POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Review on Tools, Mechanics, Benefits, and Challenges of Gamified Software Testing

Original
A Review on Tools, Mechanics, Benefits, and Challenges of Gamified Software Testing / Fulcini, T., Coppola, R., Ardito, L., Torchiano, M.. - In: ACM COMPUTING SURVEYS. - ISSN 0360-0300. - 55:14s(2023), pp. 1-37. [10.1145/3582273]

Availability:
This version is available at: 11583/2976037 since: 2023-07-20T13:36:23Z

Publisher:
ACM

Published
DOI:10.1145/3582273

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript, con Copyr. autore

© Fulcini, Tommaso; Coppola, Riccardo; Ardito, Luca; Torchiano, Marco 2023. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in ACM COMPUTING SURVEYS, http://dx.doi.org/10.1145/3582273.

(Article begins on next page)

26 November 2026



A Review on Tools, Mechanics, Benefits, and Challenges of Gamified Software Testing

TOMMASO FULCINI, RICCARDO COPPOLA, LUCA ARDITO, and MARCO TORCHIANO, Politecnico di Torino, Italy

Gamification is an established practice in Software Engineering to increase effectiveness and engagement in many practices. This manuscript provides a characterisation of the application of gamification to the Software Testing area. Such practice in fact reportedly suffers from low engagement by both personnel in industrial contexts and learners in educational contexts. Our goal is to identify the application areas and utilised gamified techniques and mechanics, the provided benefits and drawbacks, as well as the open challenges in the field. To this purpose, we conducted a Multivocal Literature Review to identify white and grey literature sources addressing gamified software testing.

We analysed 73 contributions and summarised the most common gamified mechanics, concepts, tools and domains where they are mostly applied. We conclude that gamification in software testing is mostly applied to the test creation phase with simple white-box unit or mutation testing tools, and is mostly used to foster good behaviours by promoting the testers' accomplishment. Key research areas and main challenges in the field are: careful design of tailored gamified mechanics for specific testing techniques; the need for technological improvements to enable crowdsourcing, cooperation, and concurrency; the necessity for empirical and large-scale evaluation of the benefits delivered by gamification mechanics.

CCS Concepts: • **Software and its engineering → Software verification and validation**;

Additional Key Words and Phrases: Software/Program Verification, Testing and Debugging Gamification, Software Testing, Software Engineering, Systematic Literature Review, Multivocal Literature Review

1 INTRODUCTION

The objective of software testing is ensuring the quality and reliability of software; it is therefore a crucial activity, especially in modern software development processes, where high-complexity software is released at a very fast pace. However, testing activities are often overlooked even by large companies and in important software projects since they are frequently considered unappealing, time-consuming, and repetitive when compared to more creative and fulfilling activities such as software design or coding [15]. Testing activities are generally classified into two main categories, based on the way they are performed: (i) automated, when testers create scripts – either coding or via the recording of user's interactions – which can then be executed by automated test runners; or (ii) manual, when testers execute manually the test sequences against the SUT (System Under Test). Automated testing is characterised by lower execution time and lower costs due to possible reuse of the scripts; conversely, the main issues concern the test suite maintenance and the effort still required to create test scripts by the tester. Manual testing is associated with higher costs, due to the time spent by the tester in the execution of test cases, though it requires simpler setup activities and guarantees higher flexibility, since testers can adapt the test case to the scenario presented by the SUT on the fly.

Authors' address: Tommaso Fulcini, tommaso.fulcini@polito.it; Riccardo Coppola, riccardo.coppola@polito.it; Luca Ardito, luca.ardito@polito.it; Marco Torchiano, marco.torchiano@polito.it, Politecnico di Torino, Corso Duca degli Abruzzi 24, Turin, Piedmont, Italy, 10129.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

0360-0300/2023/2-ART $15.00
https://doi.org/10.1145/3582273

Just Accepted

Over the years the market has pushed towards automated testing approaches, but the motivational gap is common to such techniques and the manual alternatives. Motivation and job satisfaction in software testing – and more generally in software engineering – is an important aspect which has been largely addressed in the recent years [46]. A lower engagement is also considered a cause for reduced effectiveness and efficiency of testing practices [53]. This aspect is particularly crucial; in fact surveys with software testing practitioners underline that test engineers are more interested in options to improve effectiveness and efficiency than in theoretically challenging issues [23]. At the same time, it is possible to trace back the lower interest in testing activities to a subordinate role of testing in education settings when compared to, e.g., coding [49].

A recently explored solution to increase engagement and motivation of software testers is *Gamification*, i.e., the addition of game mechanics to practices that are not ludic by themselves. The main goals of Gamification are to increase the productivity of the involved actors in tasks of any nature, by stimulating positive feelings through the incorporation of elements that are typical of game contexts in the regular activities, i.e. by creating a *gameful* or *gamified* experience. Examples of game elements that are typically utilized in gamified experiences are scoring and leaderboard mechanisms, prizes and achievements, storytelling and *levels*. Creating a gamified experience that properly suits the context is not trivial: game elements cannot simply be considered separately, since what actually affects the user is the result of their interaction [44]. For this reason some frameworks emerged to define the desired gameful experience and to assess it. Different roles and principles should be considered to systematically define and model the expected outcome of Gamification, as defined by Robson et al. [44].

Gamification, as time passes, has become more and more popular in different domains, attracting the interest of the scientific community and reaching a capillary diffusion in all the scientific disciplines over the past few years, according to O'Donnell et al. [40]: it has been therefore identified as a multidisciplinary field with a high research potential [38]. Two of the most common fields of application are Education and Computer Science, where gamification found a fertile ground given the benefits that several secondary studies identify [9] [37]. Gamification also has a history of success in Social Sciences and Pedagogy, as testified by a growing body of theory development and empirical research [34].

Gamification was leveraged also in Software Engineering since the beginning of the last decade. Albeit the research is very preliminary and – still – frequently lacks proper methodological support, the integration of gamification in SE-related activities is seen as an important challenge for both researchers and practitioners [42]. Among all SE sub-disciplines, recent academic and industrial efforts have tried to utilise Gamification to render testing activities more engaging and appealing and, as a consequence, more effective [51]. Even though several frameworks and tools have been described in the literature, no recent source has provided a comprehensive and up-to-date review of all the proposed gamification mechanics and their benefits, drawbacks, and challenges.

To bridge this gap, we performed a Multivocal Literature Review (MLR) study that covers both peer-reviewed works, also called white literature (WL), and grey literature (GL), accessible through traditional search engines. Through the latter category, we aimed at capturing valuable information provided by practitioners from the industry. We frame the current state of the art and practice in the field of gamified software testing and provide a characterisation based on the testing levels, methodologies, and domains covered, and on the types of game and mechanics provided. The present work is meant to share information regarding the current state of the art and practice, not only highlighting main trends but also outlining directions for future investigators approaching this particular domain by framing specific gaps to address.

The remainder of this manuscript is structured as follows:

- Section 2 presents background information about Gamification and Multivocal Literature Reviews; it defines the terminology used for software testing throughout the paper, and compares this work to existing secondary studies in the field;

Just Accepted

- Section 3 describes the adopted research methododology by specifying its goals, review questions, search criteria, and analysis methods;
- Section 4 reports in detail all the results collected from the analysed literature;
- Section 5 frames the results and discusses their implications, along with possible threats to the validity of this study;
- Finally, Section 6 concludes the study by providing guidelines for different actors involved in gamified software testing and introduces future prosecutions of this work.

2 BACKGROUND

2.1 Gamification

Gamification has been defined by Deterding et al. as *"the use of game design elements in non-game contexts"* [16]. This newly trending technique has been widely used in several areas such as learning, business, marketing, tourism and also computer science; in particular, for the latter, a large increase in the usage was reported in software engineering disciplines according to Barreto and França [4]: Gamification-based approaches have some important advantages from the psychological user-experience perspectives in non-ludic activities, such as increased motivation, focus and engagement, but also better performance and higher efficiency.

Several different categorisations of gaming elements in gamified approaches have been proposed in the literature. Robson et al.[44] provide a taxonomy of game elements, which they categorise under game *mechanics*, *dynamics* and *emotions*. According to Robson, game mechanics are related to the goals, rules and interactions provided by a gamified systems. Game dynamics are instead related to the interaction between the players and the system during the gamified activities. Finally, game emotions refer to the emotional outcome evoked among individual players when participating in a gamified experience. This classification identifies mechanics as common components of the gamified system for all the different players, which are independent of the user experience (e.g., the presence of a chat system), and dynamics as variable and user-dependent characteristics of the gamified activity (e.g., competition or cooperation between different players). From our preliminary analysis of related secondary studies in the field, however, we noticed that – in most cases – the proposed separation between mechanics and dynamics is not applied. Instead, the two terms are instead used mostly as synonyms to describe design characteristics of gamified tools or environments. Therefore, in the present manuscript, we will only refer to game *mechanics* to describe all possible design choices that contribute to build a gamified environment.

An essential factor in building a successful gamified environment is a careful design of the implemented game mechanics. Creating valuable gamified activities that suit the business needs and the end users' expectations is a complex task that needs a structured and well-rooted approach to be effective. To that purpose, several approaches have been proposed to provide a systematic ground for the application of gamified constructs to any activities. Among them, one of the most frequently adopted is the Octalysis framework, originally proposed by Yu-kai Chou [10]. The Octalisys framework can be used to implement *human-focused design* for a gamified tool: it identifies eight core drives representing human aspects that can be stimulated by gamification. The core drivers are then decomposed into multiple atomic finer-grained game mechanics. The framework can be used as a design tool, but also as an evaluation tool for existing instruments with gamified elements.

The eight core drives are paired with a two-dimensional, higher-level representation: *left-brain* vs *right-brain* drivers, and *white-hat* vs *black-hat* drivers. The former classification distinguishes intrinsic-motivator elements related to creativity, emotion, self-expression and social aspects (right brain elements, e.g., socialising with other people) and extrinsic motivator elements associated with logic, calculation and ownership (left brain elements, e.g., goals to accomplish). The latter instead distinguishes elements into positive motivators (white hat, e.g., a narrative built to make the user feel successful) and negative motivators (black hat, e.g., assets available only for

Just Accepted

Table 1. The Octalysis core drives with a brief definition from the official website

Core Drive	Description
Epic Meaning & Calling	*"A player believes that he is doing something greater than himself or he was "chosen" to do something"*
Development & Accomplishment	*"Making progress, developing skills, and eventually overcoming challenges"*
Empowerment of Creativity & Feedback	*"When users are engaged in a creative process where they have to figure things out and try different combinations repeatedly. People [...] need to be able to see the results of their creativity, receive feedback, and respond in turn."*
Ownership & Possession	*"Users are motivated because they feel like they own something. Players [...] innately wants to make what she owns better and own even more."*
Social Influence & Relatedness	*"This drive incorporates all the social elements that drive people, including mentorship, acceptance, social responses, companionship, as well as competition and envy"*
Scarcity & Impatience	*"This is the drive of wanting something because you can't have it."*
Unpredictability & Curiosity	*"This is a harmless drive of wanting to find out what will happen next. If you don't know what's going to happen, your brain is engaged, and you think about it often"*
Loss & Avoidance	*"This core drive is based upon the avoidance of something negative happening."*

a limited amount of time). A summary of the eight core drives defined by the Octalysis framework is presented in Table 1.

Another framework to evaluate gamified experiences and the emotional effects of their utilisation is GAMEX, defined by Eppmann et al. [19]. GAMEX identifies six dimensions that represent the different aspects of a gamified experience: enjoyment, absorption, creative thinking, activation, absence of negative affect and dominance. A total of 27 Likert questions are used to measure how the mentioned aspects influence the perceived gamified experience[1].

2.2 Software Testing

Software Testing is the process of evaluating software to ensure that it meets its originally specified requirements and revealing faults and defects that may affect the code. Its importance is considered critical in industrial software development. Evidence in the literature suggests that the cost for software testing activities can, in several cases, amount to up to 50% of the total costs of software development, according to some studies [28]. During the latest years, several methodologies have been defined to perform software testing activities in different software domains. .

Software testing can be classified based on different characteristics. It is out of the scope of this manuscript to provide a characterization of all dimensions to describe software testing. We will however consider three different ways to characterize software testing activities: test *levels*, *phases*, and *methodologies*.

Test *levels* refer to the size of the components of the Software Under Test (SUT) that are tested. Test levels range from the testing of individual atomic code units (*Unit Testing*) to the integration test of multiple units (*Integration Testing*), to *System Testing* of the whole SUT, with which the tester interacts as a final user would do. A widely used representation of the testing levels is Cohn's testing pyramid [13].

In addition to the testing level, a second categorization can be provided for testing practices, according to the *methodology* used to generate test cases. Unit and integration testing are mostly performed by utilising *scripted* techniques, in which the tester defines test scripts that can be run against the SUT to exercise its functionalities [18]. At the System Testing level, several methodologies are utilised to generate test scripts. Concerning test methodology, we adopt a classification provided by Linares-Vazquez et al. [35]:

- Manual testing involves testers that manually execute the defined test cases against the finished SUT. Manual testing can be performed according to different strategies, as reported by Itkonen et al. [30]: *Exploratory Testing* is a manual testing activity where the interactions with the SUT are performed in an unstructured way, by proceeding from feature to feature to cover the GUI features; *Documentation based*

[1]Likert questions are a common measurement in user experience evaluation, consisting of declarative statements, with response sets consisting of equally spaced numbers (typically 5 or 7) representing the agreement of the subject with the statement [27].

Just Accepted

session strategies instead rely on executing manually the instructions reported in test documents (e.g., test-cases, release notes, defect reports). Manual testing activities can also be differentiated according to how the checks are executed, i.e. by *Comparison* (visual comparison of the state of the SUT with a known working version) or by *Input/Output* verification;
- Automation APIs/frameworks rely on the manual creation of test scripts that exercise the GUI of the SUT. These scripts typically specify a series of actions that should exercise either simple units of functions of the SUT (e.g., as with the JUnit test runner) or even the whole system through its GUI (e.g., the Selenium or Appium GUI automation frameworks [8];
- Capture & Replay techniques involve the manual execution of test sequences that are then used to generate repeatable test scripts [39];
- Automated Test Generation Techniques generate a model of the SUT and then automatically exercise it by generating sequences of inputs [17]. The simplest example of automated test generation is represented by random-based input generation, which selects random components or features of the SUT to exercise. Advanced tools generate (automatically or with the manual aid of the tester) models to be traversed systematically. These models of the SUT can be in the form of Finite-State-Machines (FSM) [52], Event-Sequence-Graphs (ESG) [5], etc..

Finally, several frameworks in the literature organize software testing activities in separate *phases*. In this manuscript, we adapt the phases described in the Software Testing Life Cycle [29]. Additional details about the levels of the selected dimensions are reported in the following methodological subsections.

Especially at higher levels in the testing pyramid, testing activities are often perceived as time-consuming, error-prone, brittle and costly. Evidence in the literature suggests that these activities are overlooked and neglected even in large software projects, especially for what concerns automation of system-level testing: Berner et al. report a set of case studies where the missing design for testability and the difficulty in maintaining testware leads to inappropriate test automation strategies and to the eventual prevalence of manual testing activities [6]. All these issues are mainly due to the perception that testers have of the activity they are performing. Being the benefits of gamification well-known from studies in other disciplines, many works in the related literature have thus identified software testing as a possible area for a fruitful application of gamification mechanics [14], with some identifying it as a promising but still underrepresented area [42].

A practical example of a gamified environment for software testing is Code Defenders, the tool proposed by Fraser et al. [WL05] where the gamified activity is mutation testing. Testers play the roles of defenders and attackers: the former have to enrich an existing test suite with new tests, predicting any possible mutations generated by attackers and detecting with their test code, and the latter have to inject code to break defenders' test suite making their tests fail. Attackers earn points if they manage to break the defenders' test suite, while defenders score if the built test suite manages to pass all the tests.

In the mentioned gamified environment, the adopted game aspects are the competition between the two teams, the scoring system, and a leaderboard showing testers with the highest score. These elements have been proved by the authors of the tool as positively engaging the testers in their mutation testing activities, encouraging them to test the SUT more thoroughly. Thus, gamification of software testing activities can be seen as an incentive for software testers to perform more thorough and systematic testing of the SUTs. In section 2.4 (Related Work) we discuss more thoroughly the existing findings in secondary studies about gamified Software Engineering and Software Testing.

2.3 Multivocal Literature Reviews

A Multivocal Literature Review (MLR) differs from a Systematic Literature Review (SLR) in that it includes the Grey Literature (GL) in addition to the White Literature [41]. Grey Literature is defined as *what is produced on all*

Just Accepted

levels of government, academics, business and industry in print and electronic formats, but which is not controlled by commercial publishers, i.e., where publishing is not the primary activity of the producing body [47]. Adams et al. classify Grey Literature into three different categories: 1st tier (or high credibility), which includes books, magazines, government reports, and white papers; 2nd tier (or moderate credibility), including annual reports, news articles, presentations, videos, question and answers websites; 3rd tier (or low credibility), including blogs, evidence from e-mails, posts on social networks [3].

A formalisation of the MLR methodology for SE has been provided only recently by Garousi et al. [24]. The authors base their guidelines on well-established methodological guidelines to conduct traditional Systematic Literature Review, while stressing the benefits provided by having an overview on both the state of practice and academic state of the art. The combination of the two points of view, in fact, allows to analyse and understand emerging trends coming from dual perspectives, therefore reducing the risks of neglecting aspects of the topic.

Rigorous MLRs have recently been conducted in the field of SE to investigate, for instance, the need for automation for software testing [26], and software test maturity assessment and test process improvement [22].

2.4 Related Secondary Studies

Many works are available in the literature discussing the application of gamified approaches to various aspects of the discipline of Software Engineering [21].

Compared to existing secondary studies related to the application of gamification to software testing [36] [14], we assessed an extended set of testing-related dimensions, considering testing levels, phases, methodologies (compatible with the mentioned studies), domains, and proposed tools (which were not analysed in previous literature). Another additional characterization, previously unexplored, focuses on the applied game elements and their classification according to the Octalysis core drivers. We also perform an analysis of future challenges, pros, and cons presented by the sources.

A complete discussion of related secondary studies is reported in online Appendix A.

3 RESEARCH METHOD

To conduct the MLR, we followed the guidelines for including grey literature in reviews for the software engineering discipline, provided by Garousi et al. [25]. These guidelines extend Kitchenham's guidelines for conducting Systematic Literature Reviews [33]. The procedure of conducting an MLR is divided into three distinct phases:

(1) Planning: in this phase, the need for conducting the MLR is established, and the goals and review questions of the MLR are specified;
(2) Conducting: the MLR is conducted by defining the search process, selecting the sources, assessing the quality of the sources, extracting and synthesising the collected data;
(3) Reporting: the review results are reported and tailored to the selected destination audience.

3.1 Planning

This section describes the sub-phases of the Planning phase: motivating the need for an SLR, defining the goals for the review, and formulating the Review Questions to answer.

3.1.1 Motivation behind conducting an MLR. To motivate the need for a literature review, we utilise the decision table proposed by Garousi et al. [25], based on the guidelines by Adams et al. [3]. According to these authors, one or more positive answers to the question in the decision table suggest the inclusion of Grey Literature in addition to White Literature in a review. The interested reader can find a detailed motivation for the positive answers to each question in the online Appendix B.

Just Accepted

Table 2. The review questions used for the data extraction

Goal 1: mapping of the contributions	
RQ1.1	What are the different categories of contributions of the considered sources?
RQ1.2	Which research methodology have been applied in the considered sources?
Goal 2: testing-focused characterisation	
RQ2.1	To which testing level is gamification applied?
RQ2.2	To which testing phase is gamification applied?
RQ2.3	What are the testing methodologies considered by the contribution?
RQ2.4	What are the existing testing frameworks or tools adopted by the papers?
RQ2.5	What is the language / domain of the testing tool that is gamified?
RQ2.6	Is gamification applied with a practical or educational focus?
Goal 3: Gamification-focused characterisation	
RQ3.1	Which are the gamification mechanics adopted for gamified software testing practice and education?
RQ3.2	Which are the tools available to perform gamified software testing practice and education?
RQ3.3	Which are the advantages of gamification and which are the empirical results, if any available?
RQ3.4	Which are the drawbacks of gamification and which are the empirical results, if any available?
RQ3.5	Which are the discussed challenges, open questions and focus areas for future research directions?

3.1.2 Goals and Review Questions. When defining the review questions, we identified three main goals for this review work:

- Goal 1: provide a mapping of the studies regarding the utilisation of gamified mechanics in the software testing discipline.
- Goal 2: identify contexts in the discipline of software testing to which gamification is applied, i.e., identify which levels, phases and testing methodologies are addressed by the collected literature, what are the application domains considered and which testing tools are leveraged and possibly extended.
- Goal 3: characterise the gamification mechanics, tools, and elements applied to testing and the provided advantages, drawbacks and open challenges in the field.

We formulated a set of Review Questions for each of the defined goals to analyse the three identified aspects in-depth. The Review Questions are reported in Table 2.

3.2 Conducting

In this section, we report the methodology employed in the Conducting phase and its sub-phases: selection of literature sources, formulation of the search strings, definition of the paper selection process, definition of the data extraction procedure. The process is synthesised in the diagram in Figure 1. The details of each step of the Literature Review are reported in Appendix C.

3.2.1 Search Approach. To conduct the review, we applied the following steps:

- *Application of the search strings*: the specific strings were applied to the selected online libraries (for white literature) and on the Google search engine (for grey literature);
- *Search bounding*: To stop the search for grey literature and to limit the number of sources to a reasonable number, we applied the *Effort Bounded* strategy, i.e., we limited our effort to the first 100 Google search hits as suggested by Garousi and Mäntylä [26]. On the tenth page (i.e. over the 100th entry) no relevant contributing source was found, confirming the hypothesis that relevant results usually appear only on the first pages. Therefore, having no reason to proceed further, the effort limit was set at 100;
- *Removal of duplicates*: in our pool of sources, we consider a single instance for each source that is present in multiple repositories;

Just Accepted

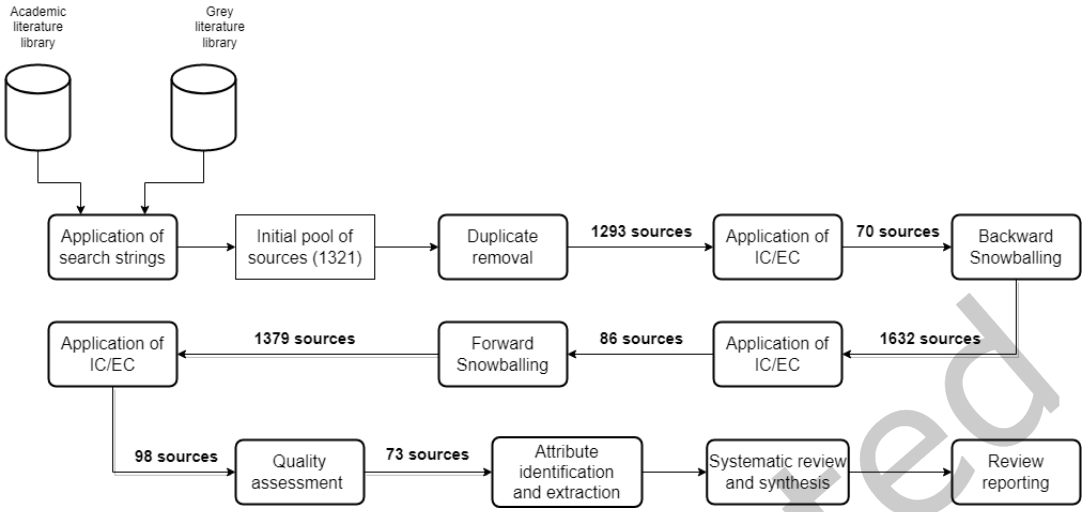

Fig. 1. Process of conduction of the literature review

Table 3. Number of papers after quality assessment of sources defining the final pool.

Repository	Search	IC/EC	Backward Snowballing	Forward Snowballing	Quality Assessment
IEEE Xplore	328	17	23	25	21
ACM Digital Library	151	8	14	17	12
Springer Link	396	3	3	3	2
Elsevier Science Direct	231	1	2	2	2
Google Scholar - WL	95	11	12	16	11
Google Search - WL	45	3	3	3	2
Google Scholar - GL	20	9	9	12	11
Google Search - GL	55	18	20	20	12
Total - WL	1246	43	57	66	50
Total - GL	75	27	29	32	23

- *Application of inclusion and exclusion criteria*: we defined and applied the inclusion and exclusion criteria directly to the sources extracted from the online repositories, based on an examination of titles, keywords, and abstracts of the papers;
- *Backward Snowballing* [31]: all the articles in the reference lists of all sources were added to the preliminary pool and evaluated through the application of the previous steps. We also added to the pool of grey literature the grey literature sources cited by white literature;
- *Forward Snowballing* [31]: we looked for articles in the online libraries and in the search engine that cited sources in the pool and, if not already present, we added them to the pool;
- *Quality assessment*: every source from the pool was entirely read and evaluated in terms of the quality of the contribution;
- *Documentation and analysis*: information about the final pool of paper was collected in a form including all data needed to answer the formulated review questions.

3.2.2 Final Pool of Sources. The papers that resulted from the search merging the different digital libraries were a total of 1221: 328 from IEEE Xplore, 151 from ACM Digital library, 396 from Springer Link, 231 from Science Direct and 115 from Google Scholar (of which, one item was repeated twice). The search for grey literature was limited to the first hundred results from Google.

Just Accepted

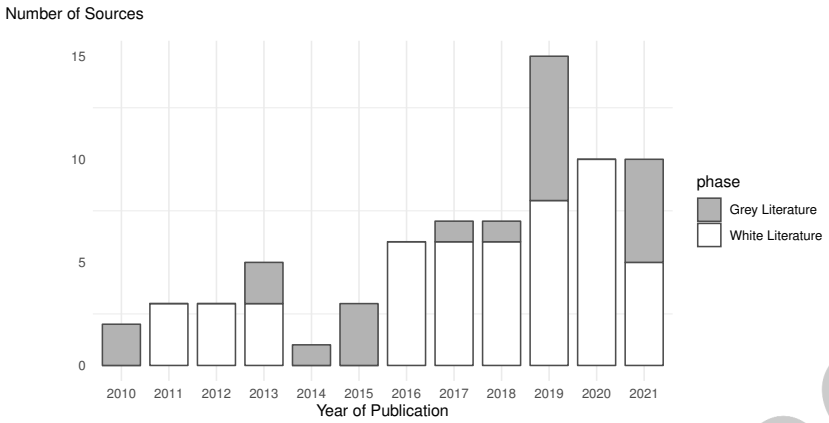

Fig. 2. Number of white and grey literature sources per year

The total number of collected items was 1321, counting both grey and white literature. After removing the duplicate, the papers remaining were 1293. After removing duplicates, inclusion and exclusion criteria were applied: this operation has shrunk the number of resources in the pool to 70 units, 43 items of white and 27 of grey literature.

The following step was the backward snowballing: from the selected papers, a total of 1562 sources came up, including duplicate papers both from the initial set and within the resulting pool. We applied duplicate removal and inclusion/exclusion criteria to this resulting group obtained through snowballing. This first snowballing process allowed us to add 16 results (14 white papers and 2 items of grey literature). Each title of the contribution found this way has been searched in the same starting digital libraries to assign the newly discovered publications in the correct repository, as shown in Table 3.

The same process was used for the forward snowballing, where the total number of retrieved studies was 1281. After the filter was applied, we included the remaining 12 items (9 pieces of white literature and 3 of grey literature). The resulting set of literature has been subjected to quality assessment.

After applying the stages described in the previous sections, our final pool included 73 sources. The full list of sources is reported as online additional material in Appendix D. Table 3 breaks down the number of sources that were present in the pool after each of the review stages. We report the information about all contributions in a publicly-available spreadsheet [2]. Our final pool comprised 50 white literature sources and 23 grey literature sources. The number of grey literature sources found, nearly half of the white literature sources can be considered a first confirmation of the need to include such sources in a literature review.

The distribution of sources per year is reported in Figure 2. The chart discriminates between grey and white literature sources. For grey literature sources, we considered the "first published" parameter as the year of publication without taking into account further modifications of the web pages. We observe a steady increase in the number of sources starting from 2016 for both types of literature. The increase in the number of grey literature sources with the year of publication is an expected result since the Effort Bounded strategy (i.e., including only the top N search engine hits [25]) for searching the Google engine tends to favour more recent sources. At the same time, older grey literature sources that are not permanently archived can become unavailable several years after publication. We cannot find an immediate rationale for the absence of grey literature sources from 2020 or for the decrease in the number of both types of sources in 2014-2015.

[2]https://doi.org/10.6084/m9.figshare.19804147

Just Accepted

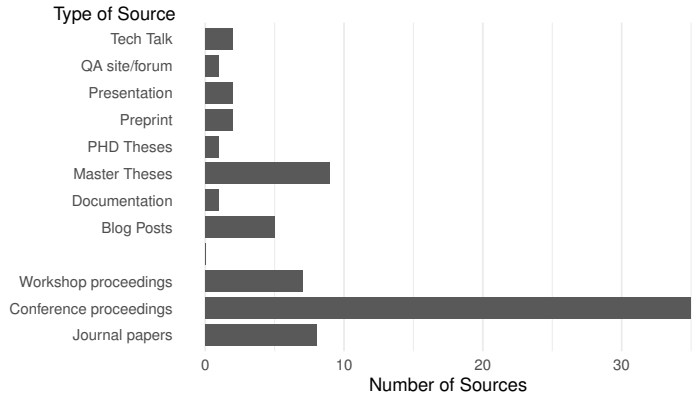

Fig. 3. Number of white and grey literature sources per category of source

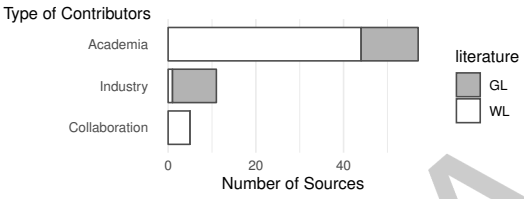

Fig. 4. Number of sources by type of contributors

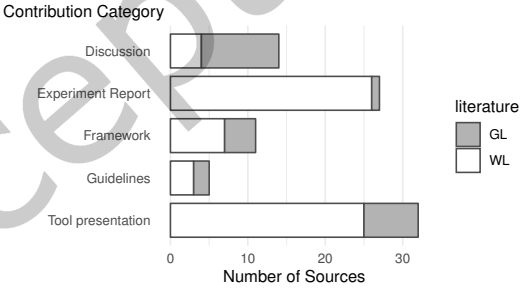

Fig. 5. Number of sources by category of contribution

The distribution of the sources for contribution type is reported in Figure 3. Regarding white literature sources, we collected 35 works published in conference proceedings, 8 journal papers and 7 works published in companion proceedings of conferences (i.e., workshop papers). Regarding grey literature, master theses were the most frequent type of contribution, with 9 sources. We also included one PhD dissertation in the pool and 3 preprints. These numbers testify that academia plays a fundamental role also in the production of grey literature sources about software testing gamification. Finally, we counted 5 blog posts, one documentation web page, 2 webinar presentations, one item from a question & answer website (namely, StackExchange), and 2 tech talks.

Figure 4 shows the number of sources per type of contributors. The sources were divided into three different categories: (i) *academia*, i.e., sources whose all authors were affiliated with universities or research institutions; (ii) *industry*, i.e., sources whose all authors were working in the industry; and (iii) *collaboration*, i.e., sources for which at least one author was affiliated to university or research institutions, and at least one author was working for the industry. All-academic sources outnumbered all industrial sources (44 vs 1 for WL, 13 vs 10 for GL). Collaboration works were present only among WL sources (5 items).

4 RESULTS

4.1 RQ1 - Mapping

Just Accepted

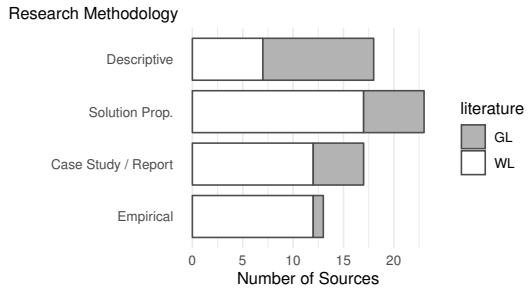

Fig. 6. Number of sources by research methodology

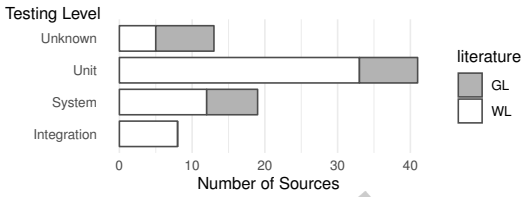

Fig. 7. Number of sources by mentioned testing level

4.1.1 Categories of contributions (RQ1.1). From all the resulting sources, five main categories of contribution were found:

(1) *Experiment report*, i.e. works that describe the methodology of the empirical evaluation carried out and/or report the results of the on-field experimentation of gamified mechanics;
(2) *Framework*, i.e. works that propose and present a set of rules or mechanics to be fully or partially implemented or to evaluate gamified approaches;
(3) *Tool presentation*, i.e. works that are aimed at the presentation of a tool to support or realise, in a gamified environment, the software testing process or any of its sub-activities;
(4) *Guidelines*, i.e. works that argue on how and when to use a specific tool, exploring effects, challenges and solutions possibly with data from previous experiments;
(5) *Discussion*, i.e. works arguing about problems related to GUI testing and analysing possible solutions or tool overview, explaining the main concepts, without going into details (posters, discussion about supposed effectiveness of game mechanics, etc.).

The difference between experiment reports and guidelines is that while the former argues on the result of one single case of application of a particular tool, the latter provides an analysis at a higher level of abstraction.

We assigned one or more categories (i.e., categories are not mutually exclusive) to each literature item by application of open coding. The bar plot in Figure 5 reports the number of sources that were assigned to each category. We observe that, for White Literature sources, the most common types of contribution were Experiment Reports (26 sources) and Tool presentations (25 sources), while the numbers of theoretical frameworks, guidelines and discussion papers were limited. On the other hand, for Grey Literature sources, we identify a prevalence of contributions that we flagged as Discussion (10 sources) followed by Tool Presentation (7 sources) and reports. This result suggests that the Grey Literature sources about gamification in software testing are less technical than White Literature ones, and that no actual validation of the benefits of gamification is carried out outside peer-reviewed academic research efforts.

4.1.2 Methodologies applied (RQ1.2). We divided the sources into four different methodological categories by utilising a subset of the categorisation provided by Petersen et al. [43]. We assigned a single category (the most formal applicable) to each source, i.e. research methodologies were considered mutually exclusive. The four research typologies that we considered are the following:

(1) *Descriptive and opinion studies*: The studies in this category provide anecdotal evidence and theoretical opinions about gamification in software testing. The studies in this category do not propose any technical solution to improve or analyse the context of gamification in software testing.

Just Accepted

(2) *Solution proposals*: The studies in this category describe and detail technical solutions (e.g., gamification tools, frameworks, and extensions of existing tooling with gamified mechanics). The studies only describe the solutions without performing any evaluation of them.
(3) *Experience reports and case studies*: The studies in this category perform and detail evaluations of tools, frameworks or gamified mechanics. Such evaluation is conducted utilising experience reports and/or industrial case studies. The studies in this category feature small-scale experiments that do *not* involve formal empirical methods.
(4) *Empirical studies*: The studies in this category provide evaluations of tools, frameworks and/or components for gamification of software testing by setting up formal empirical studies (e.g., with the planning of controlled experiments, formulation of research questions, and hypothesis testing).

In Figure 6 we report the distribution of the sources from the final pool according to the type of research methodology adopted. The largest set was that of solution proposals (23 studies, 17 WL and 6 GL), followed by Descriptive studies (18 studies, 7 WL and 11 GL), Case Studies and Reports (17 studies, 12 WL and 5 GL), and Empirical Research (13 studies, 12 WL and 1 GL). These results confirm an expectable predominance of solution proposals and descriptive studies in grey literature, with a higher prevalence of empirical research (12 sources out of 48) in White Literature.

4.2 RQ2 - Testing-Focused Characterisation

4.2.1 Testing levels. We referred to the traditional test automation pyramid to identify the testing levels covered by the gamification mechanics described in the sources. Therefore we identify three different testing levels:

- **Unit testing**: lowest testing level, with individual atomic code units tested separately;
- **Integration testing**: this second testing level is meant to combine the different existing units by verifying their collective behaviour;
- **System testing**: highest testing level, which is meant to test the finite system as a whole. System testing includes practices like End-2-End testing (i.e., testing the system by executing the end user's use cases), and GUI-based Testing (i.e., End-2-End exercised through the Graphical User Interface of the finalised system, which also includes a verification of the actual presentation of the SUT). Although, in many cases, the GUI and E2E test level can match, in some contexts, the system can be a complex environment of multiple elements interacting without the aid of any graphical presentation (e.g., in the IoT domain).

We identify in each source the testing levels to which the discussed gamification aspects were or could be applied. The categorisation was not considered mutually exclusive, since a single tool or technique can cover different levels of the testing practice. For papers not mentioning any specific testing level, we considered the testing level as undefined.

In Figure 7 we report the distribution of the sources according to the mentioned testing level. We identify Unit testing as the most mentioned testing level in sources applying gamification to testing (41 mentions, 33 WL and 8 GL), followed by system-level testing (19 mentions, 12 WL and 7 GL) and integration (8 mentions, all from WL). 13 sources (5 WL, 8 GL) did not explicitly specify any testing level, and no information could be deduced by reading the implementation details provided. It is worth noting that unit testing is more mentioned in white literature, while grey literature is equally focused on system and unit testing. This result can be justified by the more industrially-leaned orientation of grey literature, which is more likely to discuss or evaluate system testing frameworks to be directly used by practitioners. 14 sources in the pool (6 from WL, 8 from GL) did not explicitly mention any testing level to which the tools, frameworks or guidelines for gamification could be applied.

4.2.2 Testing phases. The second aspect of the testing discipline we analysed is the testing phase that the gamification tool, mechanic or framework supports.

Just Accepted

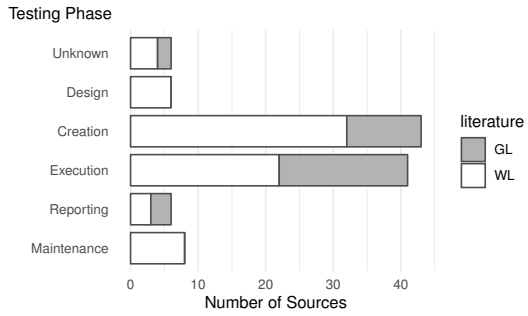

Fig. 8. Phases of the testing process mentioned in the selected papers

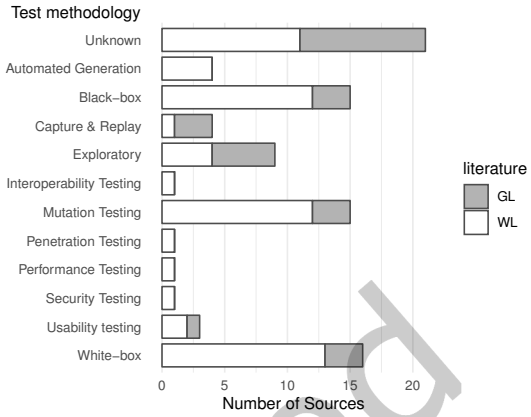

Fig. 9. Testing methodologies mentioned in the selected papers

We consider the following phases of the testing discipline:

(1) **Design**: the phase of planning a testing session, by defining the methodology, target, test conditions, test input, and test oracles;
(2) **Creation**: the implementation phase of the test suite, which involves the definition of all the test steps, and/or the code writing if the test methodology is scripted;
(3) **Execution**: the phase in which the test sequences are executed, and the results are evaluated. This activity can be performed automatically by executing existing test scripts or manually by a tester with a direct interaction with the system;
(4) **Reporting**: the description of the results obtained in the execution phase. A detailed report documents the found defects, faults and possible non-functional properties measured during the execution of test cases;
(5) **Maintenance**: the process of evolution of the test suite to resolve issues in the suite itself or to co-evolve with the SUT.

We collected from all the sources the test phases to which the discussed gamification aspects were or could be applied. The categorisation was not considered mutually exclusive, since it is possible that a single tool or methodology covers different phases of the testing process. For papers not mentioning any specific testing phase, we considered the testing phase as undefined.

In Figure 8 we report the distribution of the sources according to the mentioned testing phase. The majority of the literature items discussed the application of gamified mechanics in the phase of test creation (32 WL and 11 GL items), closely followed by test execution (22 WL and 19 GL items). We found a largely smaller number of items mentioning the phases of test design, maintenance and reporting. 6 sources (4 WL, and 2 GL) did not explicitly mention any testing phase. Interestingly, no GL items discussed the phases of test design and test maintenance, suggesting a lesser interest from non-academic sources in these phases of the verification and validation process.

4.2.3 Testing methodologies. Throughout the analysis of all the selected sources, we found the mention of 11 different methodologies for testing. We did not consider the testing methodologies as mutually exclusive in the literature sources as some items explicitly report the adoption of different testing methodologies during the use of the gamified environment. In figure 9 we report the distribution of the sources according to the mentioned test methodology.

Just Accepted

Table 4. Testing tools and frameworks mentioned in selected literature sources

Tool Name	Description	URL	Mentions
Bugzilla	Bugzilla is a web-based bug tracking system and testing tool.	https://www.bugzilla.org/	[GL10]
EclEmma	EclEmma is a free Java code coverage tool for Eclipse	https://www.eclemma.org/	[WL49]
EvoSuite	EvoSuite is a tool for automatic test cases generation with assertions for classes written in Java.	https://www.evosuite.org/	[WL06], [GL19]
GUnit	GUnit is a library which extends Google. Test and adds support for Gherkin to it.	https://github.com/cpp-testing/GUnit	[GL06]
IBM Rational Functional Tester	IBM Rational Functional Tester provides automated testing capabilities for functional, regression, GUI and data-driven testing.	https://www.ibm.com/products/rational-functional-tester	[WL49]
JaCoCo	JaCoCo is a free Java code coverage library distributed under the Eclipse Public License.	https://www.jacoco.org/	[WL14], [GL06]
JUnit	JUnit is an open source unit testing framework for Java programming language.	https://junit.org/	[WL01], [WL02], [WL03], [WL05], [WL14], [WL22], [WL27], [WL32], [WL39], [WL40], [WL49], [GL11], [GL16], [GL19]
Major	Major is a mutation analysis framework that enables to generate and embed mutants during the compilation and run the actual mutation analysis.	https://mutation-testing.org/	[WL06], [GL19]
Mockito	Mockito is an open-source testing framework for Java supporting test cases' creation, execution, and report.	https://site.mockito.org/	[WL02]
MuJava	MuJava is a system for Java that automatically generates mutants for both traditional mutation testing and class-level mutation testing.	https://cs.gmu.edu/offutt/mujava/	[WL10]
Nose	Nose is a tool that extends Unittest by collecting tests automatically and organising the library and test code.	https://nose.readthedocs.io/	[GL23]
PMD	PMD is a static source code analyser allowing the identification of code smells.	https://pmd.github.io/	[WL04]
Randoop	Randoop is a unit test generator for Java. It automatically creates unit tests for your classes in JUnit format.	https://randoop.github.io/randoop/	[WL14]
Redmine	Redmine is an open-source web-based cross-platform and cross-database project management and issue tracking tool.	https://www.redmine.org/	[WL03]
Scout	It records and learns from manual test sessions. The system keeps track of coverage and issues and can estimate the quality of the app so that the tester knows when to stop testing.	https://store.synteda.se/product/ eye-scout/	[WL33] [GL02], [GL03], [GL05]
Selenium	Selenium is an open-source tool for the automated management of browsers. It is used as a web testing framework.	https://www.selenium.dev/	[WL01]
Testlink	Testlink is an open-source web-based test and requirement management system allowing the creation, management and plan of test cases.	https://testlink.org/	[WL01], [WL03], [WL11]
Unittest	Unittest is a framework supporting test automation for Python and aggregating tests into collections. Tests are independent of the reporting framework.	https://docs.python.org/3/library/unittest.html	[GL23]

The methodologies that were mentioned the most in the selected sources were *White-box* testing (13 WL, 3 BL), *Black-box* testing (12 WL, 3 BL), *Mutation* testing (12 WL, 3 GL). Some methodologies to test non-functional properties were present: *Performance* testing (one source), *Security* testing (one source), *Penetration* testing (one source), *Interoperability* testing (one source), *Usability* testing (three sources). Over the whole set, there were 11 WL sources and 10 GL sources not mentioning any specific testing methodology. 21 sources (11 WL, 10 GL) did not explicitly state any testing methodology on which Gamification was applied.

From the selected literature, it is evident how the focus of GL is less aimed towards developers or testers who define test scripts at the code level: the most mentioned testing methodologies in GL were, in fact, *Exploratory* testing (5 sources) and *Capture & Replay* (3 sources).

4.2.4 Testing frameworks/tools adopted. We report all the existing testing tools or frameworks that were adopted by the retrieved items in Table 4. We retrieved mentions of 18 different testing tools in our literature pool. For each testing tool, we report the tool name, a short description, a URI for tool retrieval, and the literature sources where it is mentioned.

What emerges from the Table 4 is that the most used tool is by far JUnit, the most widely used tool to develop, execute and report the results of unit test cases in Java. The tool was mentioned in 14 different literature sources.

Just Accepted

Table 5. Language used in the discussed gamification tools or frameworks

Language Name	Mentions
C	[WL31], [WL48], [WL50]
C++	[WL15], [WL50]
Java	[WL02], [WL03], [WL04], [WL05], [WL06], [WL10], [WL13], [WL14], [WL19], [WL21], [WL22], [WL24], [WL26], [WL27], [WL32], [WL39], [WL40], [WL44], [WL45], [WL47], [WL49], [WL50], [GL01], [GL06], [GL11], [GL14], [GL15], [GL16], [GL19]
Javascript	[WL42]
Python	[WL31], [WL50], [GL23]
SQL	[WL50]

Table 6. Target domain for the proposed gamification tools or frameworks

Domain	Mentions
IoT	[WL38]
Mobile	[WL33], [WL35], [WL36], [GL03], [GL13]
Web	[WL11], [WL29], [WL33], [GL02], [GL04], [GL05]

Other tools that were mentioned multiple times were Scout, an augmented tool for visual GUI testing described in white literature by Nass et al. (4 mentions), Major Mutation Framework, EvoSuite and the code coverage tool JaCoCo (all with 2 mentions). Several tools were mentioned only in GL Sources, Bugzilla, GUnit, Nose, and Unittest. It is worth mentioning that over the 60% of the items (45 over 73) did not specify the tool used (either bespoke or already available).

4.2.5 Languages and domains. A further categorisation can be performed about the programming languages and domains addressed by the pool of studies. In Table 5 we report the target languages mentioned by the papers and the specific literature items mentioning them. According to our investigation, the most targeted language is Java, in accordance with the most used tool (JUnit), which is specifically used for unit testing in Java. Our findings agree with the study of Abdullahi et al. [2] which reports Java as the most used testing language. In fact, we observe that a consolidated infrastructure is often the starting point for the development of new tools: this explains why many gamified tools adopt Java as the target language. Additionally, StackSocial reports Python as the most used in universities to teach coding [50], for this reason, it appears natural to also consider Python for gamified testing in education. Although the interest in Python is rising, its actual usage spread in universities around the world is more recent mainly because academic courses are less prone to change a consolidated language than practitioners. It has to be underlined that the majority of the contribution did not specify any language (39 over 73). We did not assign any categorical value to all the studies that did not explicitly mention a language.

In Table 6, we report the target domains mentioned by the papers and the specific literature items mentioning them. We identified two main domains to which pieces of literature could be assigned: the web domain (6 sources) and the mobile domain (5 sources). One literature source mentioned both web and mobile domains. One additional paper was specifically tailored for the Internet of Things domain. The sparsity of the data also characterises the testing domain: only 11 items over a total of 73 (less than 20%) explicitly identified a domain for the gamified technique or tool. It can be assumed that most of the remaining sources describe approaches that are not domain-specific.

4.2.6 Educational or practical focus. We discriminate between three different focuses for the sources that we consider:

- *Educational*: The source presents a gamified environment or tool for the only purpose of educational aspects of the software testing discipline.
- *Practice*: The source presents a gamified environment or tool whose primary purpose is to support and improve the execution of the testing activity at any phase or level.

Just Accepted

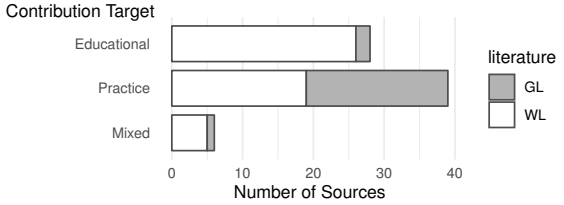

Fig. 10. Educational or practice focus of the sources

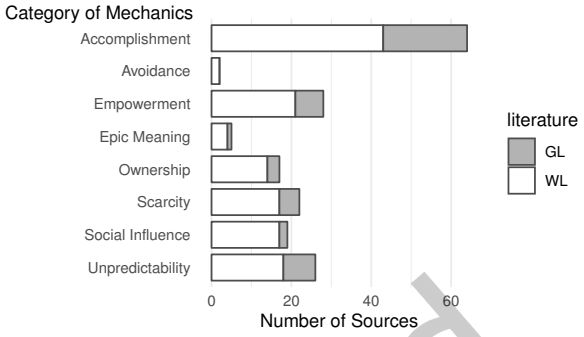

Fig. 11. Number of mentions for each category of mechanics in the Octalysis framework

- *Mixed*: The source presents a gamified tool, environment or framework which can be used to improve both educational aspects or to perform at least one of the testing phases.

Figure 10 reports the distribution of sources for each of the three categories, divided by grey literature and white literature. The majority of the sources deals with the practical aspect of the application of gamification to software testing, with an even distribution between grey literature and white literature. Educational-focused sources come principally from items of white literature. Unsurprisingly, we notice that grey literature provides contributions mostly for testing practice.

4.3 RQ3 - Gamification-Focused Characterisation

4.3.1 Gamification Mechanics. In Table 7 we report all the gamification mechanics proposed, adopted or mentioned in the final pool of sources. We applied a coding procedure to the mechanics described in the sources, by labelling each one with one of the mechanics codified in the Octalysis framework. In the table, we report the name of the mechanic, its definition, the synonyms with which the mechanic is also mentioned, and all the sources in which it is mentioned. We also report for each mechanic the Octalysis core driver to which it can be assigned. This way, we are able to provide a higher-level characterisation of the gamification mechanics mentioned in software testing literature.

We measure a median of 10 citations per gamification mechanic. 10 mechanics had more mentions than the median. The most popular mechanic was *Score*, which was implemented or discussed in 45 different sources, followed by the highly-correlated leaderboard mechanic (36 mentions), and graphical feedback (22 mentions). The less mentioned mechanics were *Epic Meaning* (one mention), *Punishment* and *Auctions* (two mentions each). The absence of punishment mechanics suggests that research and development in the gamified testing field have mostly focused on providing positive motivation to the users instead of negative disincentives.

In Figure 11 we report the number of mentions for each gamification core drive, according to the Octalysis Framework. From the graph, it is evident how the main focus of available gamification implementations for software testing is to provide *Accomplishment* to the users as a positive means of motivation (43 WL and 21 GL mentions). Conversely, only two WL sources implemented mechanics related to the *Avoidance* dimension, which is related to the enforcement of correct patterns by applying punishments and maluses to non-conforming users. Few mentions were also gathered by the gamification mechanics related to the *Epic Meaning* macro category of the Octalysis framework. We consider such a low number of mentions as an effect of the still prototypal nature of most of the described tools, which did not allow for the implementation of complex narratives.

Just Accepted

Table 7. Game mechanics mentioned and utilised in the pool of literature sources

Mechanic	Octalysis Core Drive	Definition	Synonyms	Mentions
Achievements	Accomplishment	A mechanism to show the user his or her progress and achievements within the system.	-	[WL16], [WL17], [WL24], [WL25], [WL28], [WL47], [GL12], [GL17], [GL23]
Auction	Ownership	A part of the gaming session where players bid to obtain particular virtual goods or real resources.	-	[WL30], [GL18]
Badges	Accomplishment	Graphical rendition of a particular achievement that can be used to enrich a player's profile or to certificate the fulfillment of certain target	Medals, Titles.	[WL01], [WL03], [WL13], [WL16], [WL17], [WL20], [WL28], [WL34], [WL36], [WL39], [WL47], [GL01], [GL04], [GL06], [GL10], [GL14], [GL16], [GL18], [GL21]
Challenges	Empowerment	Challenges to motivate a player to perform a certain task under a particular set of assigned condition.	-	[WL01], [WL14], [WL21], [WL22], [WL25]
Duels	Social Influence	A clash involving two or more players or factions where only one of them may prevail.	Battle, Combat System	[WL02], [WL05], [WL06], [WL07], [WL12], [WL14], [WL16], [WL17], [WL19], [WL27], [WL32], [WL40], [GL11], [GL19]
Easter egg	Unpredictability	A message, image, or other types of hidden feature that can be used to encourage exploration from the user and can add unpredictability to the gamified session.	-	[WL33], [GL02], [GL03], [GL05]
Experience points	Accomplishment	Marker that a system use to represent the progression of a player in completing tasks or in the step necessaries to complete it.	Reputation Building, Progress Bar	[WL03], [WL18], [WL33], [WL42], [WL47], [GL02], [GL03], [GL04], [GL05], [GL15]
Feedback	Empowerment	Additional information about the performed task provided to the user, or encouraging messages to continue in performing the task.	Continuous Feedbacks, Visual Artifacts, Discovery Marker	[WL01], [WL02], [WL05], [WL06], [WL10], [WL18], [WL22], [WL25], [WL33], [WL34], [WL36], [WL39], [WL40], [WL43], [WL44], [GL01], [GL02], [GL03], [GL05], [GL06], [GL08], [GL09]
Hints	Empowerment	Information that can be given to players under their request to support them in solving puzzles, answering questions or completing generic tasks.	Cascading Information, Permission to Fail	[WL15], [WL26], [WL28], [WL36], [WL44], [WL46]
Leaderboard	Accomplishment	A system providing a ranking and comparison between the scores obtained by all users of the gamified system.	Ranking, Competition	[WL01], [WL02], [WL03], [WL04], [WL05], [WL06], [WL07], [WL08], [WL09], [WL10], [WL13], [WL14], [WL16], [WL17], [WL21], [WL22], [WL24], [WL26], [WL28], [WL33], [WL39], [WL40], [WL41], [WL47], [WL48], [WL49], [GL01], [GL02], [GL03], [GL05], [GL06], [GL07], [GL14], [GL16], [GL18]
Levels	Unpredictability	Player's progression (typically obtained through Experience Points) or system's progressive complexity	Worlds	[WL01], [WL03], [WL08], [WL09], [WL12], [WL13], [WL15], [WL16], [WL17], [WL22], [WL24], [WL26], [WL28], [WL34], [WL44], [GL04], [GL08], [GL14]
Profile	Ownership	The system offers a space that contains information about the specific user and can be customised by the user.	Avatar	[WL03], [WL04], [WL16], [WL17], [WL18], [WL20], [WL25] [WL34], [WL35], [WL36], [WL41], [GL05], [GL17], [GL20]
Punishments	Avoidance	Consequences to bad behaviour or performance performed by a player, concretised into penalties	Penalties	[WL25], [WL34]
Puzzles	Accomplishment	Challenges with simple rules that require specific actions by the users.	-	[WL02], [WL05], [WL06], [WL15], [WL19], [WL23], [WL27], [WL32], [WL40], [WL44], [WL46], [WL50], [GL11]
Quest	Scarcity	The system or other users ask the user to perform a certain activity under predefined conditions in order to advance in the story.	-	[WL10], [WL16], [WL17], [WL18], [WL24], [WL03], [WL34], [WL47], [GL15], [GL17], [GL08]
Quiz	Scarcity	A series of questions that can have a set of possible answers or open questions.	-	[WL04], [WL08], [WL49]
Randomization	Unpredictability	Usage of a random generator to generate an event or to decide its outcome.	Dice Game, Random Rewards, Randomizer, Spin the Wheel	[WL11], [WL42], [GL13], [GL22]
Rewards	Accomplishment	A form of reward that is given by the system in response to a milestone or is exchanged between player in response of an event. It can lead to awarding in the real world.	Gift	[WL03], [WL07], [WL20], [WL25], [WL26], [WL29], [WL34], [WL39], [WL41], [WL49], [GL07], [GL08], [GL09], [GL10], [GL12], [GL21], [GL22]
Score	Accomplishment	Users can earn virtual points after performing specific actions in the gamified system. The score is tracked in the system, and in some cases can be used to obtain other game mechanics (e.g. experience points, virtual goods) .	Points	[WL01], [WL02], [WL04], [WL05], [WL06], [WL07], [WL08], [WL09], [WL10], [WL12], [WL13], [WL14], [WL15], [WL16], [WL17], [WL18], [WL19], [WL20], [WL21], [WL22], [WL26], [WL27], [WL28], [WL32], [WL33], [WL34], [WL38], [WL39], [WL40], [WL41], [WL44], [WL48], [WL49], [GL01], [GL02], [GL03], [GL05], [GL06], [GL08], [GL11], [GL14], [GL16], [GL19], [GL20], [GL21], [GL22]
Social interaction	Social Influence	The feature of a system allowing players to interact in a textual or vocal way, allowing direct information exchange between different users.	Social Features	[WL01], [WL24], [WL03], [WL49]
Storytelling	Epic Meaning	A narration layer in which players and actions are involved in a fictional story that adds context to actions and real or fictional characters that are part of the system.	Stories, Story	[WL11], [WL24], [WL42], [GL22]

Just Accepted

Teams	Social Influence	Cooperation between different players with the same goal.	Collaboration	[WL16], [WL17], [WL34], [WL38], [GL20]
Timing	Scarcity	Users are given a certain amount of time in which they ought to perform or complete a specific activity.	Timing Experience, Time Pressure, Timer	[WL04], [WL09], [WL13], [WL15], [WL18], [WL26], [WL30], [WL39], [GL13], [GL14]
Virtual goods	Ownership	Game assets belonging to a virtual environment can be traded to redeem virtual artefacts. Unlike the score, which is usually used for ranking purposes, virtual goods' main purpose is to be exchanged to obtain other game assets.	Virtual Currency, Currencies, Shop System	[WL10], [WL16], [WL17], [WL20], [WL30], [WL34], [WL34], [WL42], [GL05], [GL17], [GL18]

4.3.2 Gamification tools and frameworks. In Table 8 we report all tools and/or frameworks mentioned or proposed in the selected literature. In the table, we report for each tool its name, a brief description, the adopted gamification mechanics, the testing methodology supported by the tool (if specified), and the literature item(s) where the tool or framework is mentioned. For tools that are not explicitly provided with a name, we report the name of the authors of the papers where the tool is first described. If the tool is mentioned multiple times in different literature items and different mechanics are mentioned, we report in the table all the mechanics that are mentioned at least once in the set of papers mentioning the tool. As is evident from the table, we have found the description of 30 different tools and/or frameworks in our pool of literature items. Several tools were mentioned multiple times in the collected sources:

- CodeDefenders, originally described by Rojas and Fraser [WL05] is a serious game which has originally been used to teach mutation testing in an academic context; which prototype has first published in early 2016. In the following years, the tool received further development introducing more features, along with several related experiments, which enriched the existing literature with experience reports of Code Defenders usage;
- HALO, originally described by Shet et al. [48] is an approach to gamifying software engineering with the MMORPG (Massively Multiplayer Online Role-Playing Game) game approach. The original tool has been utilised in [WL24] as a basis to implement the *"Secret Ninja"* approach by Kiniry and Zimmerman [32], which implies the application of gamification aspects while keeping the users unaware of their presence. The experience with the usage of HALO has been published later in other studies [WL47] and [GL15];
- VU-BugZoo was originally described by Silvis-Cividjian et al. [WL45] in a poster from 2020. Its usage has been reported in two papers from 2021 ([WL31] and [WL37]) along with a detailed description of the tool that extends the original proof of concept;
- WReSTT-Cyle (Web-based Repository of Software Testing Tools Cyber-Enabled Learning Environments) was originally described by Clarke et al. [11] as a repository of learning objects created with the goal of supporting software testing education. Subsequently, the project evolved, assuming the name SEP-CyLE (Software Engineering and Programming) firstly, extending the repository to support new topics, and finally, STEM-CyLE, with the extension to support the learning process of all STEM disciplinesGamification was introduced to support and empower the cyber-enabled learning environment, as mentioned in [12] and [WL13].
- *Auction-Based Bug Management*, has originally been published as a master thesis [GL18] and subsequently in a journal paper [WL30]. It proposes a gamified approach for bug management, where developers compete in a virtual auction to obtain the assignment of bugs.

It is worth noticing that three different tools were mentioned only in GL sources:

- *CoCoT* is an expansion for *TeC*, an existing serious game that supports team coordination and communication skills. The tool is adapted to support educational aspects of testing, specifically statement coverage, and teamwork skills. This tool has been documented in a PhD thesis by Alsaedi [GL20];
- *Kucmann*'s master thesis documents a tool that gamifies mutation testing inspired by *Code Defenders*, that replaces players with machine learning agents [GL19];

Just Accepted

- Finally, one master's thesis from the pool of grey literature proposed a prototype tool based on the framework of Cacciotto et al. [WL33] for the gamification of Capture & Replay testing of web applications, implementing profiles, currency and achievement management [GL05].

No testing methodologies have been explicitly mentioned for several tools (i.e., Auction-Based Bug Management, CleanGame, CodeMetropolis, DevRPG, GamiWare, GOAL and Testable).

Table 8. Tools described and used in the collected literature

Name	Description	Mechanics	Methodology	Mentions
Auction-Based Bug Management	It is a tool with the aim of improving software productivity while developers are assigning tasks. Users can bid in auctions to obtain the assignment of bugs to fix.	Auction, virtual goods, timing, badges, leaderboard	-	[WL30], [GL18]
Bodhi	Bodhi is a two-player game in which each player is shown a piece of code snippet and is asked to choose whether their partner would think there is a buffer overflow vulnerability at a given position in the code.	Score, leaderboard	White Box Testing	[WL48]
Bug Catcher	Bug Catcher is a web-based system for running software testing competitions.	Score, leaderboard, timing, hints, levels, rewards	Black Box Testing	[WL26]
Bug Hunter	Bug Hunter is a gamified environment in which students have the goal to reach the final level achieving the first ranking position. Bug Hunter includes a live feedback mechanism and forums to foster respectively competition and social interaction.	Achievements, profile, badges, duel, leaderboard, level, score, quest, teams, virtual goods	Black Box Testing	[WL16], [WL17]
Cacciotto et al.	A framework for building a plugin for GUI testing tools exploiting gaming concepts.	Score, leaderboard, feedback, easter egg, experience points	Exploratory Testing, C&R	[WL33]
CleanGame	A gamified software tool aimed to teach code smell detection, composed of two independent modules: Smell-related Quiz (i.e. questions about code smells with multiple-choice answers) and Code Smell Identification, which focuses on identifying code smells in the source code.	Quiz, score, profile, timing, leaderboard	-	[WL04]
CoCoT	The Code Coverage Testing Team (CoCoT) game is an expansion to the Team Coordination game (TeC), a zero-fidelity collaborative simulation.	Profile, score, teams	White Box Testing	[GL20]
CodeDefenders	Code Defenders is a turn-based mutation testing game: two players are involved in each game: an attacker with the aim of introducing faults and a defender with the aim of writing tests.	Duel, score, leaderboard, puzzles, feedback, challenges	Mutation Testing	[WL02], [WL05], [WL06], [WL14], [WL19], [WL32], [WL40], [GL11]
CodeMetropolis	One such tool is CodeMetropolis which is built on top of the game engine Minecraft and which uses the city metaphor to show the structure of the source code as a virtual city.	Feedback	-	[WL43]
CoverBot	CoverBot is a game for teaching statement coverage: students act as a character whose survivability depends on how effectively the player can execute all lines of code in a given level with the fewest amount of inputs possible.	Score, duel, levels	White Box Testing	[WL12]
DevRPG	DevRPG incorporates RPG-like mechanics to the everyday software development activity allowing the creation of a character that reflects the player's very own skills and actions.	Achievements, challenges, rewards, punishments, feedback, profile	-	[WL25]
EMVille	EMVille is a web-based system through which the human experts get involved in the process of analysing instances of the equivalent mutant problem through a game.	Quest, score, virtual goods, leaderboard, feedback	Mutation Testing	[WL10]
HALO	HALO is a plugin that uses game-like mechanics to make the whole software engineering process, particularly the software testing process, more engaging and social.	Social interaction, quest, storytelling, achievements, levels, leaderboard	Black Box Testing, White Box Testing	[WL24], [WL47], [GL15]
Gallotti	It is a gamification plugin for visual GUI testing of web applications	Score, leaderboard, feedback, easter egg, experience points, profile, virtual goods	Exploratory Testing, C&R	[GL05]
GamiWare	Gamiware is a SaaS open-source tool to support gamification in iterative software processes.	Duel, score, rewards, leaderboard	-	[WL07]
GATE	GATE is a game-based software testing tool that uses human computation to support automatic test generation, improving test adequacy.	Hints, puzzles	Automate Test Case Generation	[WL46]
GOAL	GOAL is a framework that supports the integration of gamification in a software engineering environment.	Score, levels, leaderboard, badges, social interaction, feedback, challenges	-	[WL01], [WL03]
GamiTracify	GamiTracify is a gamification framework with the aim of infusing engagement into human-centric traceability tasks to record trace links.	Score, feedback, experience points, profile, timing, quest	White Box Testing	[WL18]
Greenify	Greenify is a game with a purpose that generates test data based on the program's corresponding control flow graph with the aim of covering special test paths.	Levels, timing, puzzles, hints, score	White Box Testing	[WL15]
Kucmann	It is constructed in a two-player game setting. The attacker selects mutants of a program, plays against the defender, and selects test cases to find the mutants; their objective is to win by the kill factor.	Duel, score	Mutation Testing	[GL19]

Just Accepted

IoTCityLab	IoTCityLab is a collaborative role-based multiplayer game for IoT testing.	Score, teams	Security Testing, Interoperability Testing, Performance Testing	[WL38]	
OATMEAL	The OATMEAL gamified tool is an online interface which presents data about the correctness of a set of code samples on a set of test cases.	Puzzles	White Box Testing	[WL50]	
OBB	OBB is a community-based platform for the dissemination of web vulnerabilities.	Rewards	Penetration Testing	[WL29]	
Pipe Jam	Pipe Jam presents the game player with a set of related ball-and-pipe puzzles. Each pipe is either narrow or wide, and the player is allowed to control the width of some pipes. Each ball is either small or large. The player's goal is to ensure that the balls never get stuck.	Levels, puzzles, score, hints, feedback	Black Box Testing	[WL44]	
Puzzle-based Automatic Testing (PAT)	PAT is a puzzle-based testing environment that generates test cases by decomposing complex problems in small puzzles solved by humans	Puzzles	Automated Test Case Generation	[WL23]	
Rings	Rings is a game with a purpose for test data generation designed such that non-technical players can implicitly generate test data for program units when solving the game's puzzles.	Score, leaderboard, achievements, badges, hints, levels	Automated Test Case Generation	[WL28]	
Sun	It is a gamified software testing training system supporting practical training of black box, white box testing and defect repair	Score, leaderboard, levels, quiz	White Box Testing, Black Box Testing	[WL08]	
Testable	Testable is a gamified tool designed to be used in face-to-face education in undergraduate Computer Science related courses.	Storytelling, experience points, virtual goods, randomization	-	[WL42]	
VU-BugZoo	VU-BugZoo is a digital platform to teach software testing based on a repository of faulty code.		Mutation Testing	[WL31], [WL45]	[WL37],
WReSTT-CyLE (SEP-CyLE)	They are a cyberlearning environment that uses several learning and engagement strategies to help students to learn software testing.	Score, leaderboard, badges, timing, levels, rewards, social interaction, quiz	White Box Testing, Black Box Testing	[WL13], [GL14]	[WL49],

4.3.3 Advantages of gamified software testing. After the application of *Open Coding*, we came up with 35 different codes, i.e. categories of advantages discussed by the selected papers. The application of *Axial Coding* resulted in the identification of three main categories of advantages. In Table 9 we report the complete list of codes in each category and, for each code, its description and the sources where it is mentioned.

- **Better User Experience**. We include in this category all the discussed benefits related to the user's experience of the practice of software testing when gamified mechanics are adopted.
  Under this category, the most mentioned benefit (20 WL and 8 GL sources) is a higher *Engagement* (or involvement) guaranteed for the testing activity when gamified mechanics are implemented. Clegg et al. report quantitative results about enhanced engagement in the field of unit testing [WL32]; Sun reports a better engagement in testing learning by 94% of the sample of students interviewed [WL08].
  Many studies (15 WL and 2 GL sources) identified gamified activities are more *fun* than traditional testing activities. This aspect was especially highlighted in the context of testing education: Fraser et al. report that, in the context of unit testing teaching, *"Although students tend to claim they do like to write tests even outside the game, they confirm it is more fun to do so as a part of the game"* [WL14].
  Several gamification mechanics have been proven to provide additional *Motivation* to the involved testers. Saloum and Rissanen report, in the context of unit testing, that *"the use of badges motivated most of the subjects to write better unit tests [...] and to complete the tasks."* [GL16]. Other mechanics that strongly enhance the user experience of testers are *Cooperation*, and also healthy *Competition* with other players.
- **Higher Efficiency**. *Efficiency*, as defined by the ISO 9001 standard, is *"the extent to which time, effort or cost is well used for the intended task or purpose"* [1].
  We include in this category all the discussed advantages related to reduced efforts and costs in test case definition, generation or execution caused by the application of gamified mechanics.
  One of the most positively commented aspects of the environments described in the elicited sources is the possibility of adding *Informative Content* to the gamified practices, thereby reducing the effort required by the testers to gather information to complete the required testing procedures. Lorincz et al., as the result of an experiment with students, report that *"having clear goals and destinations encourages information gathering to achieve [the desired tasks]"* [WL11].

Just Accepted

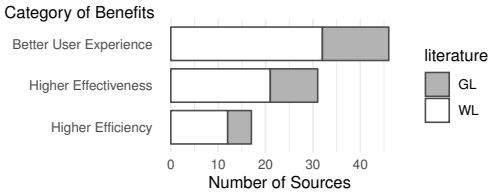

Fig. 12. Number of sources mentioning the different categories of benefits

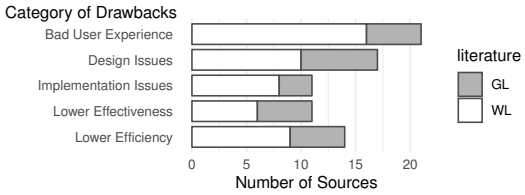

Fig. 13. Number of sources mentioning the different categories of drawbacks

Several sources consider gamification a means to reduce the required effort to perform test-related activities. The report by Usfekes, related to an auction-based crowdsourced mechanism for bug resolution, reports that the use of the tool *"makes the allocation of resources more effective, as the effort from the general public can be utilised to [perform testing activities]"* [GL18]. *Crowdsourced Contributions* is mentioned in several sources and are seen as a primary mean to improve the efficiency of gamified testing techniques. Robson et al. argue about the opportunities offered by crowdsourcing: *"It is rational to assume that the number of unit testers of an organisation is significantly smaller than the population of a game-players community, and programmers' cost is remarkably higher than a casual player"* [WL28].

- **Higher Effectiveness**. *Effectiveness*, as defined by the ISO 9001 standard, is *"the extent to which planned activities are realised and planned results are achieved"*[1].
  We include in this category all the discussed advantages related to an enhancement of the outcomes of the gamified testing procedures.
  Many sources of our final pool identified higher effectiveness in the purpose of testing education (*Improved Learning*). For instance, the empirical study conducted by Alsaedi reports *"a significant difference in the students' test scores before and after [the introduction of gamification]"* [GL20]; Sun also reports significant positive results in the scores for software testing courses [WL08]. Many studies (6 WL, 3 GL) report higher coverage as the measure of increased effectiveness of the gamified testing tool; for instance, Fraser et al. report an increased branch coverage and mutation score for gamified mutation testing [WL40]. Six sources mention a general increase in effectiveness guaranteed by gamified mechanics. Other specific effectiveness-related aspects are mentioned in other sources, e.g. effectiveness in finding bugs, identifying code smells, finding issues and adding comments to the original code. For instance, dos Santos et al. report the results of an experiment with CleanGame, where the subjects were able to identify approximately twice as many codes smells with respect to non-gamified techniques [WL04].

In Figure 12 we report the number of manuscripts that report at least one advantage for each of the main categories. It is evident that gamification is mainly evaluated in the selected sources in terms of the benefits that are provided to the tester in terms of user experience (32 WL and 14 GL sources). A lower overall amount of sources mentioned benefits related to effectiveness (21 WL, 10 GL) and efficiency (12 WL, 5 GL). The inclusion of Grey Literature allowed the identification of one benefit related to user experience (*Empowerment*), which was not mentioned in white literature.

4.3.4 Drawbacks of gamified software testing. After the application of *Open Coding*, we came up with 23 different codes, i.e. categories of disadvantages discussed in the selected papers. The application of *Axial Coding* resulted in the identification of five main categories of drawbacks. In Table 10 we report the complete list of codes and for each code, the category, and the sources where it is mentioned.

- **Design Issues**. Several signalled issues related to gamification that are mentioned in the sources are related to the design of the game mechanics implemented. Multiple sources (2 from WL and 6 from GL)

Just Accepted

Table 9. Advantages of the application of gamification to software testing

Category	Name	Description	Mentions
Better User Experience	Challenge	Users perceived the tasks as a dare to be fulfilled in the best way	[WL26], [WL31], [WL44]
	Cooperation	Interaction with other users towards the same goal	[WL11], [WL21], [WL23], [WL47], [GL12], [GL20]
	Competition	Interaction with other users competing to be the best under a certain metric	[WL01], [WL02], [WL09], [WL10], [WL11], [WL25], [WL26], [WL28], [GL07], [GL09], [GL22]
	Control	Monitoring of gamified activities	[WL25]
	Dynamicism	A more open environment with greater interactions	[WL21]
	Ease Of Use	Intuitiveness of the environment	[WL4], [WL44], [GL08]
	Engagement	Users' feeling of being transported emotionally	[WL02], [WL04], [WL06], [WL08], [WL10], [WL11], [WL13], [WL14], [WL17], [WL20], [WL23], [WL24], [WL25], [WL29], [WL32], [WL36], [WL37], [WL44], [WL47], [WL48], [GL01], [GL02], [GL03], [GL06], [GL09], [GL12], [GL13], [GL22]
	Empowerment	Users felt to be in control	[GL09], [GL22]
	Fun	Users had fun while using the tool	[WL01], [WL02], [WL06], [WL11], [WL12], [WL14], [WL16], [WL17], [WL20], [WL24], [WL28], [WL31], [WL32], [WL36], [WL48], [GL01], [GL04]
	Higher Satisfaction	Users were pleased of completing their tasks	[WL17], [WL33], [WL36], [WL40]
	Motivation	Users were more willing to perform the activity	[WL09], [WL13], [WL18], [WL20], [WL21], [WL26], [WL28], [WL30], [GL04], [GL07], [GL09], [GL16]
	Perceived Utility	Users understood the usefulness of what they were doing	[WL44], [WL47], [WL48]
	Replay Value	Interest in re-utilising the gamified system	[WL28], [WL48], [GL04]
	Stress Reduction	Lower levels of stress perceived by the tester	[WL36], [GL15]
Higher Efficiency	Crowdsourced Contributions	Community working together contributes more than a single user	[WL23], [WL28], [GL12]
	Decreased Costs	Operational, infrastructural or personnel costs decreased	[WL15], [WL28]
	Flexibility	The environment created was considered very flexible	[WL03]
	Informative Content	Content provided to support users' goal	[WL01], [WL04], [WL11], [WL20], [WL43], [WL47], [GL09], [GL18]
	Reduced Effort	Users need less operations, time or resource to complete the same tasks	[WL10], [WL23], [WL46], [GL08], [GL10], [GL18]
Higher Effectiveness	Higher Coverage	Improvement of coverage metric (code, widget, etc.)	[WL06], [WL15], [WL23], [WL33], [WL40], [WL46], [GL01], [GL02], [GL03]
	Higher Mutation Score	Improvement of the obtained mutation score	[WL06], [WL40]
	Improved Learning	Improvement of average learning effect in students	[WL02], [WL08], [WL13], [WL16], [WL21], [WL24], [WL29], [WL31], [WL37], [WL40], [WL49], [GL05], [GL06], [GL11], [GL15], [GL20], [GL22]
	Increased Effectiveness	Improvement of effectiveness of a testing practice, according to the definition	[WL09], [WL10], [WL18], [WL48], [GL07], [GL15]
	More Bugs Found	Users identified more bugs	[WL02], [GL01]
	More Code Smells	Users identified more code smells	[WL04]
	More Comments Added	Users provided more bug reports	[WL01]
	More Issues Found	Users identified more issue	[WL01]
	More Requirements Covered	Users created a test suite covering more requirements	[WL01]
	More Test Cases Generated	Users produced more test cases	[WL02], [WL46], [GL06], [GL07]

mention design issues related to specific gamification mechanics and how they proved not suitable for their purposes. For instance, Arnarsson and Johannesson report that *"the badges, in general, are not seen as motivating as other mechanics [...] the design of the badge system was unbalanced"* [GL06]; Bryce et al., in the context of software testing education, report that *"[the subjects] did not like the design choice because we did not tell them how many bugs are in each problem"* [WL26]. Garcia et al. mention possible misalignment issues, i.e. *"gamification should not interfere with other improvement actions being implemented at the same time"* [WL01]. Reported design issues are also related to the selection of specific gamified mechanics for *wrong target* or purposes (5 WL and 3 GL mentions). Breum reports that *"experience points can have some drawbacks it [sic] can create a more competitive environment which may not be suited for users who do not have a competitive nature"* [GL04]. Finally, five WL sources from the pool report the necessity to carefully calibrate the gamified mechanics to disincentivise possible *cheating* actions from users trying to exploit them to gain benefits.

Just Accepted

- **Implementation Issues**. A limited number of sources in the selected pool mentioned implementation-related issues for gamified tools and frameworks. The main researchers' implementation concerns were related to the scalability of the approach (*"For complex test path constraints, the game's length may be relatively long. In this condition, the game will be difficult to solve"* [WL28] and to generalisability to different organisations [GL01] or to different testing techniques and strategies [GL15].
- **Bad User Experience**. In this category of drawbacks, we include all those related to the negative impacts of the mechanics in terms of the final user's experience. The most mentioned user experience issue was a high learning curve for the users of the gamified techniques. For instance, Berkling and Thomas, in an educational context, report that *"the benefits of a game environment for the classroom are not evident to the students"* [WL22]. Other frequently discussed user-centred drawbacks are related to the difficulty in making users or organisations transition to the usage of gamified mechanics (*Change Resistance*). As Harranz et al. report, *"Achieving the commitment of the top managers is a very hard task."* [WL07].
- **Lower Effectiveness**. Many sources (5 WL, 4GL) report a reduced or unchanged effectiveness of the testing methodologies or procedures when gamification is introduced. A grey literature source reports that the visualisation of reached objectives can reduce the effectiveness of a gamified tool since the tester can feel a sense of early gratification and stop actively searching for defects [GL04].
  Three studies report failed attempts at gamifying software testing education. As an example, de Jesus et al. provide an experience report in which *"there was no difference regarding learning level when either traditional or gamified approaches are adopted"* [WL16].
- **Lower Efficiency**. In this macro-category, we only consider a generic drawback, i.e. *Additional Effort* required to perform the same activities when the environment, tool or methodology is gamified. Different mechanics of a gamified tool can cause overhead. Pedreira et al. highlight that setting up a gamified work environment has not a negligible cost: *"The effort (and therefore the cost) of gamifying a work environment should not be forgotten due to its importance for real organisations"* [WL03]. On the same line, de Jesus et al. report that *"building a gamified environment is a complex and incremental process, especially in the definition phase of a reward system and the ranges of scores and levels, which are related to the game mechanics and dynamics"* [WL17].

In Figure 13 we report the number of manuscripts that report at least one drawback for each of the main categories. Bad User Experience proved to be overall the most mentioned concern in the selected set of sources, with 16 mentions in WL papers and 5 in GL. The less mentioned issues were those related to a reduced effectiveness of the gamified techniques and technical issues related to the implementation of the designed mechanics. No specific drawbacks were mentioned only in grey literature sources.

4.3.5 Challenges and future research directions for gamified software testing. After the application of *Open Coding*, we came up with 22 different codes, i.e. categories of challenges and future directions discussed in the selected papers. The application of *Axial Coding* resulted in the identification of four main categories of challenges and future directions. In Table 11 we report the complete list of codes and for each code, the category, and the sources where it is mentioned.

- **Design Improvements**. Under this category, we filed challenges and future directions related to adding new gamified mechanics or improving existing ones. De Jesus et al., for instance, discuss the calibration of the difficulty of the game mechanics: *"For instance, in the current implementation, all the problem instances' difficulty has been considered equal. A possible improvement is to compute the difficulty of each problem instance and score the participants according to the difficulty of the instance that they solve correctly."* [WL10]. Another frequently required design improvement is related to the addition of graphical feedback to the gamified mechanics, which is in many cases missing due to the prototypal nature of the described tools. Many sources highlight the need for metaphorical graphical means to make the introduced mechanics

Just Accepted

Table 10. Discussed drawbacks of the application of gamified mechanics

Category	Name	Description	Mentions
Design Issues	Bad design	Problems in the game mechanics conception	[WL01], [WL26], [WL31], [GL02], [GL03], [GL04], [GL06], [GL08], [GL16]
	Cheating	Users exploiting elements of the gamified design	[WL02], [WL13], [WL32], [WL40], [WL50]
	Wrong target	Users misinterpreting the designer's focus or designer misinterpreting the users' needs	[WL02], [WL13], [WL20], [WL32], [WL35], [GL04], [GL08], [GL17]
Implementation Issues	Evaluability	Impossibility of correctly assessing the tool	[WL07]
	Limited generalisability	Tools or results cannot be scoped elsewere	[WL09], [WL30], [GL01], [GL15]
	Limited scalability	Tool does not adapt well in bigger context	[WL28], [WL32], [WL40], [GL17]
	Malicious use	User exploits the tool to invalidate its usage	[WL41]
	Privacy	Users are concerned about their data	[WL35]
Bad User Experience	Change resistance	Difficult adoption of the tool, due to resistance to change current working and learning processes	[WL07], [WL20], [WL21], [WL22], [WL28], [GL12]
	Limited cooperation	Users not interacting constructively	[WL16], [GL20]
	Limited motivation	User feeling the burden of using the tool	[WL17]
	Difficult understanding	The tool is considered cumbersome to be used by the users	[WL04], [WL22], [WL32], [GL12], [GL22]
	Discouragement	Users losing enthusiasm with the gamified environment	[WL04], [WL06], [WL24], [WL31], [WL39], [WL47], [GL01]
	Utility not perceived	Users perceiving the tool as useless	[WL22], [WL35], [GL17]
Lower Effectiveness	Decreased learning	Decrease of average learning effect in students	[WL16], [WL40], [GL20]
	Reduced performance	Decrease of ability of fulfilment of a goal	[WL06], [WL16], [WL17], [WL20], [WL39], [GL02], [GL03], [GL04], [GL16]
Lower Efficiency	Additional Effort	Users are required more time or resources to perform the same task	[WL03], [WL07], [WL13], [WL17], [WL32], [WL37], [WL47], [WL48], [WL50], [GL01], [GL07], [GL08], [GL12], [GL22]

comprehensible to the end user. In their architecture for gamification of general software engineering tasks, for instance, Pedreira et al. report that *"the engine might also be extended with a visualisation component to show, for instance, user performance and rankings. Appropriate visualisations metaphors could be used"* [WL03].

- **Implementation Improvements**. According to the selected sources, many required improvements are more related to the technical implementation of the gamified tool and frameworks than to the actual design of the game mechanics.
  The highest number of sources mentioning implementation improvements advocate a generalisable implementation that can be evaluated on variable settings.
  Several studies highlight research and implementation needs to guarantee higher scalability and dependability of the solutions and deploy the techniques as online and easily-reachable platforms to enable the collection of crowdsourced inputs. Chen and Mao, who developed Bodhi for the detection of buffer overflow in software, indicate in their agenda *"[the plan] to deploy the game on the internet to make it played by more people and detecting buffer overflows for more software"* [WL48]. In one grey literature item, it is underlined the necessity to address *"the possibility of managing synchronisation between different testing sessions carried out simultaneously on the same domain at the same time"* [GL02].
- **Evaluation**. The most frequently mentioned future directions for gamification in software testing are related to evaluating gamified mechanics. Most literature items highlight the necessity of empirical evaluations in academic settings to quantitatively assess such mechanics' theoretical and qualitative benefits.
  Four selected sources advocate for the utilisation of expert evaluation of the outcomes of gamified mechanics, especially when they allow crowdsourced contributions that need to be verified. Usfekes et al. underline that *"in the future, a development team could use a combination of known developer predictive resolution bids placed across various auctionable defects [...] This would represent a positive development for effective defect clearance through the application of gamification techniques"* [WL30].

Just Accepted

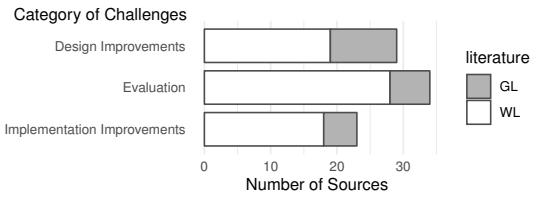

Fig. 14. Number of sources mentioning the different categories of challenges and future directions

Table 11. Discussed challenges and future directions for gamified software testing research

Category	Name	Description	Mentions
Design Improvements	Add communication	Adding instruments to make users communicate with each other	[WL02], [WL06], [GL11]
	Add competition	Addition of a resource to compete for	[GL02], [GL03]
	Add cooperation	Addition of a dynamic of collaboration between users	[GL14], [GL15]
	Ethics	Considering possible issues regarding people with some difficulties	[WL21], [GL06]
	Manage cheating	Solving design issue avoiding unexpected bad usage of the tool	[WL13]
	Narrative	Addition of a storytelling dynamic	[WL06], [WL27]
	Graphical feedback	Addition or improving the existing graphical feedback	[WL03], [WL12], [WL21], [WL35], [GL02], [GL03]
	Redesign mechanics	Addition or improving the existing game mechanics	[WL05], [WL06], [WL10], [WL12], [WL15], [WL22], [WL27], [WL28], [WL31], [WL32], [WL33], [WL36], [WL44], [WL46], [WL50], [GL02], [GL04], [GL11], [GL15], [GL20], [GL21]
	Simplification	Making the tool simpler to use	[WL11], [GL15]
Implementation Improvements	Add crowdsourcing	Implement community contribution	[WL23], [WL32]
	Decoupling	Decoupling game scenario from the code	[WL03], [WL05], [WL06], [WL28], [WL43]
	Deployment	Deploy the tool in a different way	[WL48], [WL50], [GL18]
	Generalisation	Adapt the tool to different context	[WL01], [WL02], [WL06], [WL07], [WL09], [WL12], [WL28], [WL29], [WL37], [WL49], [GL15], [GL20]
	Reusability	Application of the tool in practice	[WL06]
	Scalability	Adapt the tool to a bigger context	[WL06], [GL02], [GL03]
	Test oracles	Incorporating test oracles in the tool	[WL06], [WL45]
Evaluation	Automatic analysis	Using analysis tool to extract metrics	[WL03]
	Control results	Assessment of the obtained result	[WL13]
	Empirical studies	Additional empirical evaluation with the tool in the same or different condition	[WL01], [WL05], [WL06], [WL07], [WL10], [WL16], [WL17], [WL19], [WL24], [WL26], [WL27], [WL32], [WL33], [WL34], [WL38], [WL39], [WL41], [WL42], [WL43], [WL46], [WL49], [GL01], [GL02], [GL03], [GL05], [GL20]
	Expert evaluation	Empirical study under the supervision of expert of the field	[WL05], [WL30], [WL37], [WL45]
	Industrial case studies	Case study in an industrial context	[WL20], [WL33], [WL35], [GL16]
	Longitudinal case studies	Case study over a longer time span	[WL01]

The evaluation of gamified mechanics in real-world industrial contexts is also indicated as a primary need for the field. As Saloum et al. point out, *"it would be necessary to test the effect of gamification inside a real project in the industry where the developers act with the gamification features added to their every-day tools"* [GL16].

In Figure 14 we report the number of manuscripts that report at least one challenge or future research direction for each of the main categories. Evaluation is reported as the primary challenge and research direction in the selected literature (28 WL, 6 GL sources). The need for empirical or longitudinal evaluations was mentioned mostly in white literature sources. Fewer sources mentioned the need for design improvements (19 WL, 10 GL) or implementation improvements (18 WL, 5 GL). Two design-related needs (Add competition, and Add cooperation) were mentioned in grey literature sources only.

Just Accepted

5 DISCUSSION

In this section, we recollect and frame the results for each Review Question, and we discuss possible threats to the validity of our findings.

5.1 Summary of Findings

The main objective of our work was to identify the current state of the art and practice in the field of gamified software testing. For that purpose, we considered both white and grey literature to analyse the available gamification techniques and tools, the gamification mechanics adopted by researchers and tool developers, and the benefits, drawbacks, and future challenges. In this section, we summarize the findings that we gathered as answers to the specific review questions of the paper.

5.1.1 Literature Mapping. The first goal of our study aimed at defining a mapping of all the elicited sources. The studies were classified according to the literature category, the type of contributor (industrial vs. academic), and the kind of contribution provided to the community.

Bibliometric Trends for Gamified Software Testing. By analysing the mapping in Figure 2 we can state that gamification in software testing is a topic with a positive trend, presenting a growing number of publications over the years, from both white and grey literature. In the mapping, we see an exception with the year 2020: as a possible hypothesis for the missing grey literature items, we may mention the outbreak of the Covid-19 pandemic, and the consequent shift in working modes and communication priorities in the development and testing communities.

Our mapping also allowed us to determine how the research community in this field is distributed. The collaboration graph, shown in figure 15 (where, for the sake of clarity, we include only authors of at least two papers in our final pool) clearly shows that the different manuscripts analyzed have little to no author overlap, and therefore few collaborations are established in the field between different research groups and most prolific authors. What emerges are many isolated research groups providing autonomous contributions, with a handful of collaborations between different groups. This situation substantially differs from collaboration graphs that are reported for more mature communities (e.g., for the Graphical User Interface Testing Community [45]), confirming the relative immaturity of the field. The most significant research group is composed of G. Fraser and J. Rojas, respectively with nine and eight items produced.

Types of Contributions The results related to RQ1.1 denote that the main contributors to the state of the art are researchers from academia, who contributed five times more than industry (57 vs. 11 items) as shown in Figure 4. Industrial contributions are directed toward grey literature, while academia leaned toward the publication of white literature items. Few collaborations have been found. These data denote the fact that industry may be far from applying gamification to software testing: future research efforts should be directed toward the usage of gamified environments in industrial contexts, to execute case studies with the final purpose of validating whether the observed benefits and drawbacks, still mostly analysed in-vitro, also apply to practitioners.

More than half of the considered sources provided tool presentations or experience reports from the field. We observe a limited amount of literature items (both white and grey) providing guidelines or theoretical frameworks discussing the application of gamification to software testing, as shown in Figure 5. The effort so far has been devoted primarily to creating tools and assessing them in experimental settings. Academic literature currently lacks large-scale empirical studies able to confirm the available preliminary assessment, providing generalizable results. Current literature, especially in the educational setting, mostly provide experience reports that are gathered from ad-hoc settings and are sometimes strongly contradicting (RQ1.2). Future research in this regard should provide evidence on how the results obtained in case studies and reports can be reproduced to achieve the

Just Accepted

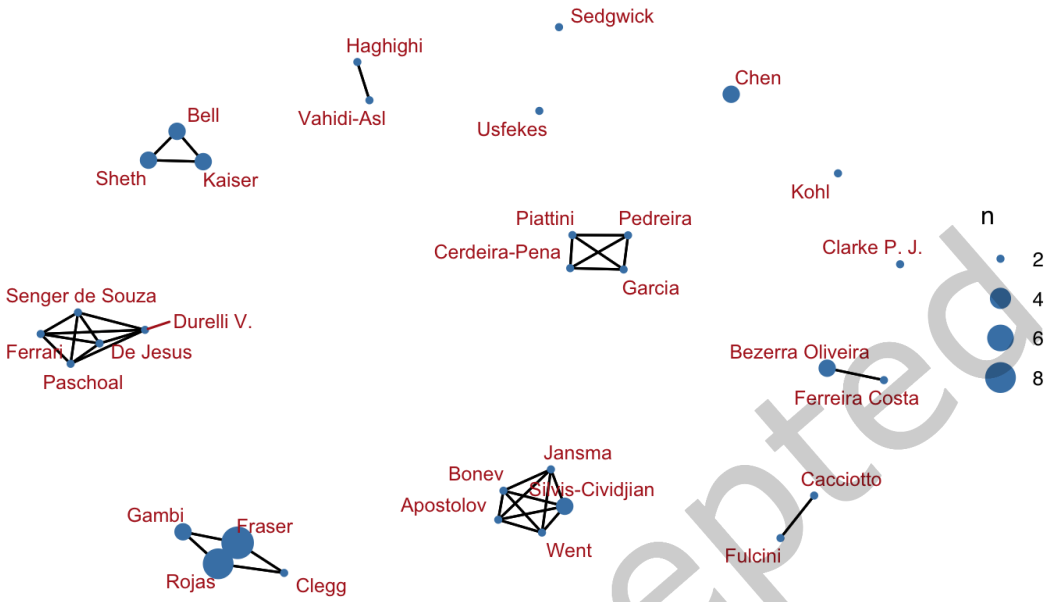

Fig. 15. Collaboration between authors (n>2) of the collected literature

claimed benefits, with practical suggestions on how to apply a successful gamified environment, and replication of experiments in order to actually demonstrate the positive impact of gamification. Future research could also focus on extracting common and universal sets of guidelines for developers of gamified environments for testing practice and education.

5.1.2 Testing-Focused Characterization. The second goal of our study aimed at identifying the characteristics of the software testing activities that are most commonly augmented with gamified components.

Gamified testing levels, phases, and methodologies. The results to review question 2.1 highlight that – currently – gamified techniques are applied mostly to Unit-testing tools, being used in more than half of the considered studies. Next are System and Integration testing tools, both with a significantly smaller number of documented applications. Some items, especially those dealing with software testing education, explicitly mentioned more than one testing level, others neither mentioned, nor intended, any level; this could be explained by the sources discussing a general approach that may be applied to different testing levels. Creating a gamified environment related to integration testing or even system testing appears to be more complex because those testing levels require the incorporation of libraries, third-party code, or more difficult deployment for the solution than just the integration within an existing unit testing engine. All those additional aspects constitute a significant execution and building overhead, possibly driving away investigators. Future research should address those higher testing levels, rather than just focusing on unit testing and its variations.

Just Accepted

We can identify five distinct process stages related to testing: design, creation, execution, reporting, and maintenance. As the results of our review question 2.2 suggest, gamified mechanics are mainly implemented in the phase of test creation and execution, whilst a limited number of applications of gamified mechanics have been provided for other software testing activities, e.g. test reporting, test maintenance, or test design. The two most frequent phases are the most operative ones, which are repeated several times during the iterative process. When test creation is automated, the creation phase involves the definition and implementation of test scripts, while in manual testing the creation typically overlaps with test execution in the exploration phases of the SUT. Given these premises, it seems natural that the effort has been directed primarily at these two phases. However, we believe that further study is needed especially for test maintenance and test reporting activities. Especially the former, in fact, has been identified by many literature items as crucial and costly in the testing pipeline. We, therefore advocate further investigations in gamifying testing activities different than test creation and execution.

Regarding the most frequently mentioned test methodologies (RQ2.3), we find that gamified activities are most frequently applied in tasks that are either simple (like mutation, black-box, and white-box testing at unit-level) or do not require significant programming skills (e.g., manual and Capture & Replay testing at the system level). The presented distribution may serve as another confirmation of the novelty of the gamified approach, mainly applied to traditional white-box and black-box testing activities. Mutation testing is an exception in this respect but, even if it has been mentioned in fifteen papers, nine of them are from the author pair Fraser and Rojas, dealing with the evolution of the same tool, i.e. Code Defenders. The importance of their work is so great that we can consider the two authors as the main investigators in the scope of this research work (as better explained in subsubsection 5.2.1). Future research works should be directed towards other less exposed techniques, aiming at identifying other fertile ground for gamification to be applied.

Tools and languages for gamified testing. The results for RQ 2.4 show that gamification was applied mostly to open-source unit-testing tools. Besides the vast majority of the items (46 out of 73) did not specifically mention any tool in their discussion, the data shows how widespread is the usage of JUnit. This result is highly justifiable since JUnit can be considered a de-facto standard test runner for any testing activity to be performed on SUTs written in Java. Another commonly mentioned tool in the elicited work is the Capture & Replay testing tool Scout. What emerges from the results is a highly fragmented situation: apart from JUnit, each research group used a specific tool. The interpretations of this result can be manifold: (i) researchers that are dealing with low-level and simple testing activities can build upon JUnit to implement gamified mechanics; (ii) researcher dealing with higher-level and more complex testing activities (e.g., GUI testing or system-level testing) rely on building completely new testing environments to implement gamified mechanics. We, therefore, stress the need for research for generic, possibly open-source, gamification frameworks that would allow the introduction of gamification in more complex scenarios. The presence of easily implementable and generalisable gamification frameworks may also fuel comparative and quantitative empirical research across multiple settings and different SUTs to better assess the benefits and drawbacks of gamified practices.

Regarding languages (RQ 2.5), the vast majority of the collected studies provide solutions that adopt Java as the target programming language. This data is in accordance with the previous review question, highlighting JUnit as the most used tool, and with other studies affirming Java as the most used language for test scripting [2]. Regarding the target testing domain, few data are available. Actually, only a few items provide specifications about the domain of application of the gamified testing tool: this data can be explained as the studies are mainly focused on unit testing, at this level the code under test is agnostic of any domain and it can be used at low-level for any application context (e.g., desktop, mobile, or web). The studies explicitly reporting a domain suggest an almost even distribution between mobile and web, with the exception of a single IoT-related source. New contributors should consider the exploration of gamified testing in the embedded and IoT domain, which remains

Just Accepted

an unexplored topic for the practice.

Practice vs. Education. The results collected for RQ2.6 show a prevalence in white literature sources for the application of gamification in educational settings. Researchers' interest in applying gamification in educational contexts is not new, indeed there is a firm and shared belief in the literature that students (especially adolescents and pre-adolescents) are likely to enjoy a gamified environment due to their familiarity with games [7]. Industrial approaches to gamified software testing are almost equally distributed between white and grey literature, meaning that practitioners are more interested in the application of game elements to support the practical testing process, rather than using them in the testing learning process. Only a few are the proposed approaches that can be used both to teach how to test and to conduct testing in industrial settings. We look forward to future research works to explore the mixed approach, proposing tools, frameworks, and guidelines able to support both test education and practice.

5.1.3 Gamification-Focused Characterization. The third Goal of the study had the objective of providing a characterisation of the most frequently adopted and evaluated gamification mechanics and tools in both white and grey literature.

Game mechanics and Gamification tools. By performing a meta-analysis on the collected data, before applying the synthesis process, we observed two main trends: firstly, the collected literature items do not adopt an unambiguous definition of game elements, in fact in most of the literature items there is no distinction between game dynamics and mechanics. Thus, we categorized all the mentioned game elements as *game mechanics* for our discussion. In future dissertations, we suggest adopting the more strict game elements classification provided by Robson et al. [44] to distinguish between game mechanics adopted and game dynamics stimulated.

Secondly, we observed that only a few items employed a standardised ad-hoc method to build or to evaluate a gamification tool[3]. We highlight the fact that, being a gamified tool highly dependent on the perception of its users, its evaluation should take into account the user experience as a whole (some examples are Octalysis [10] and GAMEX[19]). This implies the selection of the correct evaluation method for a fair assessment of the developed tool, and the usage of frameworks helping in the selection of the right game elements to build a balanced and successful gamification environment.

After performing a mapping of gamified mechanics, we observe that few stood out significantly with respect to others. We also adopted the classification provided by the Octalysis framework to aggregate the found game mechanics, this allowed us to discover trends and lacks of existing gamification design. The main trend is to adopt elements based on the *Accomplishment* core drive, providing positive feedback to the user. Although this trend is not negative in itself, the negative aspect is the lack of a counterbalance in *Avoidance* elements which make the game experience quite unbalanced in most cases. *Epic Meaning* core drive is also quite often neglected, with almost no narrative layer in the tools. . Since gamification is mainly designed to increase engagement and to make testing activities less stressful and redundant, we argue that the addition of *black-hat* or punishment-related aspects of gamification has not been yet considered by the authors developing gamified approaches, but since there is no evidence of its uselessness, future research effort should try to incorporate them.

Even though the gamified approach can still be considered in its infancy for software testing research, we found 30 distinct tools in the collected literature. It is worth, however, mentioning that the level of maturity of the tools is highly variable and ranges from mature tools already empirically evaluated in multiple settings (e.g., CodeDefenders) to pure academic prototypes and demonstrators (*RQ3.2*).

[3]see the ResearchQuestion sheet at https://doi.org/10.6084/m9.figshare.19804147

Just Accepted

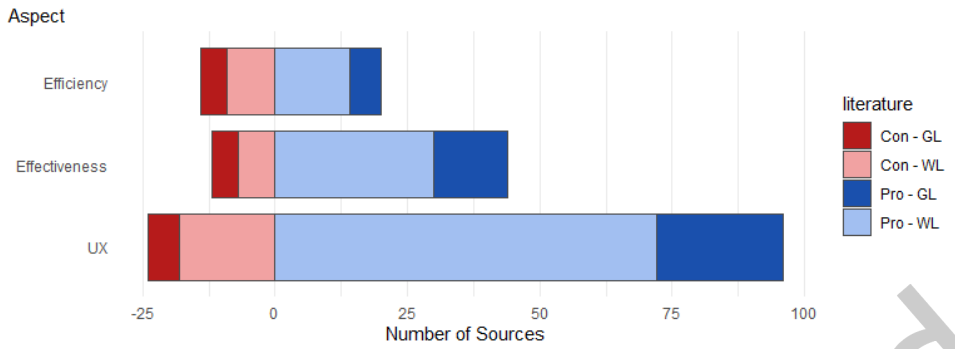

Fig. 16. Number of conflicting benefits and drawbacks of gamified testing by category

Pros and Cons of Gamification. Many studies among the collected sources analysed the benefits and drawbacks introduced by gamification in the testing practice. Among the benefits and drawbacks, we identified three factors that were mentioned in both directions: user experience, effectiveness, and efficiency. When discussing the drawbacks, the sources also mentioned difficulties in design and implementation, which were not mentioned explicitly among the benefits. Since the sources often provided little details on the analyzed case studies, to provide a general balance between pros and cons, we simply counted the number of sources that reported the given common factors as either benefits or drawbacks. Figure 16 summarizes graphically the number of mentions for each of the factors, by type of literature. A better User Experience is the most reported benefit, with 96 claims in this regard, while only 24 papers mention bad user experience as a drawback of gamified testing. The same kind of proportion is also found regarding the effectiveness of a gamified environment (44 claims in favor, 12 against). By contrast, more discordant opinions are found regarding efficiency (20 claims in favor, 14 against).

These findings show that there is little disagreement about the better User Experience obtained when gamification is applied to software testing. Negative opinions can also be observed in the literature: this contrast should not discourage researchers since, being Gamification a user-centered technique, it can be considered normal that the way it is perceived can vary in different studies, domains, and practices. A higher disagreement is found when considering the enhancement of effectiveness provided by Gamification. We attribute this disagreement to the dependence of the measurement of effectiveness in the practical domain where Gamified testing is applied (e.g., effectiveness might refer to bugs found, coverage reached, test cases produced, and so on). In this respect, we highlight the need for a unified metrics framework to assess effectiveness.Positive opinions overcome negative opinions also for efficiency - however in this aspect opinions are even more contrasting. We argue that building and maintaining a gamified environment typically constitutes a significant overhead in the software testing process, and it is reasonable that no unequivocal opinion can be found about such necessary overhead. We, therefore, highlight the need for further investigations into the efficiency provided by Gamification in future research activities.

Future Challenges for Gamification. Among the future challenges for academic researchers and industry players in the field of software testing, there is a general consensus about the need for a more thorough evaluation and experimentation of gamified mechanics in real-world settings. These evaluations are required since often the reported results refer to preliminary evaluation. Experts are also required to be involved in such validations, both to determine if the effects observed with students also apply to practitioners, and to clarify if the results reached are considered valid by the testers' community. The category Design improvements denotes the fact

Just Accepted

that tools are often developed iteratively by learning from the errors, without a systematic approach. Differently, Implementation improvements show that, even if many attempts have positive results, there is room for features and variations of gamified software testing in other contexts and domains.

Future research and practical direction regarding this topic should include:

- The definition of a set of unified and unambiguous metrics to properly assess the effectiveness and efficiency of subjects, considering all the possible aspects of software testing (i.e. using the same metric when assessing one dimension in the same way).
- Longitudinal experiments, in order to ascertain how exposing testers to gamified environments over a longer period of time affects the benefits found on their first experience.

.

5.1.4 White Literature vs. Grey Literature. Comparing efforts from academia and industry in the field of gamified software testing can be performed in two ways: firstly, comparing white literature items (mostly associated with the academia) with grey literature items (mostly including industry-related content). Secondly, comparing theoretical with practical dissertations.

The comparison between white and grey literature does not show notable differences, as from the results we can state that researchers and practitioners follow roughly the same trend. Regarding test phases, both WL and GL focus mostly on test creation and test execution; the test levels that are addressed the most are almost equally unit and system testing; also the adopted gamification aspects follow a similar distribution.

The main difference that we observe between WL and GL items concerns the relative proportion of two mechanics categories: social influence and unpredictability; the former being relatively more common in WL and the latter in GL. Another minor difference that we highlight is in the testing methodologies adopted, where exploratory testing and capture and replay have been considered more for practical usage than for research. Considering Grey Literature in our final pool of literature items, we included three tools, one possible advantage of gamification, and two future challenges that were not signalled in white literature.

By analysing the type of literature items discussing gamified software testing (either WL or GL) we notice that the main types of contributions consist of tool presentations and experiments. Literature items providing guidelines and frameworks are scarce, and typically provide experience-based analyses and not empirical results. General discussions are mainly found in grey literature, hypothesizing possible benefits or providing clues on how to apply gamification to testing activities. All this evidence supports our conclusion that the application of gamification to testing is still in an immature phase, without consolidated practice or well-rooted empirical evidence.

5.1.5 Comparison with existing secondary studies. The results and discussion presented in this paper expand the existing state of the art, by complementing the outcome of previous similar secondary studies. In particular, with respect to Mäntylä and Smolander [36], we included fifty more recent additional sources. This allowed us to find a larger set of game elements; we provided a deeper testing-related characterization by decomposing testing into seven dimensions. We also found a meta-analysis of the discussed findings and challenges, providing updated clues for those who will approach the topic in the future.

Comparing our work to de Jesus et al. [14], we included 58 additional literature items containing also grey literature, thanks to the multivocal nature of our study and the usage of both backward and forward snowballing activities, which were missing in [14]. We expanded the authors' testing characterization from three to seven testing dimensions (including the testing tools used, the target language, the domain, and the goals, whether practical or educational), providing a new point of view and new data to be analyzed. Our game characterization included 13 more game elements, plus a link to Octalysis' core drives. Another difference is that they focused

Just Accepted

on the goals that each paper had in applying gamification to testing, while we considered the stated outcomes grouped in benefits and drawbacks, considering also the open challenges presented by authors.

Our results include several new tools and frameworks that were not covered by previous secondary studies. However, we notice that most of the contributors of such tools and frameworks have a practical focus and provide few empirical results and validations; as well as they lack theoretical discussion required to establish consolidated methodologies for gamified software testing. With the current manuscript, we expand the previously presented views on the currently-available instrumentation for gamified software testing, hopefully providing inputs for future comparative investigations.

5.2 Threats to Validity

5.2.1 Threats to Construct validity. Threats to Construct validity for a Literature Review concern possible issues in reaching full coverage of the spectrum of all the studies related to the chosen topic. This study mitigated that threat by identifying five essential sources of white literature studies that we required to be present in the retrieved pool of sources coming from the used search string in the different repositories. Namely, we selected the following items of literature:

- *Gamification of software testing* by Fraser et al. [20]
- *Is It Worth Using gamification on Software Testing Education? An Experience Report* by De Jesus et al. [WL16]
- *Code Defenders: A Mutation Testing Game* by Rojas et al. [WL05]
- *A framework for gamification in software engineering* by Garcia et al. [WL01]
- *Gamification-based cyber-enabled learning environment of software testing* by Fu et al. [WL13]

We decided to train our search string iteratively until the essential sources were found. This process required only one iteration. Despite its importance, one of the essential sources was not included in the final pool of sources due to the exclusion criterion allowing only primary studies in the selection. This does not represent a threat, as these studies were used to test the search strings.

Grey literature was also considered to include gamification tools, gamification frameworks, and related empirical evaluations that are not presented in peer-reviewed papers.

For both white and grey literature, we applied a reproducible methodology based on established guidelines. To broaden the research as much as possible, we included the most commonly used terms in the search strings, as well as the main synonyms already used in other SLRs, as seen in section 2.4. However, it is still possible that some terms describing other relevant works in the literature may have been overlooked. Regarding grey literature, there is a possibility that literature items about relevant tools and frameworks have not been included in the analysis because of the inability to access the documents.

5.2.2 Threats to Internal validity. Threats to Internal validity are those related to the data extraction and synthesis phases of the Literature Review. All the primary sources resulting from the search strings application were read and evaluated by all authors and collaborators of this study to assess their quality. When in doubt about the decision to include a resource or not, the final decision was taken by a majority vote. The authors also applied inclusion and exclusion criteria and extraction the information to answer the Review Questions of the study. Hence, the validity of the study is threatened by possible errors in the author's judgment when examining the sources and/or misinterpretations of the original content of the papers. This threat was mitigated by multiple readings of the same sources and discussion among the authors about potential disagreements during the review phase.

5.2.3 Threats to External validity. Threats to External validity concern the generalisability of the findings of the Literature Review. We scoped our research to gamification applied to software testing. We also included in our search literature items in the general field of Software Engineering, but that showed clear applicability to software

Just Accepted

testing. We can not estimate the generalisability of our findings to the application of gamification to software engineering activities other than software testing. As well, very specific gamification tools and mechanics, testing tools or domains can exhibit benefits and drawbacks that have been not found in our analysis.

6 CONCLUSION AND FUTURE WORK

In the present work we defined, conducted, and documented the results of a Multivocal Systematic Literature Review applied to the field of gamification in software testing. The literature items considered were published between 2010 and the first quarter of 2022. We provide three main contributions: a mapping of the type of sources available in the literature; a characterisation of the testing-focused gamified tools; a game-related characterisation with the goal of describing all the gamified mechanics mentioned in literature and their benefits and drawbacks.

Our literature search led us to collect a total of 73 studies (50 from white literature and 23 from grey literature), bridging the gap with past secondary studies dated 2016 [36] and 2018 [14].

Based on our findings, discussed in section 5, we can state that gamification represents a promising direction in the field of software testing for both research and industry. Well-established gamification frameworks have already been defined in other disciplines, and preliminary studies and experience reports in the field of software testing have proven that its utilisation can lead to significant increases in the engagement and productivity of software testers. Gamification is also considered a valid instrument in software testing education.

Gamified mechanics, however, are by no means a silver bullet; evidence from the literature highlights the need for careful calibration of the mechanics to implement. Special care must be taken to avoid misalignment with the main purposes of the testing activities, exploitation by the *players*, and negative impacts on the efficacy or efficiency of the gamified tasks.

On top of our findings, we identify a set of actionable guidelines for different stakeholders interested in software testing gamification, that we summarise as follows:

- **Researchers** in the software testing field may consider the possibility of performing rigorous empirical assessments of the benefits introduced by the application of gamified mechanics to existing tools and practices. Longitudinal case studies are encouraged to verify the transferability of the techniques to the industrial context;
- **Developers** of software testing tools should carefully evaluate the mechanics they wish to adopt and they should follow existing frameworks of mechanics to provide a balanced gamified experience for software testers. Gamification-based approaches should be designed sensibly to match the needs of the specific technique and to prevent exploitation by the users with the ensuing possible reduction of effectiveness;
- **Educators** are highly recommended to incorporate gamified approaches in software testing education to support the execution of activities during the learning process since evidence in the literature suggests a strong motivating impact of game-like mechanics when adopted in software testing classes.

Just Accepted

REFERENCES

[1] 2005. *ISO 9001:2005 - Quality management systems - Requirements.* Standard. International Organization for Standardization.

[2] Shamsu Abdullahi, Abubakar Zakari, Haruna Abdu, Amina Nura, Musa Ahmed Zayyad, Salisu Suleiman, Alhassan Adamu, and Abdulfatahu Samaila Mashasha. 2020. Software Testing: Review on Tools, Techniques and Challenges. *International Journal of Advanced Research in Technology and Innovation* 2, 2 (2020), 11–18.

[3] Richard Adams, Palie Smart, and Anne Sigismund Huff. 2017. Shades of grey: guidelines for working with the grey literature in systematic reviews for management and organizational studies. *International Journal of Management Reviews* 19, 4 (2017), 432–454. https://doi.org/10.1111/ijmr.12102

[4] Carlos Futino Barreto and César França. 2021. Gamification in Software Engineering: A literature Review. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 105–108. https://doi.org/10.1109/CHASE52884.2021.00020

[5] Fevzi Belli, Nimal Nissanke, Christof J Budnik, and Aditya Mathur. 2005. Test generation using event sequence graphs. *University of Paderborn, Institute for Electrical Engineering and Information Technology* (2005).
[6] Stefan Berner, Roland Weber, and Rudolf K Keller. 2005. Observations and lessons learned from automated testing. In *Proceedings of the 27th international conference on Software engineering*. 571–579.
[7] Jenny Bittner and Jeffrey Schipper. 2014. Motivational effects and age differences of gamification in product advertising. *Journal of Consumer Marketing* 31 (08 2014), 391–400. https://doi.org/10.1108/JCM-04-2014-0945
[8] Andreas Bruns, Andreas Kornstadt, and Dennis Wichmann. 2009. Web application tests with selenium. *IEEE software* 26, 5 (2009), 88–91.
[9] Ilaria Caponetto, Jeffrey Earp, and Michela Ott. 2014. Gamification and education: A literature review. In *European Conference on Games Based Learning*, Vol. 1. Academic Conferences International Limited, 50.
[10] Yu-Kai Chou. 2015. *Actionable Gamification: Beyond Points, Badges, and Leaderboards*. Createspace Independent Publishing Platform. https://books.google.it/books?id=jFWQrgEACAAJ
[11] Peter John Clarke, Andrew Allen, Tariq King, Edward Jones, and Prathiba Natesan. 2010. Using a Web-Based Repository to Integrate Testing Tools into Programming Courses. In *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion (OOPSLA '10)*. Association for Computing Machinery, 193–200. https://doi.org/10.1145/1869542.1869573
[12] Peter John Clarke, Debra Davis, Tariq King, Jairo Pava, and Edward Jones. 2014. Integrating Testing into Software Engineering Courses Supported by a Collaborative Learning Environment. *ACM Trans. Comput. Educ.* 14, 3, Article 18 (oct 2014), 33 pages. https://doi.org/10.1145/2648787
[13] Andrei Contan, Catalin Dehelean, and Liviu Miclea. 2018. Test automation pyramid from theory to practice. In *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*. IEEE, 1–5.
[14] Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, Daniel de Paula Porto, and Sandra Camargo Pinto Ferraz Fabbri. 2018. Gamification in Software Testing: A Characterization Study. In *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing (SAST '18)*. Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/3266003.3266007
[15] Anca Deak, Tor Stålhane, and Guttorm Sindre. 2016. Challenges and strategies for motivating software testing personnel. *Information and Software Technology* 73 (2016), 1–15. https://doi.org/10.1016/j.infsof.2016.01.002
[16] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. c. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11)*. Association for Computing Machinery, New York, NY, USA, 9–15. https://doi.org/10.1145/2181037.2181040
[17] Arilo C Dias Neto, Rajesh Subramanyan, Marlon Vieira, and Guilherme H Travassos. 2007. A survey on model-based testing approaches: a systematic review. In *Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*. 31–36.
[18] Michael Ellims, James Bridges, and Darrel C Ince. 2006. The economics of unit testing. *Empirical Software Engineering* 11, 1 (2006), 5–31.
[19] René Eppmann, Magdalena Bekk, and Kristina Klein. 2018. Gameful Experience in Gamification: Construction and Validation of a Gameful Experience Scale [GAMEX]. *Journal of Interactive Marketing* 43 (2018), 98–115. https://doi.org/10.1016/j.intmar.2018.03.002
[20] Gordon Fraser. 2017. Gamification of Software Testing. In *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*. 2–7. https://doi.org/10.1109/AST.2017.20
[21] Gabriel García-Mireles and Miguel Ehecatl Trujillo. 2020. *Gamification in Software Engineering: A Tertiary Study*. 116–128. https://doi.org/10.1007/978-3-030-33547-2_10
[22] Vahid Garousi, Michael Felderer, and Tuna Hacaloğlu. 2017. Software test maturity assessment and test process improvement: A multivocal literature review. *Information and Software Technology* 85 (2017), 16–42. https://doi.org/10.1016/j.infsof.2017.01.001
[23] Vahid Garousi, Michael Felderer, Marco Kuhrmann, and Kadir Herkiloğlu. 2017. What industry wants from academia in software testing? Hearing practitioners' opinions. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. 65–69.
[24] Vahid Garousi, Michael Felderer, and Mika Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology* 106 (2019), 101–121. https://doi.org/10.1016/j.infsof.2018.09.006
[25] Vahid Garousi, Michael Felderer, and Mika Mäntylä. 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and Software Technology* 106 (2019), 101–121. https://doi.org/10.1016/j.infsof.2018.09.006
[26] Vahid Garousi and Mika Mäntylä. 2016. When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology* 76 (2016), 92–117. https://doi.org/10.1016/j.infsof.2016.04.015
[27] Spencer E Harpe. 2015. How to analyze Likert and other rating scale data. *Currents in pharmacy teaching and learning* 7, 6 (2015), 836–850.
[28] Mary Jean Harrold. 2000. Testing: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE '00)*. Association for Computing Machinery, New York, NY, USA, 61–72. https://doi.org/10.1145/336512.336532
[29] Itti Hooda and Rajender Singh Chhillar. 2015. Software test process, testing types and techniques. *International Journal of Computer Applications* 111, 13 (2015).

Just Accepted

[30] Juha Itkonen, Mika V Mantyla, and Casper Lassenius. 2009. How do testers do it? An exploratory study on manual testing practices. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 494–497.
[31] Samireh Jalali and Claes Wohlin. 2012. Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the 2012 ACM-IEEE international symposium on empirical software engineering and measurement*. IEEE, IEEE Computer Society, Los Alamitos, CA, USA, 29–38. https://doi.org/10.1145/2372251.2372257
[32] Joseph Kiniry and Daniel Zimmerman. 2008. Secret Ninja Formal Methods. In *FM 2008: Formal Methods*. Springer Berlin Heidelberg, 214–228. https://doi.org/10.1007/978-3-540-68237-0_16
[33] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).
[34] Richard N Landers, Elena M Auer, Andrew B Collmus, and Michael B Armstrong. 2018. Gamification science, its history and future: Definitions and a research agenda. *Simulation & Gaming* 49, 3 (2018), 315–337.
[35] Mario Linares-Vásquez, Kevin Moran, and Denys Poshyvanyk. 2017. Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 399–410. https://doi.org/10.1109/ICSME.2017.27
[36] Mika Mäntylä and Kari Smolander. 2016. Gamification of Software Testing - An MLR. In *Product-Focused Software Process Improvement*. Springer International Publishing, 611–614. https://doi.org/10.1007/978-3-319-49094-6_46
[37] Gursimran Singh Walia Mourya Reddy Narasareddy Gari and Alex David Radermacher. 2018. Gamification in Computer Science Education: a Systematic Literature Review. In *2018 ASEE Annual Conference & Exposition*. ASEE Conferences, Salt Lake City, Utah. https://peer.asee.org/30554.
[38] Lennart E Nacke and Christoph Sebastian Deterding. 2017. The maturing of gamification research. *Computers in Human Behaviour* (2017), 450–454.
[39] Stanislava Nedyalkova and Jorge Bernardino. 2013. Open source capture and replay tools comparison. In *Proceedings of the International C* Conference on Computer Science and Software Engineering*. 117–119.
[40] Nicholas O'Donnell, Dennis Kappen, Zachary Fitz-Walter, Sebastian Deterding, Lennart Nacke, and Daniel Johnson. 2017. How Multidisciplinary is Gamification Research? Results from a Scoping Review. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '17 Extended Abstracts)*. Association for Computing Machinery, New York, NY, USA, 445–452. https://doi.org/10.1145/3130859.3131412
[41] Rodney Ogawa and Betty Malen. 1991. Towards rigor in reviews of multivocal literatures: Applying the exploratory case study method. *Review of educational research* 61, 3 (1991), 265–286. https://doi.org/10.3102/00346543061003265
[42] Oscar Pedreira, Félix García, Nieves Brisaboa, and Mario Piattini. 2015. Gamification in software engineering – A systematic mapping. *Information and Software Technology* 57 (2015), 157–168. https://doi.org/10.1016/j.infsof.2014.08.007
[43] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *EASE'08*. BCS Learning & Development Ltd., Swindon, GBR, 68–77. https://doi.org/10.14236/ewic/EASE2008.8
[44] Karen Robson, Kirk Plangger, Jan Henrik Kietzmann, Ian McCarthy, and Leyland Pitt. 2015. Is it all a game? Understanding the principles of gamification. *Business Horizons* 58, 4 (2015), 411–420. https://doi.org/10.1016/j.bushor.2015.03.006
[45] Olivia Rodríguez-Valdés, Tanja EJ Vos, Pekka Aho, and Beatriz Marín. 2021. 30 years of automated GUI testing: a bibliometric analysis. In *International Conference on the Quality of Information and Communications Technology*. Springer, 473–488.
[46] Richard Ryan and Edward Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist* 55, 1 (2000), 68. https://doi.org/10.1037/0003-066X.55.1.68
[47] Joachim Schöpfel and Dominic Farace. 2010. Grey Literature in Library and Information Studies. *Encyclopedia of Library and Information Sciences* (2010), 2029–2039.
[48] Swapneel Sheth, Jonathan Bell, and Gail Kaiser. 2011. HALO (Highly Addictive, Socially Optimized) Software Engineering. In *Proceedings of the 1st International Workshop on Games and Software Engineering (GAS '11)*. Association for Computing Machinery, New York, NY, USA, 29–32. https://doi.org/10.1145/1984674.1984685
[49] Joanna Smith, Joe Tessler, Elliot Kramer, and Calvin Lin. 2012. Using peer review to teach software testing. In *Proceedings of the ninth annual international conference on International computing education research*. 93–98.
[50] StackSocial. 2014. The most popular coding language at top US universities. http://blog.stacksocial.com/popular-coding-language/ Accessed: 2022-10-5.
[51] Philipp Straubinger and Gordon Fraser. 2022. Gamekins: Gamifying Software Testing in Jenkins. abs/2202.06562 (2022), 85–89. https://doi.org/10.1145/3510454.3516862
[52] Mark Utting and Bruno Legeard. 2010. *Practical model-based testing: a tools approach*. Elsevier.
[53] James A Whittaker. 2009. *Exploratory software testing: tips, tricks, tours, and techniques to guide test design*. Pearson Education.

Just Accepted