node.

The complete testing session, made of different test cases, can be reconstructed by running through all the paths of the different trees. Each session is uniquely identified by a *testerID* and timestamp based on when it was terminated.

Our framework includes four 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. We adopted the scoring and leaderboard mechanics since we aimed at incorporating the competitiveness dynamic in the gamified tool and, according to the

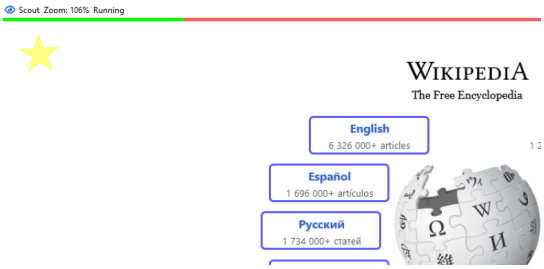

Fig. 2: Graphical feedback of the tool: progress bar, interactable widgets, and *star* to indicate a newly discovered webpage

characterisation study by de Jesus et al. [4], they are the most widely used in gamified software testing. A sample prompt of what this looks like to the user is presented in Figure 1. The score is computed as a function of the following metrics:

- *Coverage*: Computed as the ratio of number of widgets the user has interacted with over the total number of widgets on a page, averaged over all visited pages. The coverage is normalized in the range [0,1] for each visited page, in order to allow comparison between among different pages. Whenever a tester visits a new page within their session, the coverage corresponding to the said page is included in the average coverage computation;
- *Exploration*: The percentage of new pages and widgets a user has interacted with for the first time. New interactions are compared to known interactions performed by all users. Currently, the prototype only works offline, i.e. the comparison works only if two users share the same machine to test the same SUT. We argue that in order to keep this metric significant, the exploration should refer to the specific version of the SUT being tested, resetting the widget discovery at each release or visual modification of the SUT.
- *Diversity*: Computed as the ratio between the total number of interactions and the number of unique widgets the user interacted with. Thus quantifying the diversity of the test cases executed by the tester;
- *Time*: The duration of the test session, where longer active sessions are rewarded. To prevent users from exploiting the scoring system, we use a multi-level multiplier based on the ratio of actions per minute. Longer sessions are rewarded as we assumed that they allow users to test the GUI more thoroughly;
- *Problems*: The number of issues and bugs signalled by the tester during the exploration of the SUT. *Tracing this metric allows us to evaluate the effectiveness of the test session in finding defects in the SUT.*

At the end of the exploratory testing session, the tester receives a score which is computed as a function of all the five components described above. Such score is normalised between 0 and 100, and based on it the tester is then given a rank as a five-value grade from D (lowest) to S (highest), following popular conventions adopted by games [23]. The interested reader can find the detailed formula for the computation of the final score in Appendix A.

2) *Injected Bugs*: the framework includes a module to generate random visual mutations in the SUT. The injection of random mutants comes with visual feedback and a score shown when the user correctly identifies the mutation on the page. This is considered a gamification element belonging to the *Accomplishment* and *Unpredictability* dimensions of the Octalysis framework. One widget is selected randomly for each page, among all the clickable widgets with hyperlinks present on that page. A visual mutation is applied to such a widget. The mutation can be a *Delete* mutation (i.e., the widget is removed), a *Change* mutation (i.e., the content or appearance of the widget is changed), or an *Add* mutation (i.e., the widget is duplicated). The categories of widgets are taken from existing works in the literature about mutation testing [24].
Injected bugs were adapted from the practice of mutation testing and from existing literature about gamified software testing. A popular example of a gamified tool involving mutant injection is Code Defenders [16]. Our tool, which is not specifically designed for education about mutation testing, does not consider mutants as a central game element, but instead a secondary component to encourage more diverse and longer GUI explorations. Injected mutants, in fact, serve both as a form of competition between the users and evidence of progression. Finding actual bugs through the pages explored encourages page exploration by providing immediate feedback to the testers.
Injected bugs represent a way to evaluate the tester's ability and the tool's effectiveness. Since the presence of actual bugs cannot be guaranteed, the injection of additional, random and dynamic mutants is required to compare the performance of testers. Placement of known bugs also helps control for metrics of interest, e.g. user's inclination to explore the SUT.

3) *Progress Bar*: Feedback mechanics are also used to increase the user's engagement with the tool and understanding and self-evaluation of their activities. Real-time visual feedback shows the user the number of widgets they have interacted with as a green line. The progress bar is rendered over a red one, representing the total number of widgets on the current page. The visual feedback for coverage is complemented by a mechanism already implemented by the *Vanilla* tool. When hovered, the widgets that can be interacted with are contoured by a semi-transparent grey box. This function is also used to render already interacted widgets with a blue contour (see Figure 2). Each time a new interaction in the current page occurs, the green

portion of the progress bar expands.
Additionally, for pages visited by at least one other tester, the progress bar shows a blue line between the green and the red area that represents the maximum coverage ever reached on that page, i.e. equivalent to the current high score to beat. This visual element is designed to introduce competition in the tool. Similarly to the exploration metric, this element currently works offline when different testers share the same device. The progress bar element has no equivalent in the existing literature and can be seen as a visual rendition of the high-score gaming mechanic.

4) *Exploration Highlights*: A real-time feedback mechanism highlights when a page is discovered for the first time with a semi-transparent star in the top-left corner of the page (see Figure 2). This element is inherently linked to the exploration component of the score, being related to the pages that were visited for the first time in a session as opposed to all past sessions performed by all the users. To avoid the exhaustion of this mechanic, it is necessary to reset the relative values release by release. We use exploration highlights to incentivise the testers to explore more, and diverse widgets and pages. Exploration highlights can be considered a form of achievement shown to the user, a well-established mechanic in gamified tools. This element shares with the exploration metric and the progress bar the precondition that the test sequences must be generated on the same machine to appreciate the highlighting.

The selection of the game mechanics was also partially influenced by the nature of the tool that was extended, and the type of evaluation that we intended to perform. The Scout augmented tool was particularly apt for the implementation of the injected bugs mechanics (since it is based on Selenium, which allows the live modification of web pages during exploration) and the visualization of graphical feedback. We focused on simpler gamification mechanics to evaluate, since more complex elements (e.g., narratives, profile management, quests) would have required separate longitudinal experiments to be properly evaluated.

4 Experiment Design

We describe the experimental study design according to the guidelines for reporting software engineering experiments defined by Jedlitschka et al. [25]. We validated our experiment design description by using the checklists in the Empirical Standards for Software Engineering Research by Ralph et al. [26].

The *purpose* of this experiment is to investigate the impact of applying gamification on exploratory GUI testing. We have defined our research objectives using the Goal-Question-Metric template [27] as reported in Table 1.

The *goal* of the experiment is to gain an understanding of how gamification affects test case generation in terms of effectiveness and efficiency when traversing the GUI of a web application. In particular, if gamification has any positive effects on the generated test case. The experiment results are interpreted from the point of view of developers and testers of web applications, developers of testing tools, and researchers in the field of GUI testing.

TABLE 1: GQM template for the experiment

Object of study	Manual Exploratory GUI testing of web applications
Purpose	Compare traditional and gamified approach
Focus	Efficiency, Effectiveness
Context	Web applications
Perspective	Developers, Researchers

TABLE 2: Experiment Design

	Session 1		Session 2	
	Object	Tool	Object	Tool
Group 1	Mezzanine	Gamified	Wagtail	Vanilla
Group 2	Mezzanine	Vanilla	Wagtail	Gamified
Group 3	Wagtail	Gamified	Mezzanine	Vanilla
Group 4	Wagtail	Vanilla	Mezzanine	Gamified

4.1 Research questions

We define the following set of research questions to frame the experiment design.

- *RQ1*: Does gamification improve the *effectiveness* of test cases generated during manual exploratory testing of web applications compared to non-gamified manual exploratory testing?
- *RQ2*: Does gamification improve the *efficiency* of manual exploratory GUI test case generation for web applications compared to non-gamified manual exploratory testing?
- *RQ3*: Does gamification improve the *quality* of test cases generated during manual exploratory testing of web applications compared to non-gamified manual exploratory testing?
- *RQ4*: Does gamification improve the *User Experience (UX)* of manual exploratory testing of web applications compared to non-gamified manual exploratory testing?

It is worth underlining that the research questions are defined to cover related but different aspects that can be influenced by the application of gamification aspects on exploratory testing procedures: RQ1 focuses on absolute quantitative measures on the generated test cases; on the other hand, RQ2 focuses on the influence of gamification on the efficiency (i.e., results over time) of the tool.

4.2 Design

The primary focus of the study is the investigation of the resulting test cases generated from manual exploratory testing through the GUI of web applications. This investigation was organised as a within-subjects 2x2 full factorial (crossover) experiment where the treatment was administered to participants through two versions of a tool, with gamification (*Gamified*) and without (*Vanilla*).

All the participants received both treatments and had to test both the two subject applications in two consecutive tasks (periods). Two possible sequences occurred: *Gamified-Vanilla* and *Vanilla-Gamified*. In addition, there are two possible orders of applications (objects) for a total of four

different groups. The experimental design is summarised in Table 2. The participants were assigned randomly to the four groups.

After the second test session, the participants were asked to answer the TAM questionnaire about their experience [28]. The objective of the questionnaire was to investigate the usability of tool augmentation by means of game mechanics. The TAM questionnaire provides questions to evaluate four different high-level measures: *Perceived Usefulness* (PU), i.e., the degree to which an individual believes that using a particular system would enhance his or her performance; *Perceived Ease of Use* (PE), i.e., the degree to which an individual believes that using a particular system would be free of physical and mental effort; *Behavioural intention to use* (BI), i.e., the strength of one's intention to perform a specified behaviour; *Attitude towards usage* (ATU), i.e., the user's evaluation of the desirability for them to use the system [29]. Given the currently limited amount of game mechanics implemented and the strict correlation between them, we did not include questions about individual game elements in the questionnaire. Our focus was to assess the usability and acceptability of the system as a whole. Nonetheless, we arranged an open question to let participants express their suggestions and preferences. The interested reader can refer to section F of the survey script to find all the specific questions of the TAM questionnaire[2].

4.3 Operationalisation of Variables

Our experimental design is based on three independent variables. One main factor (*treatment*) corresponds to the testing tool, having two levels: Gamified and Vanilla, i.e. with/without gamification augmentation. Three controlled co-factors related respectively to the web application used as a test subject (see section 4.5), the order of the task, that is, experimental subject 1 or 2, and the sequence of treatments, i.e. Gamified-Vanilla and Vanilla-Gamified.

The dependent variables evaluated in the experiment relate to the proposed gamified approach's effectiveness, efficiency, and quality. In addition, the TAM constructs are used to assess the user experience.

4.3.1 Effectiveness

Effectiveness, as defined by the ISO 9001 standard [30], is *"the extent to which planned activities are realised and planned results are achieved"*. It was measured in the experiment using three variables: the *Coverage* of the generated test suites, the number of identified *True Positives*, and the *Bug Reporting Precision* of the test cases.

Coverage was chosen as a metric due to its regular use in software testing research and because of the correlation between greater coverage and more significant defect finding ability [31] [32].

More precisely, widget interaction coverage was measured. It is the proportion of widgets (e.g. buttons, dropdown menus or text fields) a participant has interacted with on a page over the total number of widgets on the page. As such, higher coverage is assumed to be better.

2. https://figshare.com/articles/online_resource/Question_forms/16537002

Session coverage is then computed as the average of all pages visited by the participant.

Regarding coverage metrics, we have only adopted metrics that can be computed at the GUI level of the SUT [33]. The reason for such selection is twofold: first, computing metrics at lower levels (model-based or code coverage metrics) would require instrumentation of the web application and hence limit the generalisability of their computation; second, code coverage metrics are not necessarily comparable between web applications developed with different languages or frameworks.

Next, since the experiment was performed with externally developed open-source software, the number of actual defects and their location were unknown. We assumed that no defects were originally present in the software, therefore we relied on the concept of mutation analysis, injecting artificial defects in the SUT. This approach allowed us to measure the defect finding ability of the test cases/suites generated by the experiment participants [34] as the number of *True Positive* test results, where a higher number is assumed to be better. However, since the number of reported defects varies among participants and includes both True- and False-Positive test results, the *Bug Reporting Precision* was also measured as the number of True-Positive results over the total number of reported defects. Once more, a higher number is assumed to be better.

4.3.2 Efficiency

Efficiency as *"the extent to which time, effort or cost is well used for the intended task or purpose"* [30].

In our experiment, we define *Efficiency* as the ratio between the number of reported bugs and the duration of the test session, i.e. *Reported bugs over time* [35]. For this metric, a higher value is perceived better. In addition, since one of the introduced gamification mechanics was to increase the tester's score based on exploration, we also measured the number of pages visited by each participant and the number of widgets they interacted with. These metrics were normalised by the duration of time spent on developing the test suite, i.e. resulting in *Visited pages over time* and *Interacted widgets over time*. In both cases, a higher value is considered better.

4.3.3 Quality

To study the *Quality* of test cases produced from the testing sessions, we evaluated the type of interactions carried on by the participants and the structure of generated test cases. First, since we modelled test sessions as tree structures with each path through the tree representing a test case, we measured the ratio between the breadth and the depth of the generated test tree (*Breadth-Depth Ratio*). Additionally, for each sequence, the ratio of added assertions per page traversed was calculated (*Assertion per page*). For this analysis, we only measured the assertions that are allowed by the specific tool on which we implemented our mechanics: the Scout tool allows to generate simple assertions about the presence or the content of an element, to be used in further re-executions of the test cases. Differences in measurements of this metric between the gamified tool and the vanilla tool would indicate that gamification affects how testers navigate the GUI.

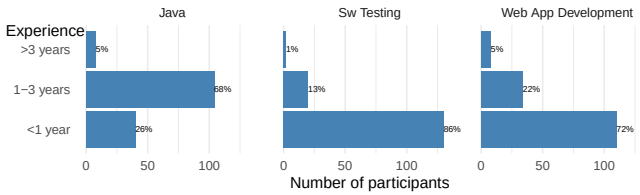

Fig. 3: Summary of participant's expertise

To complement this analysis, we applied the *Open coding* technique [36] to identify sequences of interactions that were common to many participants. After identifying the most common test cases, we performed a second pass on all the executed test cases to assign them to one category derived using open coding.

The final realism score was computed as a combination of the following five criteria (each worth one point): (i) the test case corresponds to one of the most common test cases that were coded in the *Open Coding* phase; (ii) at least one interaction was performed in each visited page; (iii) the test case performs operations on pages that are related (i.e., belonging to the same navigation tree in the web site hierarchy); (iv) the test case has a number of assertions which is lower than the number of instructions (e.g., clicks or types) performed; (v) the test case does not traverse the home page of the SUT multiple times.

This score was used as a proxy measure of the *realism* of the generated test cases. Realism is perceived as important since it is considered one of the benefits of the exploratory testing approach compared to model-based or random test generation techniques [37].

The four variables of the standard TAM (*Technology Acceptance Model*) questionnaire were considered as proxies to evaluate the *User Experience* (UX) of the tool. Each metric was measured on a 0-5 scale by computing the rounded average of all the responses to the related Likert sub-questions. For all metrics, in the results, a higher average value is considered positive.

4.4 Participants and Sampling

We recruited 144 participants for the experiment by convenience sampling. All participants were students at Politecnico di Torino, enrolled in a graduate-level Software Engineering course held during the spring semester of 2021. The students were encouraged to participate in the study and were given an additional point to their final course grade if they completed and delivered all the assigned tasks.

We also surveyed the participants' demographics, explicitly asking the participants about their expertise with object-oriented programming, web application development and testing (3-point ordinal scale: *less than one year*, *between one and three years*, *more than three years*), and whether they worked as professional software developers.

A summary of the responses is reported in Figure 3. Analysis of the survey showed that most students had less than one year of expertise in Software Testing (85%) and Web development (72%). In contrast, 68% of participants had between one and three years of expertise in Object-Oriented Programming. Additionally, 16% of the participants had previous working experience in the field.

4.5 Experimental subject applications

The task given to the experiment participants was to generate test cases for two different web-based SUTs manually. For the selection of the SUTs, we started from a list of open-source web applications available in grey literature[3]. All applications in the list were deployed and launched by one of the authors of the paper, who analysed the application structure and performed exploratory testing sessions with Scout, in order to evaluate the following inclusion criteria:

- The default instrumentation of the Scout tool had to be applicable to perform interaction sequences on the application without incurring any of the known limitations of the tool (e.g., the inability to send inputs to JavaScript forms);
- The applications should allow for the generation of test cases of reasonable length for scenario-based testing. We assumed a lower bound of at least five interactions per scenario and at least five statically loaded pages at startup. These measures were used as parameters in previous empirical GUI testing research [38].
- The application had to provide sufficient diversity of widgets. We verified this property by checking that the application featured at least one widget for each category of widgets mentioned in the HTTPArchive State of the Web report[4];

After applying the filtering procedure described above, 41 open-source were considered suitable for selection. From these, we randomly selected pairs of applications until we found applications that were comparable in terms of complexity. We assumed that the chosen applications had to be comparable in terms of the complexity of supported user interactions. We, therefore, measured the widget density of each application (i.e., the number of widgets per page). Comparisons based on the total number of widgets and pages were not deemed meaningful since many different applications in the pool allowed the user to create new pages dynamically.

At the end of such an iterative process, *Mezzanine*[5] and *Wagtail*[6] were selected as experimental objects. Since the Wagtail project already provides a set of 25 demo pages, one of the authors of this manuscript defined an equivalent amount of pages for the Mezzanine SUT, to provide the participants with the possibility of generating test sequences of the same length.

4.6 Instrumentation

We set up four virtual environments corresponding to the four combinations of the two independent variables – i.e. *Treatment* (Vanilla vs Gamification) and *SUT* (Mezzanine vs

3. https://github.com/unicodeveloper/awesome-opensource-apps
4. https://httparchive.org/reports/state-of-the-web
5. https://github.com/stephenmcd/mezzanine
6. https://github.com/wagtail/wagtail

Wagtail). This approach allowed the participants to conduct the experimental task independently on their own machines.

The testing environments were created with VirtualBox system images, using Ubuntu 18.04 OS, 4GB RAM, single CPUs without execution cap, and 15GB drive size. The VirtualBox images included, on their desktops, a link to a survey designed to guide the participants through the different phases of the experiment. The survey was also used to elicit additional information from the participants during the study.

To evaluate the defect-finding capability of generated test cases, we injected two sets (one per SUT) of twelve *static* bugs, in randomly chosen features/functions of the web applications. The injected bugs were the same for all sessions. Regarding mutants, the vanilla version of the tool does not feature any feedback mechanism when a mutant is found during a test case, i.e. the user is not aware of the presence of the mutant and they can only implicitly detect it – as if it were a regular bug – during the page exploration.

For this study, the choice of statically placed mutants was necessary to make the different test sessions comparable. Although placing fixed mutants in a particular SUT could be considered as a non-playful element, we argue that, since each participant is not aware of the actual mutant operators, and the bugs vary between the two sessions performed, a sufficient degree of randomness is maintained and the nature of the game element is not affected.

We injected mutants belonging to all the three macro-categories of mutation operators, i.e. addition, change and deletion. We randomly selected the pages and widgets in the SUTs to apply the mutant operators and the number of mutants for each macro category. Specifically, we introduced five delete mutants, four change mutants, three add mutants in Mezzanine, four delete mutants, three change mutants, and five add mutants in Wagtail. The mutants were added by modifying the source of the altered pages and injecting it on-the-fly in the emulated browser through Selenium API calls. A detailed description of the added mutants is reported as supplementary material in the online replication package for this manuscript[7].

The developed prototype only allowed for a comparison of the scores and results (e.g., coverage, exploration, progress) achieved on a single machine. Therefore it was not possible to show online updates of the highest score during our experimental sessions. We resorted to using asynchronous leaderboards, providing the same high scores to all the users on startup. These scores were obtained by four independent test sessions in each experimental subject to create an initial state, shared among all the participants, that simulated the behaviour of other testers. The four test sessions aimed at reproducing different attitudes of the testers, such as reaching higher coverage in few pages, exploring several pages with few interactions, or testing only one precise feature per test case. The game mechanics relying on other testers' metrics (leaderboards, exploration highlight, blue progress bar) were all based on these mock sessions. The use of simulated leaderboards and high scores is also necessary to avoid biases and make the users' results comparable. Since they are asynchronous by design, if we were not using the same fixed leaderboard, the users of the first sessions would only experience empty leaderboards. This would be in contrast with the primary use of leaderboards as intrinsic motivators: for that purpose, leaderboards must be populated with realistic results that are at the same time challenging and achievable.

Participant guidance and data collection were performed through a questionnaire. The first page of the questionnaire collected demographic information, followed by a 30-minute video tutorial about the basic functions of the experiment tool. After this preliminary phase, the participants were instructed to launch two shell scripts to run the correct version of the experiment tool and SUT, depending on which group they belonged to. At the end of each test session, the participants had to fill out a report about what bugs they found in the SUT. After the second test session, the TAM questionnaire was administered to the students to measure the user experience with the tool. Each test session lasted between 30 and 90 minutes.

The virtual machines and surveys can be found on the provided replication package.

The testing tool was instrumented to automatically log all the testers' interactions and collect the metrics defined to answer our research questions. *Bug Reporting Precision* and *Realism* were the only metrics that were measured through manual examination of bug reports entered in the questionnaire and session logs.

4.7 Hypotheses

To answer research questions 1, 2 and 3, the hypotheses in Table 3 have been formulated, for which the variables discussed in Section 4.3 were defined. Quantitative metrics are used to answer these questions through the use of formal statistical analysis, discussed further in Section 4.8.

TABLE 3: Null hypotheses for the experiment.

Name	Hypothesis
H_{c_0}	Gamification has no statistically significant impact on the *Coverage* of generated test suites.
H_{tp_0}	Gamification has no statistically significant impact on the *True Positives* found by generated test suites.
H_{p_0}	Gamification has no statistically significant impact on the *Precision* of reported bugs in the generated test suites.
H_{pt_0}	Gamification has no statistically significant impact on the number of *pages visited over time* during generation of test suites.
H_{wt_0}	Gamification has no statistically significant impact on the number of *widgets interacted with over time* during generation of test suites.
H_{rb_0}	Gamification has no statistically significant impact on the number of *bugs reported over time* during generation of test suites.
H_{bd_0}	Gamification has no statistically significant impact on the *Breadth-Depth ratio* of generated test suites.
H_{ap_0}	Gamification has no statistically significant impact on the number of *Assertions per page* in the generated test suites.
H_{r_0}	Gamification has no statistically significant impact on the *Realism* of the generated test suites.

4.8 Analysis Method

To test the hypotheses, we adopt a non-parametric approach since we expect most variables to not be normally

7. https://figshare.com/projects/GamificationReplicationPackage/127202

distributed. All statistical analysis is performed using the statistics tool, R [39].

Given the full factorial crossover design adopted for our experiment, as recommended by Vegas et al. [40], we analysed the data using a repeated measures linear mixed model also considering the sequence and order design factors to deal with the possible threats to validity deriving from the design. We analysed variance on such a model to check the statistical significance of the factors and their relative interactions. We opted for not using non-parametric tests since the sample size – 136 participants for a total of 272 data points – may suffice to make the central limit theorem hold [41], letting us interpret the results despite slight departures from normality. Moreover, linear mixed models have less constraints compared to other methods in the case of repeated measures ANOVA.

The following formula describes our data analysis:

$$X = c_0 + c_T \cdot Treatment + c_A \cdot Application + c_O \cdot Order + c_S \cdot Sequence + \text{Error}(Subject)$$

Where:

- X is any of the dependent variables,
- $c_.$, are the coefficients of the regression,
- $Application$, $Treatment$, $Sequence$, and $Order$ are the variables corresponding to the factors, and $Subject$ is the subject id variable.

We summarise the results reporting the p-values for each factor. Each hypothesis is evaluated at a 0.05 significance level (alpha) with two-tailed tests.

Since we perform tests on nine different dependent variables, we apply the Bonferroni correction [42] to compensate for the family-wise error rate. Although the use of this method is under debate [43], a specific case, when it is recommended, is when searching for associations without a predefined set of hypotheses supported by a consistent theory.

In practice, we divide the significance level by the number of tests performed on the same data set, i.e. 9. As such, any statistical test with a $p < 0.0056$ is considered significant and implies that the null hypothesis (H_0) for the said test must be rejected in favour of the alternative hypothesis (H_1). For example, if H_c0 in Table 3 has a statistically significant result, this would imply that gamification has an effect on coverage.

RQ4, regarding user experience, was evaluated using a questionnaire with answers that are not easily analysable using formal statistics. Instead, we opted for descriptive statistics, reporting the results in a stacked diverging bar chart [44]. Answers are grouped according to the four major TAM constructs: PU, PE, BI, and ATU.

4.9 Threats to Validity

We discuss the potential threats to the validity of the study according to the four categories reported by Wohlin et al. [45].

Conclusion validity threats concern drawing the appropriate conclusion based on test results. We employed non-parametric statistical tests that have essentially no statistical prerequisite. All measures were collected automatically. Thus we expect them to be not impacted by human errors in the collection. The treatment was administrated using two different operational environments, so the participants were randomly assigned to groups. No significant difference existed among the groups in terms of expertise. The participant allocation to groups is a confounding variable that could have influenced the statistical result, i.e. a different allocation could have resulted in other statistical test results. However, due to the number of participants, and their distribution of knowledge, in each group, this threat is considered improbable but cannot be eliminated entirely.

Construct validity threats concern the relationship between theory and observation. While there are a few widely accepted metrics in software testing, there is no single universally accepted mapping to abstract constructs such as effectiveness or efficiency; this is due to the inherent multifaceted nature of such concepts. In our approach, concerning efficiency, effectiveness, and test quality, we, therefore, opted for mapping each of them to three distinct metrics. This choice balances two contrasting issues: to cover as many facets as possible and to have a simple and easy-to-understand design.

To measure the true positives over time signalled in the test session, we had to resort to defining and injecting a set of pre-defined bugs in both applications. The number and nature of bugs were equivalent for both applications, so we do not expect an influence on the comparison between the average bugs found in different sessions. There is, still, a researcher bias related to the absolute discoverability of the introduced bugs, depending on the perceived visibility of the bugs injected by the individual researcher performing such a task. A second researcher bias is related to the manual addition of a set of default web pages in Mezzanine to make the SUTs comparable in size for the generation of exploratory sequences. Albeit the introduced pages were designed to have the same widget complexity as the default pages in Wagtail, there is a possibility that the added pages had an impact on the resulting measures for the metrics defined in the experiment.

As far as technology acceptance is concerned, we adopted a widely used questionnaire (Technology Acceptance Model) to have a generally accepted mapping. Albeit a custom questionnaire could have yielded more in-detail results about the participants' appreciation of the individual gamified constructs, using a consolidated evaluation framework allows the comparison of the results with other tools on which the evaluation has been already performed, or to compare the results with the evaluation of future gamified mechanics yet to be designed and implemented.

Internal validity threats concern factors that may affect a dependent variable and were not considered in the study. The validity of a crossover design can be exposed to threats related to fatigue and learning effects on the participants. We allowed the participants to execute the experiment remotely. We did not impose when the students had to perform the two sessions – to avoid concurrency with other academic tasks. However, we did impose a time limit for each session. The two test sessions did not have to be performed consecutively. However, we verified that most of the participants performed the second session immediately after the conclusion of the first one.

The adoption of a 2x2 factorial design, by construction, mitigates learning biases. However, since the gamified and vanilla treatments were administered with two versions of the two testing tools that share a common base, we expect a small learning effect on the results of the second testing session due to acquired familiarity with the tool base mechanics in the first one. The experimental design also reduces the fatigue effects: such effects were primarily experienced during the second session, where participants were evenly split between the two Treatments. Fatigue and learning could be detected by looking at the effects of the order in the analysis. Eventually, since the two tasks were performed in immediate sequence, we might observe a carryover effect, i.e., when a treatment is administered before the effect of another previously administered treatment has completely receded. Consequently, the treatments administered last might result in more (or less) effective than the first [40]. Such an effect might be revealed by a significant effect of the specific sequence (G-V or V-G) in the analysis.

We do not consider the incentive, consisting of one additional grade point, sufficient reason to have attracted more intrinsically motivated students as a threat towards the study or the results [46].

External validity threats concern the generalisability of the results. We expect an impact of the software subject used as SUT on the dependent variables. We selected experimental objects belonging to a popular category of web applications that feature an average number of different pages and an even distribution of web controls. The selection of only two subjects is however an inherent limitation for the generalisability of our experiment since we cannot guarantee that our results can be applicable to any category of web application, and even less so to applications belonging to other software domains.

Finally, we expect limitations in the generalisability of results regarding the representativeness of the injected bugs. To make the results of different test sessions comparable, we injected two fixed sets of twelve bugs in the SUTs. However, these manually defined bugs may not represent real, random bugs in actual web applications.

5 Results

Before performing the statistical analysis, we confirmed, using the Shapiro-Wilk test [47], that most variables are not normally distributed, thus suggesting the use of non-parametric tests.

We applied a pre-filtering of responses by evaluating the answers to the control questions of the questionnaire. No response had to be excluded for such a reason.

5.1 RQ1 - Effectiveness

In Table 4 we report the mean, median and standard deviation for the metrics selected to answer RQ1, i.e. average coverage per page, number of true positives found, and precision of reported defects. Figure 4 shows boxplots for the distribution of each metric aggregated by treatment. From a visual inspection, the gamified version of the tool appears to achieve higher average coverage, whereas the number of true positives detected and the bug reporting precision were both slightly higher for the vanilla version of the tool on average. The results of the ANOVA on the linear mixed-effects model for the measures related to RQ1 are reported in Table 5.

TABLE 4: Summary statistics for RQ1 (Effectiveness) metrics

		Mezzanine		Wagtail		All	
		V	G	V	G	V	G
Coverage	Mean	8.0%	8.9%	8.6%	10.8%	8.3%	9.9%
	Median	6.2%	7.4%	6.5%	8.8%	6.5%	8.1%
	Std. dev	6.8%	6.0%	5.5%	6.5%	6.2%	6.3%
True positives	Mean	1.87	1.57	3.68	3.58	2.75	2.58
	Median	2.00	1.00	1.00	4.00	2.00	2.00
	Std. dev	1.23	1.06	1.57	1.68	1.67	1.72
Precision	Mean	0.57	0.51	0.68	0.70	0.63	0.61
	Median	0.55	0.50	0.67	0.73	0.67	0.60
	Std. dev	0.33	0.30	0.22	0.23	0.29	0.28

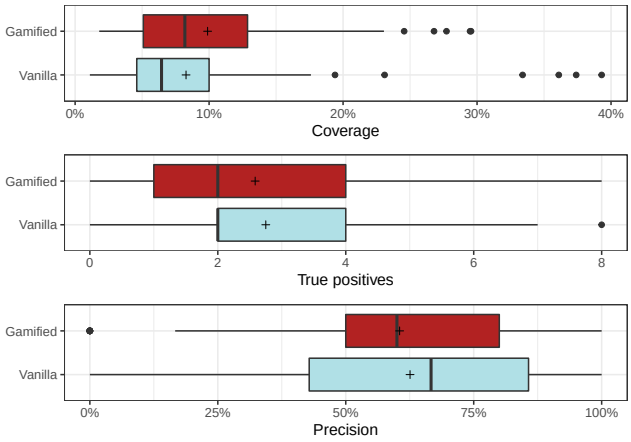

Fig. 4: Boxplots for RQ1 metrics

Based on the p-values, we can reject H_{c_0} (p = 0.0014) and conclude that the application of gamified mechanics has a statistically significant positive effect on the coverage achieved during exploratory GUI testing. The coverage is increased by 1.6%, which is a remarkable result when compared to an overall average of 9.9%.

The test results do not allow us to reject H_{b_0} (p = 0.28) and H_{tp_0} (p = 0.46). Therefore, we must conclude that neither the number of true positives nor the precision of how they are reported are affected by gamification. The Application had a significant impact on both the True Positives found and the measured Precision. We did not measure any significant impact by either the Order or the Sequence on the three metrics related to RQ1 hence we can conclude that we had no carryover or learning and fatigue effects in our experiment.

5.2 RQ2 - Efficiency

In Table 6 we report the mean, median and standard deviation for the metrics extracted from test sessions to answer RQ2, i.e. the ratio between visited pages and session duration, the ratio between interacted widgets and session duration, and the ratio between reported bugs and session duration. Figure 5 shows the boxplots for the distribution

TABLE 5: Results of Anova for RQ1 metrics: coefficients and p-values of individual factors

Factor / vars:	Coverage (H_{c_0})		True Positives (H_{tp_0})		Precision (H_{p_0})	
	coeff.	p-value	coeff.	p-value	coeff.	p-value
Treatment	0.0174	**0.0014**	-0.1681	0.2849	0.2849	0.4609
Order	-0.0071	0.2428	0.2346	0.0212	0.0267	0.2378
Application	0.0149	0.0075	1.9039	**<0.0001**	0.1508	**<0.0001**
Sequence	-0.0057	0.5247	-0.4177	0.0313	-0.0811	0.0314

TABLE 6: Summary statistics for RQ2 (Efficiency) metrics (time in minutes)

		Mezzanine		Wagtail		All	
		V	G	V	G	V	G
Pages / Time	Mean	0.61	0.49	0.59	0.49	0.60	0.49
	Median	0.54	0.42	0.53	0.45	0.54	0.44
	Std. dev	0.45	0.26	0.29	0.23	0.38	0.24
Widgets / Time	Mean	4.70	4.80	4.72	4.57	4.70	4.68
	Median	3.86	4.61	4.39	4.13	4.01	4.33
	Std. dev	2.81	2.22	2.31	2.32	2.58	2.27
Reported Bugs / Time	Mean	0.17	0.14	0.26	0.20	0.21	0.17
	Median	0.14	0.11	0.24	0.20	0.19	0.17
	Std. dev	0.11	0.08	0.13	0.09	0.13	0.09

of each metric aggregated by treatment. In Table 7 we report the results of the statistical analysis performed for the metrics regarding RQ2.

Regarding pages visited over time, the application of gamified mechanics (*Treatment* line in the table) has a significant (negative) effect (p = 0.0019). The mean of visited pages per minute decreases from 0.60 to 0.49 (-18.3%). As such, we must reject H_{pt_0}.

Regarding widgets interacted with over time, gamification did not affect significantly the number of widgets interacted per minute. As such, we cannot reject H_{wt_0} (p = 0.5).

Finally, the application of gamified mechanics has a significant (negative) effect on the number of reported bugs over time ($p < 0.0001$). The mean of reported bugs per minute decreases from 0.21 to 0.17 (-19%). As such, we must reject H_{rb_0}.

Regarding confounding factors, the test results point out a significant impact of the AUT on the reported bugs over time. We also observe a significant impact of the order of administration of the treatment on pages over time and bugs over time. This result can be interpreted as a possible fatigue effect. The Sequence variable had no significant impact on the measured metrics; hence we can conclude that we had no carryover effect in our experiment regarding RQ2.

5.3 RQ3 - Test case quality

In Table 8 we report the mean, median and standard deviation for the metrics used to answer RQ3, i.e. the ratio between *breadth and depth* of test cases, *number of assertions per page*, and *realism* of test cases. Figure 6 shows boxplots for the distribution of each metric aggregated by treatment. As can be observed, the differences between treatments are negligible for test realism and breadth-depth ratio. However, with gamification, a slightly larger difference was measured regarding the number of assertions per page.

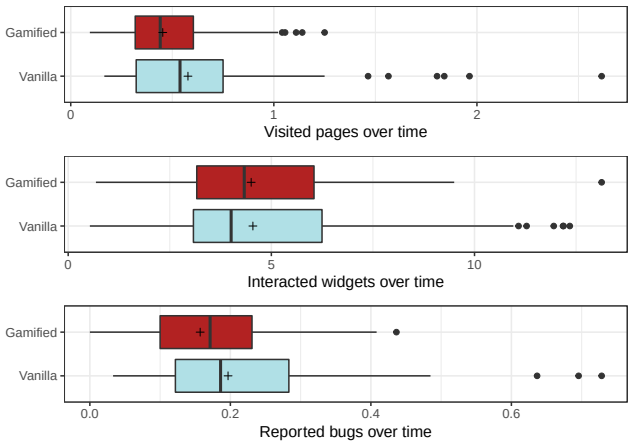

Fig. 5: Boxplots for RQ2 metrics

TABLE 7: Results of Anova for RQ2 metrics: coefficients and p-values of individual factors

Factor / vars:	Pages/Time (H_{pt_0})		Widgets/Time (H_{wt_0})		Bugs/Time (H_{rb_0})	
	coeff.	p-value	coeff.	p-value	coeff.	p-value
Treatment	-0.0854	**0.0019**	0.1132	0.5083	-0.0392	**<0.0001**
Order	0.1923	**<0.0001**	0.3445	0.0286	0.0461	**<0.0001**
Application	0.0033	0.9091	0.0566	0.7186	0.0741	**<0.0001**
Sequence	-0.0205	0.5729	0.0108	0.9764	-0.0046	0.7452

The results of the Anova for the variables related to RQ3 are reported in Table 9.

We observe a statistically significant effect of gamification on the average number of assertions per page; thus we reject H_{ap0} (p = 0.0001). The number of assertions is increased from 3.63 to 4.69 (+29%). No significant effect could be detected on the other dependent variables; thus we cannot reject H_{bd0} (p = 0.30) and H_{r0} (p = 0.92).

For what concerns confounding factors, we observe no effect of the order of tasks on any of the dependent variables. The SUT has a significant impact on the structure of the test cases (measured as the ratio between breadth and depth). Finally, the Sequence of tasks significantly impacted the realism of test cases. We speculate that this change was due to the fact that the used gamification elements encouraged some specific behaviours that are related to our definition of test case realism, and therefore a carryover effect could be experienced when gamified sessions were performed before non-gamified sessions.

5.4 RQ4 - User Experience

Figure 7 shows the distribution of the answers to the TAM Questionnaire, which all 144 participants of the study answered. To make the representation and the discussion of the results more compact, for each participant, we consider the rounded mean of the answers for each category of the questionnaire (i.e., Attitude towards Usage or *ATU*, Perceived Usefulness or *PU*, Behavioural Intention or *BI*, Perceived Ease of Use or *PE*) [48]. We observe, on average, that participants had mostly positive perceptions towards all the metrics measured by the TAM model. The metric with the most positive responses was the Attitude towards

TABLE 8: Summary statistics for RQ3 (test case quality) metrics

		Mezzanine V	Mezzanine G	Wagtail V	Wagtail G	All V	All G
Breadth / Depth	Mean	0.34	0.36	0.44	0.46	0.39	0.41
	Median	0.32	0.31	0.42	0.41	0.35	0.37
	Std. dev	0.13	0.17	0.17	0.23	0.15	0.21
Assertions / Page	Mean	3.77	4.45	3.47	4.92	3.63	4.69
	Median	2.51	3.07	2.33	3.71	2.45	3.63
	Std. dev	5.25	4.63	3.34	4.25	4.42	4.43
Realism	Mean	3.50	3.52	3.48	3.44	3.49	3.48
	Median	3.67	3.65	3.66	3.64	3.67	3.64
	Std. dev	0.77	0.74	0.79	0.80	0.78	0.77

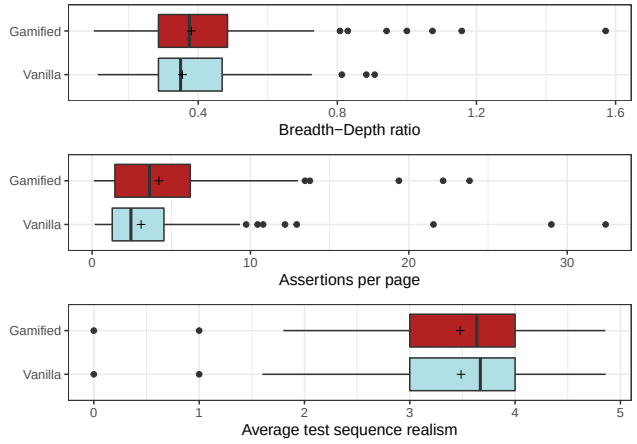

Fig. 6: Boxplots for RQ3 metrics

TABLE 9: Results of Anova for RQ3 metrics: coefficients and p-values of individual factors

Factor / vars:	Breadth/Depth (H_{bd_0}) coeff.	Breadth/Depth (H_{bd_0}) p-value	Assertions/Page (H_{apo}) coeff.	Assertions/Page (H_{apo}) p-value	Realism (H_{r_0}) coeff.	Realism (H_{r_0}) p-value
Treatment	0.0199	0.3030	1.2326	**0.0001**	-0.0098	0.9206
Order	-0.0221	0.4131	-0.8272	0.0097	-0.0433	0.6249
Application	0.1020	<**0.0001**	0.3470	0.2645	-0.0459	0.6211
Sequence	0.0104	0.6258	-01365	0.8290	-03415	**0.0003**

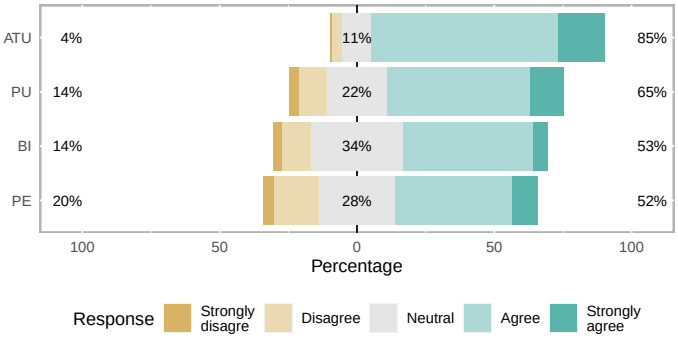

Fig. 7: Distribution of the answers to the TAM Questionnaire

Usage metric, with 17% of participants strongly agreeing and 66% agreeing that gamification is a desirable addition to the practice of exploratory testing. High positive perceptions were also measured for Perceived Usefulness and Behavioural Intention. These results suggest that most of the participants would use tools with gamification if they had to perform testing activities in the future. Additionally, it indicates that gamification was perceived as valuable and usable.

The most negative perceptions were aimed toward Perceived Ease of Use (16% Disagree, 4% Strongly disagree). This result, however, can be explained by the prototypical nature of the tool, and the administration of the experimental sessions through a virtual machine that could have lowered the overall usability of the tool.

By considering the TAM metrics as proxies for user experience and usability of a software system, we can state that the overall good results we gathered suggest a good UX for our gamified prototype tool.

Participants' suggestions, collected by means of open questions in the questionnaire, were mainly related to the inclusion of profile customisation (e.g., with avatars) and reward mechanics such as badges. Negative textual comments were mostly related to technical issues and low responsiveness experienced with the virtual machine.

6 Discussion

For this study, gamification was added to a testing tool to investigate the impact on manual exploratory testing. We evaluated the impact of added gamification mechanics in terms of effectiveness, efficiency, and realism of the generated test cases.

The most evident result that we verified is a statistically significant increase in the coverage obtained with the gamified testing tool. In fact, gamification induced the participants to focus more on the exploration of the pages of the SUT. The main explanation for this behavioural change is the addition of gamification mechanics, especially the progress bar. This feature shows the user's current coverage of the currently tested page and, due to the instant feedback, it enables the user to see when they are making progress; thus it helps guide their efforts, rewarding them for being thorough. In practice, testers are highly motivated to perform more interactions on the same page to increase the coverage – visually shown by the progress bar – instead of deeply exploring the whole page tree of the SUT. This result was indeed expected since the gamification elements implemented in the prototype tool were designed with increased coverage as one of the main objectives. Therefore, this result can serve as a demonstration that tailored gamification mechanics can successfully modify testing habits when designed with a specific purpose. Regarding test effectiveness, no other significant effects (neither positive nor negative) were observed with the introduction of gamification.

Regarding the efficiency of the exploratory testing process, we measured statistically significant negative effects of gamification mechanics: a reduction of the number of pages visited over time, and a reduction of the number of bugs reported over time with the issue signalling feature of the Scout tool. Albeit these effects may suggest a detrimental effect of the application of gamified mechanics in general, it is worth underlining that the implemented mechanics demanded attention from the testers for the number of interactions performed on the same page, and newly interacted widgets. Therefore, it is reasonable that the efficiency in performing operations can be lowered when participants are

encouraged to perform more focused operations on the SUT. Hence, each application page was tested more thoroughly as opposed to a shallower exploration of pages observed without gamification. We hypothesize that the adoption of gamified elements designed specifically for better efficiency (e.g., timers) could have an opposite impact on such metrics. Our prototype tool actually evaluated the pace of the performed interactions as one of the components of the final score of the user - we however suppose that such a scoring component was not sufficiently evident to the participants to generate biased behaviours towards faster interactions with the SUT. We, therefore, suggest that such mechanics ought to be additionally evaluated and calibrated to ensure that the benefits guaranteed by some gamified elements are not provided at the cost of the overall efficiency of the testing technique.

Regarding the quality of the generated test sequences, we measured a significant increase in the number of generated assertions per page when gamified mechanics are used. Conversely, we observed no significant changes in the realism of test cases – measured as a function of multiple properties of exploration sequences, as described in section 4.3.3 – neither in the breadth over depth of test cases. The higher number of assertions per page can be justified again as the outcome of a changed behaviour by the testers, who were encouraged to stay more focused on individual pages rather than moving onto different ones. This reasoning also justifies the small (and not significant) increase in the ratio between the breadth and depth of generated test cases. The added assertions in the generated exploratory sequences could also be an object of game exploit by the participants, for instance by putting random assertions on irrelevant web elements or adding an assertion on the same web element multiple times. We therefore performed an analysis of the logs of the generated sequences, and we verified that no participant performed any exploit.

It is worth stressing that the nature of the test suites and of the generated assertions was not influenced by the adoption of gamification elements. This result is important because the adoption of highlighting mechanisms like the progress bar could have encouraged the participants to perform random operations on the SUT to increase their scores. On the contrary, the produced test cases followed the main use cases of the experimental objects, resulting in mostly valid test cases for the SUTs. We can argue, based on such results, that testers, during exploratory testing, limit their exploration to sequences that make cognitive sense rather than random sequences. The fact that gamified mechanics do not affect the realism and structure of test cases is a strong point in favour of gamification adoption.

Concerning the acceptance by users, results showed that the proposed gamification approach led to a reasonable user experience and good usability. As such, the approach can provide solid support in practice, mitigating manual testing challenges (i.e. that it is costly and frequently perceived as an unappealing activity) while preserving all the benefits of human test case generation over automated or random generation.

Finally, we remark that the present study results only allow us to consider the impact of all gamified mechanics as a whole without discriminating the impact of each mechanic taken individually. Further experiments would be needed to pursue that end. An example of still needed experimentation is an evaluation of the mutant injection mechanism. The current analysis has only employed static, researcher-injected mutants in both the SUTs. Also, since one of our research questions was about the bug-finding effectiveness of generated test sequences, the same mutants were injected into the non-gamified version of the tool, to enable comparability between the test sequences. As such, injected mutants by now are a means to increase the participants' engagement and to complement the evaluation of the other gamification mechanics that have been implemented. Future research efforts are needed to evaluate the mutant injection mechanism in isolation and to understand how the injection of random mutants can impact other metrics that are not directly related to bug finding (e.g., on exploration or coverage metrics). More specific experimental settings would also be needed to evaluate the impact of the leaderboard mechanic. In the described evaluation, since each participant conducted only one task with the gamification, this feature was highly unlikely to alter their behaviour.

6.1 Current Limitations

The main objective of the present paper was to provide insights on the results of the application of the basic concepts of gamification to an existing testing tool. The technical contribution (i.e., implementation of advanced gamification concepts) was not the focus of the paper. Regarding the type of mechanics, we focused on the implementation of the gamification elements that could be evaluated in the parallel experimental setting. Other mechanics (e.g., profile creation, achievements, social interaction) would have required longitudinal course-long projects to be evaluated properly.

One of the main technical limitations of the current version of the tool is the missing support for online synchronous competition between different *players*. This limitation is however not perceived as detrimental for the evaluation described in the present paper, since an unbiased evaluation of leaderboards and highest score mechanics would have in any case required the definition of a stable asynchronous starting point for all the involved users. Enabling live competition (or collaboration) would however allow for the implementation of more refined mechanics and provide the basis for additional analyses about the most impactful gamification elements in our setting. The need for an unbiased setup led to a similar approach regarding the *Bug Killing Score* mechanic: to allow for comparable results between different users, we resorted in providing a static set of bugs for all the testers of the same SUT. The tool already contains an engine capable of generating random bugs across the different pages explored by the tester.

A second limitation of the tooling adopted for the experimentation is the absence of a nuanced possibility of adding assertions. The Scout tool allows only to verify the presence or the textual content of a widget encountered during an exploratory session. On one hand, this limitation in the assertion reduces the variability in the complexity of the created assertions, which would make impossible a comparative evaluation of the generated test sequences. On the other hand, since many assertions can be easily placed

with such mechanism in the same or similar pages, there is a theoretical possibility of exploiting such a mechanism by placing multiple assertions on a page which is known to be particularly stable. It is, however, not possible to generate assertions that are always true by definition.

Finally, some limitations are due to the type of SUTs considered. Web applications are typically large in terms of number of pages available - in some cases, e.g. for CMS systems, additional pages can be created at run time during the execution of test scenarios. Without the implementation, as of now missing, of crawlers of the full application structure, it was not possible to provide a measure of the widget coverage over all the widgets of the web application. As of now, the tool only computes the widget coverage over the set of visited pages, by adding the normalized widget coverage for each page that is encountered by the tester in their test session. This way of computing coverage is indeed not optimal - since by construction, it creates a temporary drop in the overall average coverage every time a new page is discovered. Future work may be conducted to analyze alternatives to the current widget coverage formula. The use of web applications as SUTs also poses generalizability issues when computing coverage at levels that are different from that of the GUI components. Even though through instrumentation of the backend it is possible to compute code-level coverage when performing exploratory testing of web applications through the GUI, the high variability of frameworks and languages that can be adopted in the development stack of web applications invalidates any assumption on code coverage measures on different SUTs.

7 Conclusion and Future work

In this study, we have investigated the impact of gamification on manual exploratory testing. In particular, we defined a set of game mechanics and implemented them by adding such gamified elements to a tool called Scout.

The main element of novelty of our tool is that it is applied at a different testing level from those existing in related work, which applies gamified mechanics to software test generation. Available research efforts have proposed gamified mechanics to improve the effectiveness of test case generation at the unit level and for mutation testing (e.g., [16], [18]). Instead, to the best of our knowledge, the set of mechanics that we discuss constitutes the first non-educational gamified testing framework applied to end-to-end exploratory GUI testing.

We ran an experiment comparing 144 human participants' performance when using the tool with and without gamification. In particular, gamification was compared to the non-gamified version of the tool in terms of effectiveness, efficiency, quality of tests and user experience.

The study showed that gamification provides higher page coverage than non-gamified testing from an effectiveness standpoint. From an efficiency standpoint, gamification has shown to have no beneficial effects. Instead, non-gamified testing was found to result in both more pages being tested over time and more bugs being reported over time. In terms of test quality, gamification resulted in an addition of more assertions. Finally, as implemented, gamification was considered to provide a good user experience.

In summary, although the number of variables of significance in the experiment was underwhelming, these results still indicate the value of gamification and its impact on the testers' behaviour. As such, we propose that further research is warranted into gamification features that aid testers through test augmentation and guide the tester's behaviour in different ways depending on testing needs.

To overcome the tool-based limitations described in section 6.1, our primary future work will include implementation efforts targeted to the construction of a more flexible framework to implement further gamification mechanics, to assist the integration of more refined gamification elements in the practice of exploratory testing. Some examples of more advanced gamification elements are avatars, achievements, and narrative metaphors of the testing activities being carried on. Research should also focus on the empirical evaluation of each element in isolation, to pinpoint which effects on testing practice, user experience and user behaviour can be associated with each element. It would also be beneficial to extend the set of experimental subjects used in the current study, also considering multiple categories of web applications, to enhance the generalisability of our findings. Finally, we also plan to evaluate the possibility of introducing multi-tester activities (either online or offline) in order to combine the benefits provided by crowdsourced architectures and gamified testing.

Acknowledgment

This study was carried out within the "EndGame - Improving End-to-End Testing of Web and Mobile Apps through Gamification" project (2022PCCMLF) – funded by the Ministero dell'Università e della Ricerca – within the PRIN 2022 program (D.D.104 - 02/02/2022). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

References

[1] E. Borjesson and R. Feldt, "Automated system testing using visual gui testing tools: A comparative study in industry," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 350–359.

[2] O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering–a systematic mapping," *Information and software technology*, vol. 57, pp. 157–168, 2015.

[3] M. V. Mäntylä and K. Smolander, "Gamification of software testing-an mlr," in *International conference on product-focused software process improvement*. Springer, 2016, pp. 611–614.

[4] G. M. de Jesus, F. C. Ferrari, D. de Paula Porto, and S. C. P. F. Fabbri, "Gamification in software testing: A characterization study," in *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing*, 2018, pp. 39–48.

[5] O. Liechti, J. Pasquier, and R. Reis, "Supporting agile teams with a test analytics platform: a case study," in *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*. IEEE, 2017, pp. 9–15.

[6] G. Fraser, "Gamification of software testing," in *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*. IEEE, 2017, pp. 2–7.

[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] 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.

[9] T. Lorey, S. Mohacsi, A. Beer, and M. Felderer, "Storm: A model for sustainably onboarding software testers," *arXiv preprint arXiv:2206.01020*, 2022.

[10] J. Itkonen, M. V. Mäntylä, and C. Lassenius, "The role of the tester's knowledge in exploratory software testing," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 707–724, 2012.

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

[12] Z. Long, G. Wu, X. Chen, W. Chen, and J. Wei, "Webrr: self-replay enhanced robust record/replay for web application testing," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1498–1508.

[13] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: using gui screenshots for search and automation," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, 2009, pp. 183–192.

[14] M. Nass, E. Alégroth, and R. Feldt, "Augmented testing: Industry feedback to shape a new testing technology," in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2019, pp. 176–183.

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

[16] G. Fraser, A. Gambi, M. Kreis, and J. M. Rojas, "Gamifying a software testing course with code defenders," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 571–577.

[17] S. Elbaum, S. Person, J. Dokulil, and M. Jorde, "Bug hunt: Making early software testing lessons engaging and affordable," in *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 2007, pp. 688–697.

[18] R. M. Parizi, "On the gamification of human-centric traceability tasks in software testing and coding," in *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 2016, pp. 193–200.

[19] T. Laurent, L. Guillot, M. Toyama, R. Smith, D. Bean, and A. Ventresque, "Towards a gamified equivalent mutants detection platform," in *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2017, pp. 382–384.

[20] J. Wang, M. Li, S. Wang, T. Menzies, and Q. Wang, "Images don't lie: Duplicate crowdtesting reports detection with screenshot information," *Information and Software Technology*, vol. 110, pp. 139–155, 2019.

[21] F. Pastore, L. Mariani, and G. Fraser, "Crowdoracles: Can the crowd solve the oracle problem?" in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, 2013, pp. 342–351.

[22] Y. Chen, M. Pandey, J. Y. Song, W. S. Lasecki, and S. Oney, "Improving crowd-supported gui testing with structural guidance," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.

[23] C.-I. Lee, I.-P. Chen, C.-M. Hsieh, and C.-N. Liao, "Design aspects of scoring systems in game," *Art and Design Review*, vol. 5, no. 1, pp. 26–43, 2016.

[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.

[25] A. Jedlitschka and D. Pfahl, "Reporting guidelines for controlled experiments in software engineering," in *2005 International Symposium on Empirical Software Engineering, 2005.* IEEE, 2005, pp. 10–pp.

[26] P. Ralph, S. Baltes, D. Bianculli, Y. Dittrich, M. Felderer, R. Feldt, A. Filieri, C. A. Furia, D. Graziotin, P. He, R. Hoda, N. Juristo, B. A. Kitchenham, R. Robbes, D. Méndez, J. Molleri, D. Spinellis, M. Staron, K. Stol, D. A. Tamburri, M. Torchiano, C. Treude, B. Turhan, and S. Vegas, "ACM SIGSOFT empirical standards," *CoRR*, vol. abs/2010.03525, 2020. [Online]. Available: https://arxiv.org/abs/2010.03525

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

[28] Y. Lee, K. A. Kozar, and K. R. Larsen, "The technology acceptance model: Past, present, and future," *Communications of the Association for information systems*, vol. 12, no. 1, p. 50, 2003.

[29] C. Gardner and D. Amoroso, "Development of an instrument to measure the acceptance of internet technology by consumers," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, 2004, pp. 10 pp.–.

[30] "Iso 9001:2005," International Organization for Standardization, Standard, 2005.

[31] A. Mockus, N. Nagappan, and T. T. Dinh-Trong, "Test coverage and post-verification defects: A multiple case study," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 291–301.

[32] A. S. Namin and J. H. Andrews, "The influence of size and coverage on test suite effectiveness," in *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, ser. ISSTA '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 57–68. [Online]. Available: https://doi.org/10.1145/1572272.1572280

[33] R. Coppola and E. Alégroth, "A taxonomy of metrics for gui-based testing research: A systematic literature review," *Information and Software Technology*, p. 107062, 2022.

[34] R. A. Oliveira, E. Alégroth, Z. Gao, and A. Memon, "Definition and evaluation of mutation operators for gui-level mutation analysis," in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2015, pp. 1–10.

[35] A. Gupta and P. Jalote, "An approach for experimentally evaluating effectiveness and efficiency of coverage criteria for software testing," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 2, pp. 145–160, 2008.

[36] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research: a critical review and guidelines," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 120–131.

[37] O. Stadie and P. M. Kruse, "Closing gaps between capture and replay: Model-based gui testing," in *1st INTUITEST Workshop*, 2015.

[38] E. Borjesson, "Industrial applicability of visual gui testing for system and acceptance test automation," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 475–478.

[39] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/

[40] S. Vegas, C. Apa, and N. Juristo, "Crossover designs in software engineering experiments: Benefits and perils," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 120–135, 2015.

[41] J. C. De Winter, "Using the student's t-test with extremely small sample sizes," *Practical Assessment, Research, and Evaluation*, vol. 18, no. 1, p. 10, 2013.

[42] J. M. Bland and D. G. Altman, "Multiple significance tests: the bonferroni method," *Bmj*, vol. 310, no. 6973, p. 170, 1995.

[43] T. V. Perneger, "What's wrong with bonferroni adjustments," *Bmj*, vol. 316, no. 7139, pp. 1236–1238, 1998.

[44] J. Bryer and K. Speerschneider, *likert: Analysis and Visualization Likert Items*, 2016, r package version 1.3.5. [Online]. Available: https://CRAN.R-project.org/package=likert

[45] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[46] P. C. Jordan, "Effects of an extrinsic reward on intrinsic motivation: A field experiment," *Academy of Management Journal*, vol. 29, no. 2, pp. 405–412, 1986.

[47] P. Royston, "Approximating the shapiro-wilk w-test for non-normality," *Statistics and computing*, vol. 2, no. 3, pp. 117–119, 1992.

[48] J. R. Lewis, "Comparison of four tam item formats: Effect of response option labels and order." *Journal of Usability Studies*, vol. 14, no. 4, 2019.

Riccardo Coppola Riccardo Coppola received the M.Sc. degree in computer engineering from Politecnico di Torino, Turin, Italy, where he is currently working towards a PhD degree. His research interests

include automated GUI testing for web and mobile applications and the evaluation of non-functional properties of testware.

Tommaso Fulcini Tommaso Fulcini received a M.Sc. degree in computer engineering, Software branch, from Politecnico di Torino, Turin, Italy, where he is currently working towards a PhD degree. His research interests include gamification in Software Engineering, particularly software testing. He is currently working on building a gamified environment for end-to-end testing.

Luca Ardito Luca Ardito is an Assistant Professor at Dept. of Control and Computer Engineering of Polytechnic of Turin, where he works in the Software Engineering research group. He received BSc, MSc, and PhD in Computer Engineering from Politecnico di Torino. His current research interests are mobile development and testing, gamification in software engineering, green software and empirical software engineering methodologies.

Marco Torchiano Marco Torchiano is an associate professor at the Control and Computer Engineering Dept. of Politecnico di Torino, Italy; he has been a post-doctoral research fellow at the Norwegian University of Science and Technology (NTNU), Norway. He received an MSc and a PhD in Computer Engineering from Politecnico di Torino. He is a Senior Member of the IEEE and a member of the software engineering committee of UNINFO (part of ISO/IEC JTC 1). He is the author or co-author of over 140 research papers published in international journals and conferences of the book “Software Development—Case studies in Java” from Addison-Wesley, and co-editor of the book “Developing Services for the Wireless Internet” from Springer. He recently was a visiting professor at Polytechnique Montréal studying software energy consumption. His current research interests are green software, UI testing methods, open-data quality, and software modelling notations. The methodological approach he adopts is that of empirical software engineering.

Emil Alégroth Emil Alégroth is an Assistant Professor at the Software Engineering Research Lab (SERL) at the Blekinge University of Technology in Karlskrona, Sweden. His research has primarily focused on automated GUI testing, in particular Visual GUI Testing, which was the topic of his dissertation that he defended in 2015. Emil’s research is also empirical in nature, performed in collaboration with industry, including companies such as Spotify, Saab, Jeppesen and Grundfos.