Conceptualization of Multi-User Collaborative GUI-Testing for Web Applications

Conceptualization of Multi-User Collaborative GUI-Testing for Web Applications

Riccardo Coppola[0000−0003−4601−7425], Tommaso Fulcini[0000−0001−8765−6501], and Marco Torchiano[0000−0001−5328−368X]

Politecnico di Torino, Turin, Italy
{name.surname}@polito.it

Abstract. System testing through the Graphical User Interface (GUI testing) is often overlooked in real-world software projects. The mechanics proper of Gamification are often adopted to increase the motivation and engagement of testers in such scenarios. There is however little evidence in the literature about collaborative – rather than competitive – Gamification mechanics for software testing. In this manuscript, we describe a prototype for a collaborative GUI-testing prototype for web applications based on micro-tasks, of which we perform a small-scale preliminary evaluation. We find that the use of a microtask-based collaboration environment incentivizes depth-first testing of web applications and increases local coverage. We also collected positive experience reports from practitioners about the practice.

Keywords: GUI Testing, Crowdsourcing, Web Testing, Software Testing

1 Introduction

The creation of automated software test suites is a fundamental practice in Software Engineering, but is often seen as a tedious and time-consuming activity and is therefore neglected by developers. However, there are several testing activities that are crucial for modern-day software. For web applications, it is crucial to perform System testing through the Graphical User Interface (GUI) of the applications, to ensure that the use cases and scenarios of the application are correctly carried out and that the user can obtain the desired results by interacting with the application in the browser. Evidence in the software engineering literature highlights that there is a general preference in development teams towards manual testing of the interfaces, instead of adopting specific tooling and structured testing methodologies [17].

To increase the motivation and engagement of testers, one solution that has been many times used in literature is Gamification. Gamification consists of the use of elements, philosophies and mechanics that are typical of game design in non-playful contexts. The assumption behind the adoption of such practice is that – with the use of Gamified mechanics – testers may perceive the testing process as more entertaining and engaging, increasing the number of generated

test cases and as a consequence ensuring a higher quality for the applications [2].

Gamification has an inherent social aspect since the results and achievements of the practitioners are typically compared with those of others, to provide motivation and incentive best practices. While this competitive aspect of gamification has been deeply investigated, there is very little evidence in the literature about the collaborative uses of gamification [10].

The objective of this work is to extend an existing gamified web application testing tool with elements of collaborative gamification and crowdsourcing, with the final objective of using the collective efforts of a multiplicity of collaborating peers to obtain better results in a web application testing activity. To the best of our knowledge, this is the first practical approach that subdivides a complex test case to enable crowdsourcing in visual GUI testing, instead of simply providing complex test cases to the crowd.

The remainder of the manuscript is organized as follows: in Section II, we provide background information about GUI testing, Gamification, and Crowdsourcing; in Section III, we illustrate our assumption about collaboration in GUI testing for web applications and describe the prototype framework we used; in Section IV, we report the results of a small-scale preliminary evaluation of the tool; in Section V, we provide a conclusion to this investigation and describe future directions for our work.

2 Background and Related Work

This section introduces related work in the literature about GUI Testing and the applications in the practice of Gamification and Crowdsourcing.

2.1 GUI Testing

GUI testing is the activity of verifying the functional correctness of a software product by interacting with its Graphical User Interface. GUI testing is typically based on the interaction with individual atomic elements of the user interfaces (called *web elements* or *widgets*) and the verification of their properties after *sequences* of interaction are executed.

GUI testing techniques, activities and tools can be classified in different ways. One first classification, proposed by Alégroth et al., organizes them according to the way the widgets on the screen are identified and the used properties for functional verification [1]: first-generation (or coordinate-based) GUI testing techniques, now mostly abandoned, identify elements on the screen by using their exact x and y coordinates on screen; second-generation (or property-based) use textual properties of the GUI-description (typically the DOM model for web applications); third-generation (or visual) use graphic clues and computer vision techniques to identify and verify widgets through their exact visual appearance.

Linares-Vasquez et al. propose another classification based on the way the test sequences are created against the widgets in the GUI [12]: script-based testing

techniques take advantage of custom scripting languages to code repeatable test sequences; capture and replay techniques collect the inputs of a human user with the GUI to generate replayable sequences; model-based testing techniques use (or generate) a model of the GUI (e.g. a finite state machine or a directed graph) to automatically generate test sequences to obtain maximum model coverage; automated input generation techniques use specific criteria (from pure random to search-based) to select widgets to interact within the GUI.

A recently introduced technique for GUI testing is that of Augmented Testing, which was presented by Nass et al. to address the challenges of frequent and costly maintenance of test scripts, and the difficulty of obtaining high screen and widget coverage with traditional Capture and Replay tools [14]. The technique involves an additional interface overlaid on that of the System Under Test, providing the user with test suggestions and data, highlighting important areas, or offering other forms of support, while also allowing observation and recording of inputs. The Augmented GUI is the only interface presented to the tester, and interaction occurs solely through it. A driver translates the received inputs to the SUT's GUI. Possible inputs include checks, issues, and augmented actions, which are saved in a model and then replayed on the SUT's interface. The tool Scout was proposed to demonstrate the benefits of augmented testing and is specific to web applications[14]. Scout is a Capture and Replay tool for Augmented Testing characterized by a closed-source core that implements the logic of the state model but with other functionalities left to ad hoc plugins, including the *SeleniumPlugin* that links Scout with the target web page under testing. In Scout, the model in which the tester's interactions are saved is represented as a weighted graph, where each node represents a state of the application. Each state includes all dependencies such as internal memory, saved data, and external applications at a given moment. Nodes are interconnected by actions, which are the tester's actions, such as mouse clicks or keyboard inputs.

2.2 Gamification in Software Engineering

Gamification is a strategic attempt to improve systems, services, organizations, and activities to motivate and engage users. A comprehensive definition of Gamification is provided by Deterding, as "the use of game design elements for games in non-game contexts" [8]. This differs from the concept of *serious games*, which refers to actual games with typically educational purposes. Gamification aims to enhance non-gaming activities by increasing user productivity, facilitating learning and knowledge retention, improving the usability of provided tools, and many other benefits [11].

Several frameworks have been provided to characterize efforts in gamification. A popular framework is the Octalysis, defined by Yu-Kai Chou [18]. The Octalysis framework identifies 8 core drives for gamification, and defines a set of game mechanics for each drive: (i) Epic Meaning and Calling, i.e. the desire to participate in something greater than just the activity which is gamified; (ii) Development and Accomplishment, i.e. game elements associated with progress, such as points, badges, and rankings indicating progress; (iii) Empowerment of

Creativity and Feedback, i.e., challenges where users must express their creativity to overcome them, receiving immediate feedback on their actions; (iv) Ownership and Possession:, i.e. possess of something tangible inside gamified enviromments; (v) Social Pressure and Relatedness, i.e. the introduction of social concepts like collaboration and competition; (vi) Scarcity and Impatience, i.e. fostering specific gamified aspects by making them appear scarce in the gamified context; (vii) Unpredictability and Curiosity, i.e. fostering gamified aspects by making them ever-evolving and unpredictable; (viii) Loss and Avoidance, i.e., the possibility of something negative happening, such as losing something that has been built over time, driving people to continue their actions.

Over the years, there have been numerous attempts to apply gamification in Software Engineering education, to increase engagement, enhance performance and engagement, and incentivise best practices taught in courses [2].

Several works in Software Engineering literature have applied Gamification to the practice of software testing [10]. A prominent example of gamification in testing is Code Defenders, developed by Rojas and Fraser [16]. This tool is specific to mutation testing, which involves creating slightly modified versions of the same program called mutants. Parizi et al. implemented GamiTracify, to encourage developers to trace all artifacts involved in specific test cases during test case writing and provides incentives for doing so [15]. Coppola et al. applied a framework of gamified mechanics to GUI testing of web applications, proving that gamification has a tangible effect on the obtained coverage of generated test sequences [7].

2.3 Crowdsourcing in Software Engineering

Crowdsourcing is defined as the collective execution of a given activity, tailored to involve groups of participants. The use of crowdsourcing brings many advantages like lower costs, speed in conducting tasks, flexibility and scalability of the jobs. Crowdsourcing can be classified as *explicit*, where the users collaborate to complete specific and well-defined tasks, and *implicit*, where the users solve problems as a collateral effect of other activities that they are performing.

Testing with the aid of crowdsourcing has the advantage of being able to utilize ordinary users without specific experience to perform testing tasks. In the study by Mao et al., several examples of crowdsourcing applications are presented [13]. These activities include functionality testing, performance testing, GUI testing, automatic test case generation, and the oracle problem. Functional testing is a significant portion of the time dedicated to testing in the software development cycle and can be quite expensive. This type of testing, with the assistance of crowdsourcing, has demonstrated the ability to uncover potential issues as effectively as employing experts. The advantages of using crowdsourcing are reflected in monetary terms, product delivery speed, and accessibility to this practice.

Chen et al. proposed a tool to increase testing coverage, assisting the users by showing information on navigation paths that have been already performed by other testers [5].

Wang et al. developed an approach named SETU (ScrEenshots and TextUal description) [19]. The approach allows crowd testers to characterize screenshots of test sequences with image features and textual descriptions, whilst creating a hierarchical algorithm to detect duplicate information between different sessions.

3 Methodology

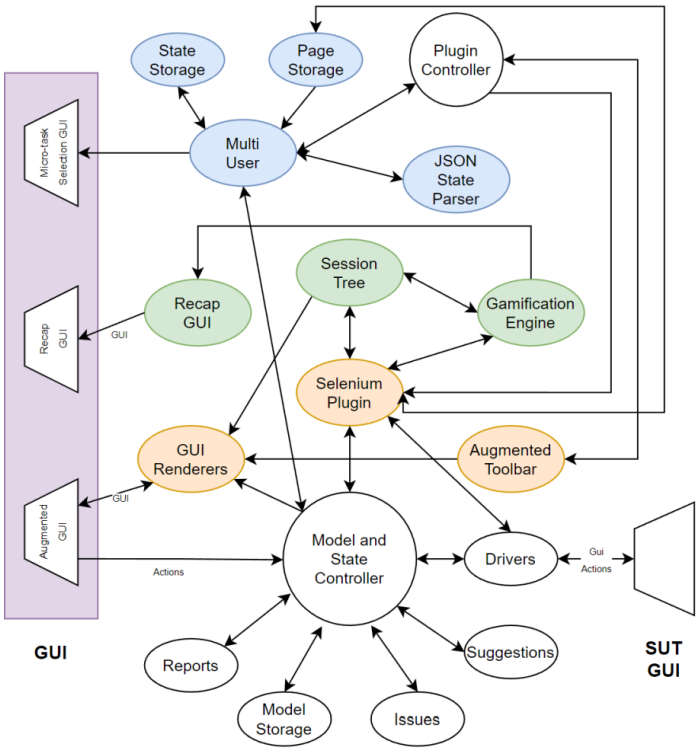

Fig. 1. The Scout tool with the Gamification and Crowdsourcing plugins

The objective of this work has been to create a prototype framework for the application of Gamification and Crowdsourcing for GUI-testing of web applications.

To that extent, we have selected the Scout tool, developed by Nass et al. [14], implementing an Augmented Testing methodology for web applications. The tool has been selected because of its high modularity and the possibility of adding new features through the development of plugins.

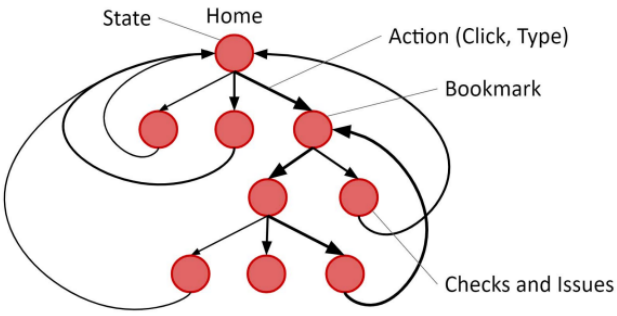

Fig. 2. The Scout state model

We have built upon a previous study by Cacciotto et al. [4], which introduced Gamification elements in the tool. A high-level depiction of the architecture is reported in Fig. 1. In the figure, the base modules of the Scout tool are reported in white. Among them, the state model is in charge of describing all the states of the SUT that were traversed during a user's testing sessions. The model is implemented as a weighted graph (as in Fig. 2) in which each node represents an application state, and the arcs are the tester's actions determining a state change. The *Augmented GUI* is the only interface the tester is using and acts as a proxy between the tester and the SUT. The Augmented GUI enriches the GUI of the SUT with suggestions and results of checks on the widgets and acts as a proxy to collect all the testers' inputs.

The components for the Gamification module are highlighted in orange (the plugins already present in Scout that were enhanced) and in green (those introduced ex-novo for gamification aspects). The Selenium plugin is in charge of obtaining information about the widget hierarchy of the current screen from the DOM model of the web application. The Gamification Engine is in charge of assigning a score to each tester according to the exploration activity, the reached coverage in the test session, and other parameters like the average speed for the execution of interactions (the interested reader can refer to the original work by Cacciotto et al. for details on the score computation [4]). The Session Tree component generates a simplified exploration graph that does not create a new state for each interaction with the GUI but instead performs a 1-to-1 mapping between pages of the web applications and states in the graph.

The components in blue in the diagram are those composing the Crowdsourcing plugin. The main components of the Crowdsourcing plugin are the Multi-User module and the JSON State Parser, which are in charge of loading the shared state, creating the micro-tasks, and resolving conflicts in merging the activities performed by different users. The ways these operations are performed are described in detail in the following subsections.

3.1 State Representation

For the purpose of this study, we adopted a state representation which is a simplified version of the one used by the original Scout tool. In our crowdsourced plugin, we use a state representation where two states are considered coincident if they are web pages with the same URL. This state representation is a limitation for Single Page Applications (SPA) which would require a finer definition of the states to allow the creation of multiple states during the navigation on the same page.

All the testers that are performing activities on the same page share a state model, with information on the widgets that have been interacted with to transition from one state to the other. Each widget interaction and page keeps information about the tester who performed the first interaction with or within it, who modified it or cancelled it from the test sequence.

The model of the collaborative testing session is represented in a human-readable JSON file. An example of a JSON file for web application exploration is reported as an online appendix[1].

3.2 Integration of sessions by multiple users

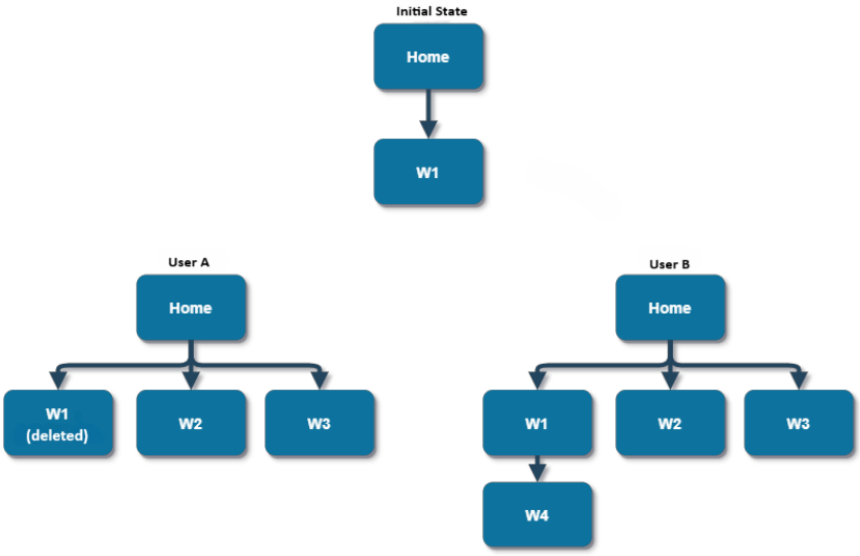

Fig. 3. A collaborative session scenario

[1] https://doi.org/10.6084/m9.figshare.25705425

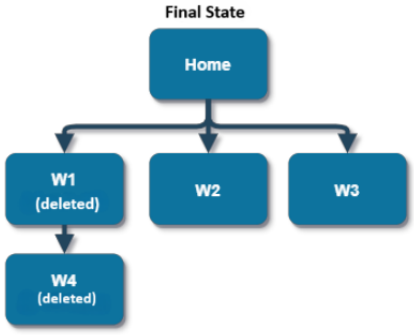

Fig. 4. Result of the collaborative session scenario

Table 1. Merge strategy depending on operations performed by users

User A	User B	Integration strategy
Add	Add	If the two additions are on the same widgets, a single status change is added to the exploration tree. Otherwise, both states are added to the exploration tree.
Add	Modify	Both actions are added to the state tree. The add creates a transition to a new state, and the modification is reported in the state tree.
Add	Delete	The delete is privileged. The state and all the other ones in the exploration tree are marked as cancelled.
Delete	Modify	The delete is privileged. The state and all the other ones in the exploration tree are marked as cancelled.
Modify	Modify	The most recent modifications by the users are kept.

One of the main requirements for a crowdsourced platform is the integration of the efforts performed by multiple users. In our context, each testing session contains interactions with widgets, that are stored with the information about the new widget status. When dealing with their test sequences, we can identify three different operations made by the users:

- *Addition* of an interaction with a widget, with the consequent modification of the state, or transition to a new state (i.e., page) in the web application;
- *Modification* of the interaction performed with a widget;
- *Deletion* of the interaction with a widget. This operation deletes all the next states that were initially explored in the test sequence.

When multiple users use the Scout tool to interact with the same application, it is necessary to perform a merge of the interactions that were performed. In case of conflicts, the rules are described in table 1. This set of rules has been defined in the conceptualization of multi-user testing performed by Bauer and Alégroth [3].

As an example of how the merge is implemented, we report two example test sequences performed by two users (Fig. 3). The users have performed, starting from the application's home page, some interactions with the widgets (w1, w2, and w3 for convenience) leading to separate states of the application. At the end of the session user A marks the interaction with widget w1 as eliminated. B performs the same interactions with widgets w1, w2 and w3, and performs an additional interaction with widget w4 in the state after w1. The merge of the explorations performed by the two users is reported in Fig. 4. With the adopted strategy, the merged exploration contains both the operations on w2 and w3 with no conflicts and both the interactions with w1 and w4 will be considered as cancelled.

The developed prototype currently handles updates to the state flow graph by synchronizing with the master graph stored in the server only at the end of each session. We acknowledge that based on the size of the difference between the resulting graph and the source graph, different update approaches may be needed instead (e.g., synchronization at the creation of each graph node, mutual synchronization between concurrent processes, etc.).

3.3 Generation of Microtasks

In addition to the possibility of merging multiple test sequences performed by users, the tool creates *micro tasks* for the users to perform in their testing session. A microtask is an atomic activity that has to be performed by the user, and it is a common decomposition used in crowdsourced processes. By definition, a microtask can be performed with limited resources and require little time to be completed. The users completing a microtask receive immediate feedback about their job. This immediate feedback can lead to an increased motivation for the user, who can immediately understand the purpose of his actions.

In the case of web application tasting, we identify a micro task as any step in a full use case/user scenario with the application. In the context of our prototype

tool, we defined micro tasks as requests for the users to increase the widget coverage (i.e., maximize the number of interactions) within a single web page of the software under test. Before the execution of every test session, the tool selects a set of web pages that still have a low widget coverage and creates a list of micro tasks that can be selected by the user (see Fig. **??**). Each microtask asks the user to explore the application in proximity to a given state (or page) of the application, instead of traversing the application in more depth. Therefore, this type of microtask privileges an in-depth exploration of the pages instead of increasing the breadth of the generated test sequences.

3.4 Evaluation of the gamification aspects of the prototype

To evaluate the gamification aspects of the prototype, we use the Octalysis gamification framework theorized by Yu-Kai Chou [18].

The starting gamified tool on the basis of which the crowdsourcing logic was added and the rationale behind its design and development is left to the interested reader who can find it in [6] [9]. The main gamification aspects covered by the tool are the following:

- **Accomplishment:** game elements concerning progress, skill-building, and overcoming challenges. This Core Drive often focuses on points, badges, and leaderboards.
- **Empowerment:** game elements enabling the user's need to express their creativity, see results, receive feedback, and respond.
- **Social Influence:** this aspect includes social elements such as mentorship, acceptance, companionship, competition, and envy. It also includes the drive to draw closer to relatable people, places, or events.
- **Scarcity:** game elements related to poor goods availability or appointment dynamics creating the need of wanting something because you can't have it right away: this motivates people to think about it all day long.
- **Epic Meaning:** game elements aiming to make players feel they're doing something greater than themselves or to be "chosen" to do something for a greater entity.

A graphical evaluation of the gamification elements introduced is reported in the Octalysis graph in Fig 5.

4 Preliminary Evaluation

To evaluate the benefits introduced by crowdsourcing to the web application testing activities conducted with the Scout tool, we set up a small preliminary experimental evaluation with a sample of testers. The phases of the preliminary evaluation – graphically described in Fig. 6 – are the following:

1. Introduction to the Scout tool and tool demo, followed by an introduction to the Gamification aspects that are embedded in the tool;

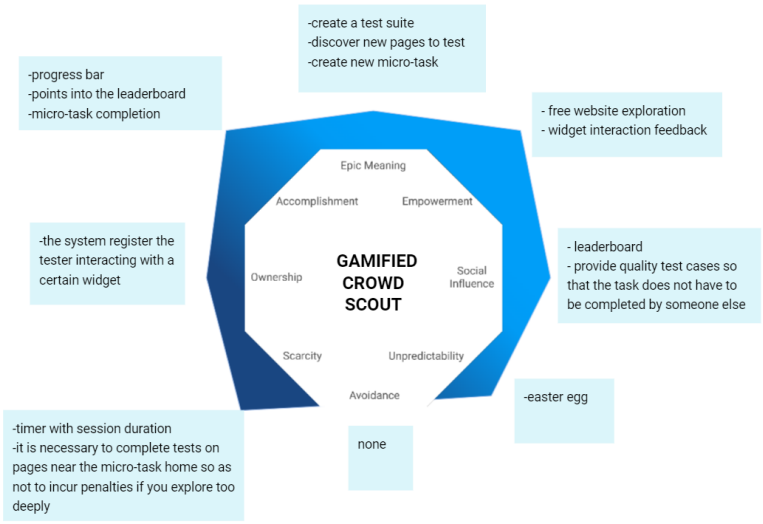

Fig. 5. Octalysis Evaluation for our gamified and crowdsourced tool

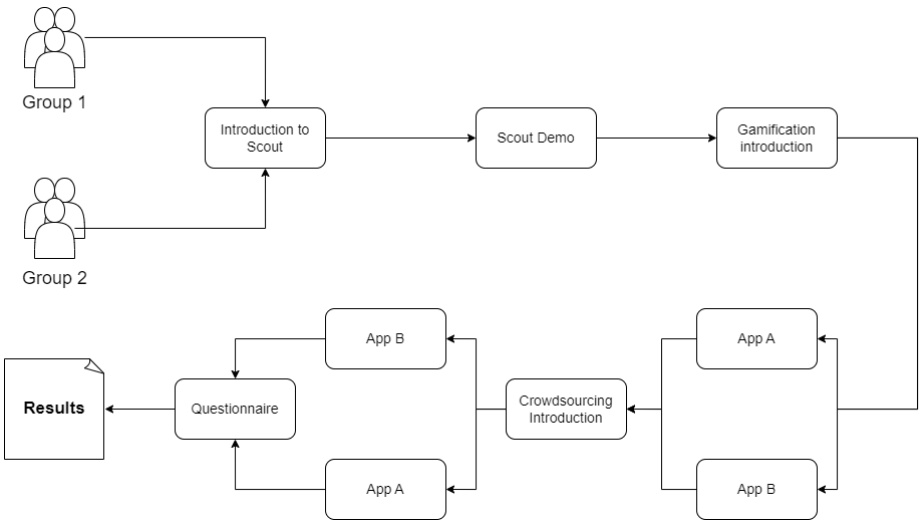

Fig. 6. Phases of the evaluation process

2. Creation of test suites with the Gamified version of the tool;
3. Introduction to the Crowdsourcing aspects that are embedded in the tool;
4. Creation of test suites with the Crowdsourced versions of the tool;
5. Questionnaire about the user experience provided by the tool.

For the preliminary evaluation, we recruited a sample of six testers from among students in the Software Engineering course. All the testers conducted Master's studies in Computer Engineering and had previous experience with programming languages. Four respondents (66%) had previous experience with web application testing. Only two respondents (33%) had previous experience with gamified platforms for software development; no respondent had previous experience with crowdsourced platforms for software development.

The groups executed test sequences on two different websites: a marketplace web application (application A)[2], and an institutional university website (application B)[3]).

Both the applications were provided with an initial test session, to simulate the existence of collaborators that already performed testing operations before the current participant in the experiment.

As can be seen in the experiment design, the respondents were divided into two groups in which the applications were tested in different orders, to reduce any learning effect of the application on the result.

All the experimental sessions were executed on the same device to avoid configuration and timing issues related to the use of the respondents' machine.

4.1 Exploration and Coverage

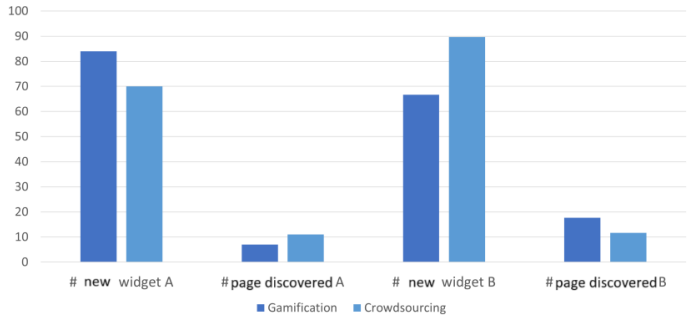

Fig. 7. Results for new widgets and new pages interacted

The first dimension that we inspected was the test exploration and coverage of the application under test that was obtained in the executed test sessions.

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

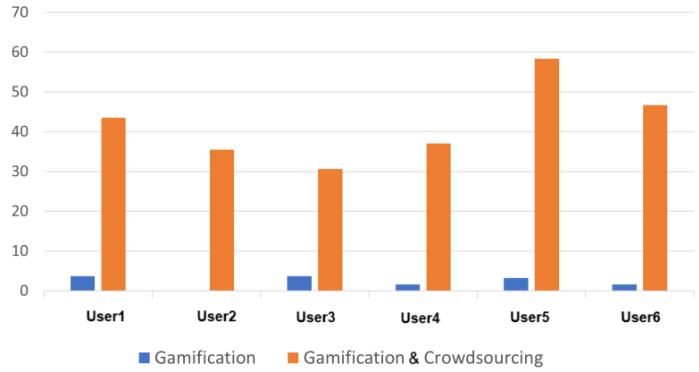

Fig. 8. Increase in local widget coverage with active crowdsourcing plugin

In Fig. 7 we report the comparison of the total number of new widgets and new pages found by the six test suites with gamification or with crowdsourcing. We notice a decrease in the number of widgets discovered for site A (-15%) and an increase in the number of widgets discovered for site B (+32%). The users discovered more pages for site A (+57%) and less for site B (-30%). Based on these figures, we can argue that the number of widgets and page coverage obtained does not depend on the type of tool used but is instead dependent on the specific software under test.

In Fig. 8 we report the increase in *Local Coverage* that was obtained by each user in the two sessions of the experiment. The concept of local coverage was defined for the purpose of this experiment as *the average widget coverage on the specific page that acts as the starting point for the test case*. We can notice that in all the test sessions that were conducted without the crowdsourcing tool, there was little to no increase in local coverage. On the other hand, there was a significant increase in the local coverage obtained with the tool implementing crowdsourcing mechanisms. This result suggests that the use of micro-tasks made the users very focused on obtaining the highest possible coverage on the web page where the test session was starting, therefore privileging a depth-first rather than a breadth-first (i.e., more explorative of other undiscovered web pages) approach to testing.

4.2 User Experience

At the end of the experimental sessions, we administered a short questionnaire with five-level Likert questions to the respondents, reported in table 2.

In Fig. 9 it can be seen the distribution of the answers to the questionnaire. The distribution suggests an overall appreciation of the tool between all the respondents. Mixed answers were received for question 6, with a disagreement among the respondents. This aspect is worth investigating in future research

Table 2. Post-experiment questionnaire

ID	Question	Type
q1	Have you understood all the features of Scout and how to use it?	Likert
q2	Have you understood the microtask concept and its finality?	Likert
q3	Did you find the screen for microtask selection comprehensible?	Likert
q4	Do you find the addition of micro tasks useful?	Likert
q5	Knowing that there is a bonus in the score if you are focusing on the microtask made you focus more on the starting page?	Likert
q6	Knowing that your work will be continued by other people made you more motivated?	Likert
q7	Do you prefer the version with crowdsourcing rather than the standard version of the gamified tool?	Likert
q8	Do you think that the tool could be easily integrated into an existing test team in a real-world scenario?	Likert
q9	What are in your opinion the main drawbacks of the prototype?	Open question
q10	Do you have suggestions to improve the prototype?	Open question

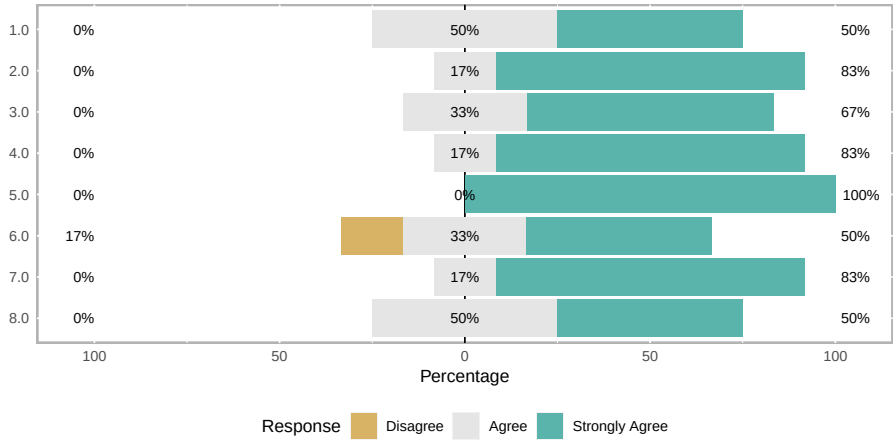

Fig. 9. Distribution of the answer to q1-q8

efforts since the motivation induced by the collaboration is the foundation for any framework implementing explicit crowdsourcing.

The open-ended questions of the questionnaire received mostly technical responses. Regarding question q9, three respondents reported no problems. Two respondents identified slow page loading as a significant issue, leading to confusion about whether Scout was loading a new page or if they could already begin interacting with the one displayed. Finally, a respondent indicated issues in the visualization of the gamified mechanics.

The new plugin places great emphasis on testing with high coverage on pages near the root of the microtask. One participant suggested adding a pop-up notification when a page reaches 100% coverage and including a numerical indication of the number of tested widgets compared to the total, as they did not find the already present progress bar sufficient. During testing with the multi-user version, all participants frequently interacted with the Augmented Toolbar to use the "Go Micro Task Home" button and increase local coverage on the starting page. A common proposal by respondents is to add keyboard shortcuts to expedite this action and eliminate the time spent interacting with the toolbar.

5 Conclusion and future work

In this paper, we described a prototype tool to integrate collaborative and crowdsourcing-based aspects into a Gamified web application GUI testing tool. We implemented a proof-of-concept of the tool as an extension of an existing Augmented testing tool, Scout, and we performed a preliminary evaluation with graduated Master's students, that highlighted the capability of microtask-based collaborative gamification of increasing the focus of the testers on performing depth-first test sequence creation.

Our immediate future work includes an empirical evaluation of the framework with a statistically significant sample of respondents and a focus on the quality of the generated test cases. Another extension of the present work is the creation of microtasks based on use cases (or portions of them) in order to incentivise the conduction of test sequences near the main functionalities of the app instead of focusing on a single screen at a time.

Acknowledgement

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

References

1. Alégroth, E., Gao, Z., Oliveira, R., Memon, A.: 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). pp. 1–10. IEEE (2015)
2. Alhammad, M.M., Moreno, A.M.: Gamification in software engineering education: A systematic mapping. Journal of Systems and Software **141**, 131–150 (2018)
3. Bauer, A., Alégroth, E.: We tried and failed: An experience report on a collaborative workflow for gui-based testing. In: 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 1–9. IEEE (2023)
4. Cacciotto, F., Fulcini, T., Coppola, R., Ardito, L.: 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). pp. 126–129. IEEE (2021)
5. Chen, Y., Pandey, M., Song, J.Y., Lasecki, W.S., Oney, S.: Improving crowd-supported gui testing with structural guidance. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. pp. 1–13 (2020)
6. Coppola, R., Ardito, L., Fulcini, T., Garaccione, G., Torchiano, M., Morisio, M.: A Framework for the Gamification of GUI Testing, pp. 215–242. Springer Nature Switzerland, Cham (2023)
7. Coppola, R., Fulcini, T., Ardito, L., Torchiano, M., Alègroth, E.: On effectiveness and efficiency of gamified exploratory gui testing. IEEE Transactions on Software Engineering (2023)
8. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining" gamification". In: Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments. pp. 9–15 (2011)
9. Fulcini, T., Ardito, L.: Gamified exploratory gui testing of web applications: a preliminary evaluation. In: 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 215–222 (2022). https://doi.org/10.1109/ICSTW55395.2022.00045
10. Fulcini, T., Coppola, R., Ardito, L., Torchiano, M.: A review on tools, mechanics, benefits, and challenges of gamified software testing. ACM Computing Surveys **55**(14s), 1–37 (2023)
11. Hamari, J., Koivisto, J.: Social motivations to use gamification: An empirical study of gamifying exercise. In: ECIS. vol. 105, pp. 18–19 (2013)
12. Linares-Vásquez, M., Moran, K., Poshyvanyk, D.: Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 399–410. IEEE (2017)
13. Mao, K., Harman, M., Jia, Y.: Crowd intelligence enhances automated mobile testing. In: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 16–26. IEEE (2017)
14. Nass, M., Alégroth, E., Feldt, R.: Augmented testing: Industry feedback to shape a new testing technology. In: 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 176–183. IEEE (2019)
15. Parizi, R.M.: On the gamification of human-centric traceability tasks in software testing and coding. In: 2016 IEEE 14th International Conference on Software En-

gineering Research, Management and Applications (SERA). pp. 193–200. IEEE (2016)
16. Rojas, J.M., Fraser, G.: Code defenders: a mutation testing game. In: 2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). pp. 162–167. IEEE (2016)
17. Straubinger, P., Fraser, G.: A Survey on What Developers Think About Testing. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). pp. 80–90. IEEE (2023), https://ieeexplore.ieee.org/abstract/document/10301252/
18. Teixes, F.: Yu-kai chou (2016). actionable gamification: beyond points, badges and leaderboards. octalysis media: Fremont. ca. RIO: Revista Internacional de Organizaciones (18), 137–144 (2017)
19. Wang, J., Li, M., Wang, S., Menzies, T., Wang, Q.: Images don't lie: Duplicate crowdtesting reports detection with screenshot information. Information and Software Technology **110**, 139–155 (2019)